UNIVERSITÁ DEGLI STUDI DI MILANO – BICOCCA

Scuola di Economia e Statistica

Corso di Laurea in Statistica e Gestione delle informazioni

AUTOMATED ANALYSIS AND VISUALIZATION TECHNIQUES OF DATA GENERATED WITH FLOW CYTOMETRY PLATFORMS

Relatore: Prof. Vincenzo Bagnardi

Correlatore: Dott. Luca Lalli

Tesi di Laurea di:

Andrea Lucini Paioni

Matr. N. 82657

Anno Accademico 2020/21

INDEX

SUMMARY OF CHARTS

ABSTRACT

The creation, development and diffusion of flow cytometer and flow cytometry techniques arise from the need to address some complex problems, related to a faster collection and an easier analysis of great amount of data concerning different branches of biology: from the analysis of blood cells, to cell cycle studies, from research about apoptosis phenomena to cancer clinical trials for both diagnosis and prognosis purposes.

During the experience I made as an intern in the IRCCS National Cancer Institute, I could see firsthand some of the progress made possible by flow cytometry techniques, both for data-exploring and analysis purposes.

Along with my tutor Dott. Luca Lalli, and with the supervision of Dott. Luigi Mariani, I was able to face biomedical problems with statistical approaches, not only with computer applications utilizing the software R, but also through research work on scientific articles and collaborations with both statistical and medical staff.

This work is divided in two parts: the first, more theoretical one, is a focus on different methods for dimensionality reduction on data generated by flow cytometry – such as t-SNE algorithms and the UMAP plots –, with possible applications leading to clustering procedures – like K-means methods, the Self-Organizing-Map approach and the Density-Based Spatial Clustering of Applications with Noise, or DBSCAN –; the second part is a practical approach to the methods analyzed in the first part, with the purpose to find the most efficient, meaningful and relevant ways to work with a biomedical database obtained with flow cytometry techniques.

INTRODUCTION TO FLOW CYTOMETRY

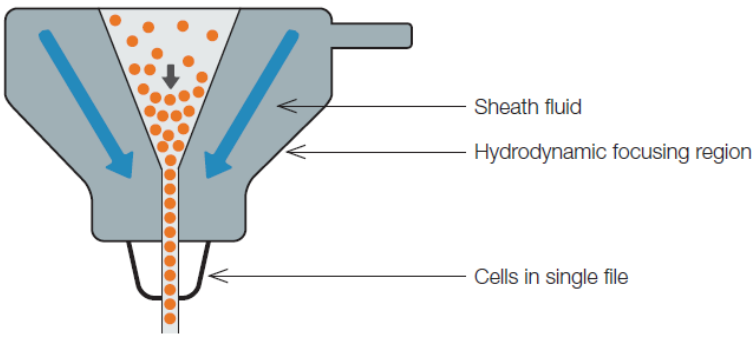
The flow cytometer

Flow cytometry is a widely used laboratory method that provides useful data with the purpose of analyzing physical and chemical properties of particles, distributed in samples flowing past multiple lasers. It's a very common technique in biomedical studies, with many possible applications, such as the detection and characterization of microbial cells, or the definition of population of interest within cells.

It's commonly used for immunophenotyping, crucial in diagnosing oncologic phenomena from different biological tissues, but also a very useful method to quickly collect quantitative data from great amount of cells, for cell sorting trials, analysis about cell cycles or the identification of apoptosis among cell deaths.

A flow cytometer basically consists of:

- a flow cell, in which the cell sample is injected and afterwards distributed in a 3D space larger than the cells themselves;
- the fluidics system, that allows the creation of an ordered stream of cells, thanks to the hydrodynamic focusing created by the narrowing of the sheath fluid;
- the interrogation point, where the particles, now distributed in a single line, encounter a beam of focused light;
- the detectors, where the pulses generated in the interrogation point are collected;
- a computer, which converts these pulses into outputs that can be collected, analyzed and turned into useful information.

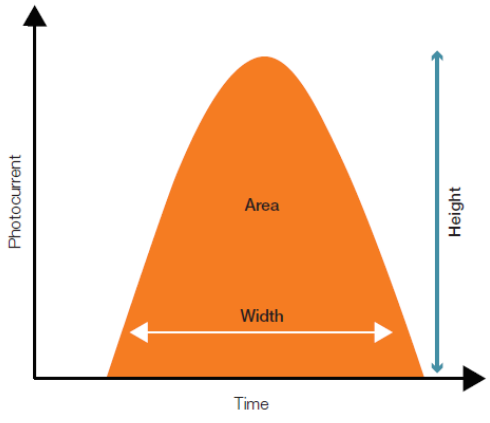


The collecting system is formed by different detectors: the light scattered in a forward direction is directed into the so-called FSC, or Forward Scatter Channel. The FSC usually collects information about the size of the particles, and this channel is usually set up with a 20° offset from the axis of the laser beam, but the slope is different between each type of flow cytometer, depending on the purpose of the machine.

Another type of Channel is the SSC, or Side Scatter Channel, with a 90° angle from the laser's axis: with this channel, we can collect information about granularity and different characteristics of the particles. In addition, different channels can be used together to acquire several types of information in just one flow of particles: in order to do this, different types of filters are used to allow lights just at specific wavelengths to pass through and return information. The dichroic filters, for example, are able to differentiate lights at a chosen wavelength to have both an FSC and an SSC with the same laser.

The stream of particles is analyzed thanks to the light scattered from the interaction between the laser and every single particle, and it is collected by detectors like the PMT – or PhotoMultiplier Tubes – or the APD – or Avalanche PhotoDiodes – . These detectors are able to collect information from the number of photons that hit the photocathode, and these "events" produce information, useful for sorting decisions, comparing different populations or measuring the intensity of the fluorescence.

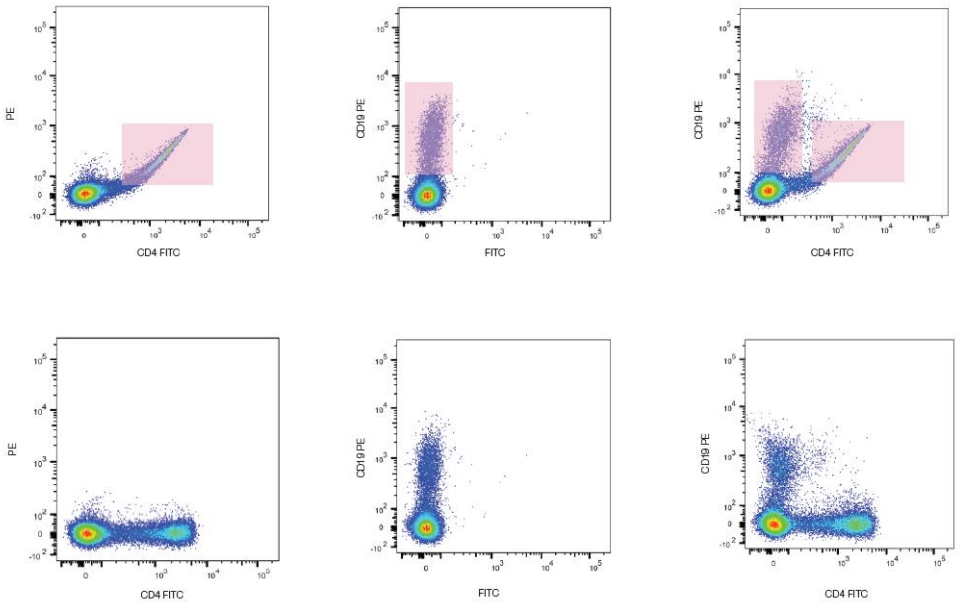
The output of these events can be plotted and quantified, so that every density plot provides information about the maximum amount of current – which is the height of the plot –, the time interval during which the pulse occurs – which is the width, useful to understand the nature of every cell, from single particles, to particles interacting with each other, to doublets –, and the integral of the pulse – which is the area –.



Fluorophores, the spillover matrix and compensation

Useful to obtain the expression of different kinds of molecules, the fluorophores are able to absorb light at a given wavelength – in the part of the process called "excitation" – and then re-emit this light at a higher wavelength – the so-called "emission" –. Between these two phases of the process, following a conformational change, the remaining energy is released as fluorescence, resulting in different colors at the beginning and at the end of the excitation-emission cycle.

When processing with two or more fluorophores at the same time, there could be some disturb among them, affecting the result collected by the flow cytometer with the so-called spectral overlap between fluorophores. This phenomenon can be avoided by considering a mathematical method that creates a matrix named Spillover matrix: this is a square matrix, obtained by estimating – in percentage – how big the spillover from each fluorophores to every other one is.



The Compensation matrix, similar to the Spillover matrix, can be obtained with simple mathematical operations on the Spillover Matrix: having both of these matrix is fundamental to measure the true signal of every fluorophore, allowing us to receive data already compensated, so that no fluorophore interferes with each other.

It is also possible to avoid this kind of problem, by choosing fluorophores without overlapping for the emission spectra; however, this decision has his downsides, mostly because it would reduce the amount of data we can collect from a single experiment, costing more in terms of time and resources.

Both of these matrix are available in R: they can be called, respectively, with the command *flowFr_name@description$`$SPILLOVER`*, for the Spillover Matrix, and by inverting the Spillover Matrix, with the command *solve(spill_matrix)*, for the Compensation matrix.

Gating

The detectors in the flow cytometer are really sensitive, and they can obtain information about non relevant events to the biomedical experiment: in this case, a threshold for a particular detector – the so-called trigger channel – can be set to keep these kind of less relevant observations out of the study. However, dust, debris, very small particles or stray lights are all among these irrelevant events, and sometimes they can be a problem in the study, if they outnumber the events of interest. Therefore, a threshold is useful to keep these non-essential data away from the important ones.

Gating operations are among the most important to execute in flow cytometry. If a population of cells has homogeneous characteristics, it can be defined as a population of interest: a numerical boundary can be created, so that all the cells inside these gates share similar properties. Gating addresses to some of the main goal of flow cytometry:

- How many and what are the populations of interest in my study? Do they have common characteristics?
- Which biomarkers can be associated with every populations? Are these directly related to healthy/disease states?
- Can I use these populations of cells to confirm the validity of an hypothesis in my study?
- Is there a suggested method for the analysis, more suited for a population with certain characteristics?

Mainly, gating operations are used to choose populations of interest in a study or to get rid of non-interest cells, such as dead cells or debris. These sorts of cells can be recognized mainly

because of lower levels in the Forward Scatter Channel: adopting a threshold can help getting rid of these events, so that they won't interfere in the study.

Backgating is also a very useful application of gating techniques: we can use backgating multiple times in a study, to confirm a gating operation previously made, to confirm multiple populations of cells as the study progresses, or even when we require other information to identify an useful population of interest.

Bioconductor, the FlowCore package and FCS format

Bioconductor is a free open source software project with the aim to produce easier and faster analysis in biology experiment using R, providing tools for the analysis and comprehension of high-throughput genomic data.

Among the Bioconductor packages available, the package FlowCore is crucial for several reasons: to easily represent flow cytometry data, to provide common procedures related to this format of data – such as gating, transformation of the data structure, subsetting data, compensation, creation of spillover matrix… – and also to give an useful method to deal with the problem of dimensionality reduction.

There are two class of objects that can be analyzed using the FlowCore package: the flowFrame and the flowSet, which consists of different flowFrames – this is because sometimes multiple FCS files need to be analyzed together– .

Multiple FCS files in a flowSet can be imported together using the function *read.flowSet*, and a single flowFrame can be called and extracted with the usual extraction operators – for example, *fs[3]* for the third flowFrame in a given flowSet –. However, most of the main procedures inside the FlowCore package are based on flowFrames, whose structure reminds of the FCS file received by the flow cytometer's acquisition software.

An object of class flowFrame is different from a dataframe one, mainly because of his structure: a flowFrame is formed by three main elements, which are his description, his parameters and his expressions. All of these can be called on the console using simple commands, such as *flowFrame_name@description*, providing the possibility to explore every aspect of the data inside the flowFrame.

The description part of the flowFrame consists of an object of class list that contains information, called meta-data, about the flowFrame itself: the FCS version, the path of the FCS file, the Spillover and the Compensation matrix, other information about the file, whether or not we applied a transformation to the data inside the flowFrame… All of these can be called and extract creating a new object into the R environment, for example using the command *flowFr_name@description$*FILENAME. We can extract everything we are interested in simply by changing FILENAME.

The parameters are providing information about the channels located in the FCS file, such as the name, the description, a range of the parameter and a minimum and maximum value after the transformation, all inside a matrix that can also be extracted. We can obtain this matrix with the command *flowFr_name@parameters@data*, and we can even specify one of the variable of interest simply adding *$maxRange* (or any other information needed) after the last command.

The last component of a flowFrame object are the exprs, which are expressed as a numeric matrix formed by all the events or observations – the rows of the matrix – and all of the parameters – the columns of the matrix – collected inside the FCS file. The matrix can be extracted with the command *df ← as.data.frame(flowFr_name@exprs),* creating an object of class dataframe, providing the chance to operate on it like a normal dataframe. Multiple flowFrames can also be merged together, using a channel of interest shared by all of these flowFrames.

METHODS OF DIMENSIONALITY REDUCTION

In recent years, many methods of dimensionality reduction were developed to find new, cheaper and faster ways to solve few problems:

- to figure out easier interpretations of info hidden in the data; to find a way to clean data from a "noise" component formed by useless and sometimes biased info;
- to build ranking system to solve classification problems;
- to define new indicators;
- to create useful and meaningful bidimensional or tridimensional visualization of the data, to better understand the real structures beneath the data;
- to compact information in low-dimensional files, with less storing and processing problems;
- to remove multicollinearity problem from multivariate data;
- just to simplify the data – although, by doing that, usually a part of the initial information is lost –.

Reducing the dimension basically means "pressing" the data together in order to create structures among them. Over time, a lot of methods to reduce dimensionality have been developed, with some of them being very common in flow cytometry: among the most used for flow cytometry data, we considered the UMAP algorithm, which derives directly from Stochastic Neighbor Embedding and in particular from t-SNE algorithms.

t-SNE

Stochastic Neighbor Embedding algorithms are a set of procedures for non-linear dimensionality reduction in two – or three – dimensional output spaces. The main feature of Stochastic Neighbor Embedding methods is the fact that the map that reduces the dimensionality, from k-dimensional data to 2/3 dimensional ones, does not aim to preserve directly the geometric structure of the input points, but it rather retains a set of probability distributions associated with them. Such distributions incorporate information about the space

distribution of the points and therefore, in the final analysis, they try to conserve the geometry, but in an "indirect" way and using approximation criteria.

The main goal of Stochastic Neighbor Embedding algorithms is to build the map $\theta_{sne}(\cdot) : R^k \rightarrow R^p$ $(p=2,3)$ that associates each x_i in the input space to a corresponding $\theta_{sne}(x_i) = \hat{x}_i$ in the output space. With that purpose, for every x_i (with $x_i \neq x_j$) we establish a probability distribution

$$p^*_{j|i} = e^{-\frac{\|x_j - x_i\|^2}{2\sigma_i^2}}$$

with:

- $\|\cdot\|$ as an Euclidean norm in R^k
- σ_i as the standard deviation of the gaussian kernel. A big σ_i may aggregate in the output space points further away in the input space, while smaller σ_i may inaccurately space out points closer in the input space. σ_i depends on a parameter, called perplexity, that helps to define the real density of the data: a greater perplexity creates more well-defined structures among data, while lower-value perplexity is more likely to create looser clusters.

These quantities can be transformed into a probability distribution $P_i = \{p_{j|i}\}_{j \neq i}$ associated to x_i, in which every $p_{j|i}$ is

$$p_{j|i} = \frac{p^*_{j|i}}{\sum_{j \neq i} p^*_{j|i}}$$

and it can be interpreted as the probability that x_j is one of x_i "neighbor".

A particular aspect of these algorithms is the fact that Stochastic Neighbor Embedding chooses to maintain the distances between points in the input space, at the cost of distorting those between distant points. This is because this method isn't able to preserve all the geometry from an input space into a lower-dimensionality space. In a practical way, closer points in the input space will tend to be mapped as closer points in the output space – therefore conserving as much as possible from the original distances –, while distant points in the input space could be mapped in closer or further points than the original ones; this is just to concentrate the distortion into big distances, but this is also one of the main problem of SNE algorithms, leading to output configurations with the majority of points crammed in the center.

One possible solution to this problem is presented by some variations of the normal Stochastic Neighbor Embedding, such as the UNI-SNE and the t-SNE: both these procedures stretch the metric of the output space, leading to low probabilities – further points in the input space – corresponding to far points in the output space, and high probabilities – close points in the input space – corresponding to closer points in the output space. This result can be achieved by raising the output kernel tails, by adding a constant, in the UNI-SNE, equal to an uniform distribution *n(n-1).*

The t-SNE, however, can weight down the kernel tails by using, instead of the gaussian kernel, a Cauchy distribution, that corresponds to a t-student distribution with 1 degree of freedom:

$$p(x) = \frac{1}{\pi} \cdot \frac{1}{1 + x^2}$$

with heavier tails than a normal distribution. This way, further distances from the center are needed to reach lower probability values.

The t-SNE is a better evolution of the Stochastic Neighbor Embedding, but still presents a lot of problems:

- The perplexity parameter is crucial for this dimensionality reduction method, but there isn't a proper way to establish the right value anytime this method is used;
- The algorithm is a little unstable, needs a great number of iterations but, again, there isn't a specific criterion to choose the right number of interactions.
- Sometimes, repeating the same process, with the same parameters and the same data, leads to different results;
- Sometimes, some clusters created by these methods are not very significant;
- Great computation times are usually needed to operate these procedures with a lot of software, like R.

To solve a few of these problems, in the last years a few variants of these methods have been suggested: among these, the UMAP is one of the most suited for FCS files.

UMAP

A new algorithm, called Uniform Manifold Approximation and Projection (UMAP), compared to t-SNE, was created in 2018 by Leland McInnes, John Healy and James Melville to preserve, as much of the local and more of the global data structure, with a shorter runtime.

UMAP and t-SNE are both common methods to reduce dimensionality and therefore create tighter structures built among cell populations with common characteristics. However t-SNE tends to separate into different clusters considerably more than UMAP; in return, UMAP seems to be more consistent than t-SNE in multiple replications, making interpretations easier.

In addition, UMAP methods preserve more of the global structure, and, notably, the continuity of the cell subsets: both algorithms exhibit strong local structures, but UMAP is better at separating groups with similar categories from each other. Plots are way easier to interpret, and this also improves its utility for generating hypotheses related to cellular development. On a practical level, UMAP outputs are faster to compute and more reproducible than those from t-SNE.

The UMAP algorithm is based on strong mathematical principles that ensure the construction of easily-understandable 2D visualizations; UMAP can be associated with a well-known classification method, called K-Nearest Neighbor, or KNN.

Like KNN procedures, UMAP first builds a weighted k-neighbor graph that provides a representation of the k nearest neighbors for every x_i in the input space: the likelihood that two points are connected corresponds to the weighted edge of the graph. The local structure of the graph is preserved by decreasing the likelihood of connection as the distance among points grows.

In the second phase, the UMAP methods produce a two-dimensional representation as an optimized version of the high-dimensionality graph produced in the first phase of the process. A key aspect in this stage of the UMAP process is a deep understanding of 3 parameters crucial in UMAP's algorithm: the number of epochs, the number of neighbors and the minimum distance. They ensure the benefits of UMAP over t-SNE: the result is a method providing more significant visual results, in less time, keeping most information from input data.

The number of epochs, or *n_epochs* parameter, establishes how many times the low-dimensional representation "trains" to optimize and to arrive to the final two-dimensional representation: higher values provide more accurate embeddings. The default value on R is 200 for larger datasets, 500 for smaller ones.

The number of neighbors, or the so-called *n_neighbors* parameter, is the approximate number of points considered nearest neighbors during the construction of the initial high-dimensional graph. Based on the value chosen, we can have UMAP that prioritizes local structures – with lower values – or UMAP that privileges the great structures present in the data while some of the local layout is lost. The default value on R is 15.

The last parameter is the minimum distance, or *min_dist* in R. His value establishes how tightly the UMAP points are aggregated together: lower values are synonym of more clustered points, while higher values tend to produce more isolated points in the output. The default value on R is 0.1.

CLUSTERING METHODS

One of the most important aspect of exploratory data analysis is clustering: not only grouping different observations can help to discover information that may pass undetected at a first glance, but it can also help to identify anomalies among great amount of data, or even to provide classification methods, with a lot of possible applications in many different fields, from social network analysis, to market research and customer segmentation, to document classification and network traffic classifications.

Some of the most common procedures, like K-means or SOM, can also be applied to flow cytometry data, in order to obtain more information during biomedical studies with research purposes.

K-means

K-means is one of the most widely used and best performing clustering algorithm. Furthermore, it is a very simple algorithm to implement and use. It is based on the so-called centroids, points in the feature space that averages the distances between all the data in the associated cluster. They therefore represent a kind of barycentre of the clusters and in general, due to their characteristics, they may or may not be some of the points of the input dataset.

The K-means algorithm proceeds this way:

1. Not knowing the classes actually in the input dataset, the first thing to do is to decide the number of classes (or better clusters, in this case) in which we want to divide the dataset itself. This number is called K, from which the name of the method K-means (the term “means” implies the use of centroids, i.e. average points).

2. K centroids belonging to the feature space are chosen randomly. The only condition is that they need to be not coincident, or even better they need to be usually far enough apart. Otherwise the algorithm may have problems converging.

3. The distance between each point in the dataset and each centroid is calculated.

4. Each point in the dataset is associated with the cluster connected to the nearest centroid.

5. The position of each centroid is recalculated by averaging the positions of all points in the associated cluster.

6. Iterate from step 3 until there is no more input changing clusters.

The K-means algorithm is very suitable for scenarios where you can create groups of similar objects from a collection of randomly distributed objects. Generally, it is suitable in cases where you know the number of clusters a priori and you can obtain distinct groups from the dataset (clusters).

The K-means algorithm has the advantage of being quite fast, as few calculations and consequently little computer processing time are required to calculate the distances between data points and centroids at each iteration (obviously this depends on the dataset and the number of clusters).

On the other hand, K-means have a couple of disadvantages: first of all, you have to select how many clusters you want to display; this is sometimes trivial as it is not always possible to do so, especially for higher complexity problems. In addition, K-means also start with a random choice of centroids and therefore may produce different clustering results depending on the different sequences of the algorithm. Therefore, the results may not be repeatable and may lack consistency.

SOM

SOM – or Self-Organizing Maps – are a non-linear method to reduce dimensionality, developed to create bi-dimensional visualization of multi-dimensional probability distribution. The n statistical units, represented by vectors in R_k, are approximated by m (typically $m < n$) k-dimensional vectors, called “codebooks”, which are associated with the cells of a two-dimensional rectangular grid and then reproduced in a graphic visualization. Codebooks play a role similar to the centroids in k-averages, which is to quantize the input density; however, unlike k-averages, codebooks are "intelligently" arranged in the plane, so that regular patterns emerge, summarizing the multidimensional density.

Technically, the SOMs are a particular type of neural network, structured through a process of competitive learning that realizes a “tessellation of Voronoi” of the input space and maps the Voronoi cells on the cells of a bi-dimensional grid, substantially constructing a bi-dimensional histogram, in the logic of non-parametric procedures to estimate a density.

The algorithm for generating a SOM consists of a sequence of learning cycles, called epochs, in which the map "learns" from the data and self-organizes, generating the codebooks associated with the grid cells. To make the algorithm operational, the analyst must specify some basic characteristics and some parameters of the map, for example its size, the structure of each cell – usually squared or hexagonal – and the topology of the map – toroidal, which means the border cells from one side are considered close to the ones on the opposite side; or planar, where cells from opposite borders are considered far apart –.

In every epoch, the vectors of the input dataset are "presented" to the map, one at a time and in random order, and compared with the codebooks available at the beginning of the epoch. For each vector presented and for each codebook, the Euclidean distance is calculated so that one codebook minimizes it:

$$d(\boldsymbol{v} - \boldsymbol{c}_i) = \|\boldsymbol{v} - \boldsymbol{c}_i\|_2$$

That codebook is the so-called "winner codebook". After that, the winner codebook is made more similar to the input vector with a transformation, in a way that, as the epochs proceeds, it decreases, so that the map can become more stable epoch after epoch.

$$\boldsymbol{c}_{i*}^{new} = c_{i*} + h(s, i^*|i^*)[\boldsymbol{v} - \boldsymbol{c}_i]$$

where $h(s, i|i^*)$ is the so-called neighborhood function, that determines how the information is spread across the map, while adapting to the codebooks already inside the map.

$$h(s, i|i^*) = \alpha(s) e^{-\frac{\|cell_i - cell_{i^*}\|^2}{2\sigma(s)}}$$

where:

- $cell_{i^*}$ is the vector that indicates the center of the cell that contains the winning codebook
- $cell_i$ is the vector that indicates the center of the cell that contains the codebook i
- $\alpha(.)$ and $\sigma(t)$ are two decreasing functions

The two functions $\alpha(.)$ and $\sigma(t)$ are two monotone decreasing functions that determine how much the strength of the changing in the map and the radius of diffusion of these changes decrease as the epochs proceed. This is essential for these methods, otherwise new epochs would just erase every progress made in the previous ones.

After these, every other codebook is modified starting from the winner codebook, based on the proximity to this one. When all the input vectors are presented to the map, the epoch is over; however, the process goes on and the codebooks are again modified in the in-between of every new epoch. The succession of epochs continues until the map is stable, which means that the codebooks do not change (or change just a little bit) with new epochs in the process.

Density Based Clustering

DBSCAN, or Density-Based Spatial Clustering of Applications with Noise, is a fast clustering method suited for datasets containing noise and great amounts of outliers. Introduced on 1996 by Martin Ester, Hans-Peter Kriegel, Jörg Sander and Xiaowei Xu, it is an evolution of K-means procedures with a focus on density-based clustering.

Unlike the K-means approach, DBSCAN doesn't require to specify, before the use, the accurate number of clusters that best reproduce the data distribution: instead, this number is established by the algorithm itself, based on different parameters chosen by the user; other differences from K-means are the shapes of clusters – with DBSCAN applications we wouldn't just have circular clusters, but they can arrange in any possible shape – and also DBSCAN is useful to identify outliers inside the data and to produce visual output following the clustering procedures executed with it.

Two of the most important parameters that need to be chosen are:

- minPts: the minimum number of points, from the input data, that we need to define a cluster: we increase this parameter to reduce the noise presented in the data and to have bigger and more significant clusters; the default value for R is 5 points.
- ε: the radius of the neighborhoods region surrounding each observations; the best value for ε can be found by inspecting the *kNNdistplot*, with a pre-chosen value of minPts in which we are interested in, to find a knee in the plot that indicates a threshold: on his

left there are normal observations, while the part on the right of that line indicates the possible outliers.

Every point in the data which has more neighbor points than the minPts chosen, inside the radius picked, is called core point; for every core point not already part of a cluster, a new cluster is created. Every point with less neighbor points inside the radius picked than the established minPts is called border point. Every other point which is neither core nor border point is a noise point or an outlier. The process is over when every point in the dataset has been inspected and established as core, border or outlier.

DATA DESCRIPTION

The dataset used in our work was based on a study about venous blood, collected prior to surgery, in patients with diagnosed colo-rectal cancer (status T2-T4, any status N and microsatellite stability), with indications for surgical removal with curative intent.

Data structure: explorative report

The flowFrame *exprs* was made of 32 channels: FSC-H, FSC-A, SSC-H, SSC-A, CD45 FITC FITC-H, CD45 FITC FITC-A, PerCP-H, PerCP-A, HLA-DR APC-H, HLA-DR APC-A, CD10 APC-A700-H, CD10 APC-A700-A, CD14 APC-A750-H, CD14 APC-A750-A, CD16 PB450-H, CD16 PB450-A, CD15 KO525-H, CD15 KO525-A, CD8 Violet610-H, CD8 Violet610-A, Violet660-H, Violet660-A, LOX1 PE-H, LOX1 PE-A, PDL1 PECF ECD-H, PDL1 PECF ECD-A, CD3 PC5.5-H, CD3 PC5.5-A, PD1 PC7-H, PD1 PC7-A, FSC-Width, TIME.

Data were compensated using both the *estimateLogicle(fcs, channels = nomi_ch, m = 5.1)* procedure, both with the *as.data.frame(transform(fcs, lgc)@exprs)*. The logicle transformation, which is the inverse of a modified biexponential function, has the following parameters:

- $w = \frac{(m - log_{10}(\frac{t}{|r|}))}{2}$
- m : the full width of the transformed display in asymptotic decades.
- t : the top of the scale data value, with greater value than 0.
- r : the most negative value to be included in the display

In particular, during the analysis stages, we focused on 13 channels in particular, representing channels of interest, that were previously scaled: CD45 FITC FITC-A, PerCP-A, HLA-DR APC-A, CD10 APC-A700-A, CD14 APC-A750-A, CD16 PB450-A, CD15 KO525-A, CD8 Violet610-A, Violet660-A, LOX1 PE-A, PDL1 PECF ECD-A, CD3 PC5.5-A, PD1 PC7-A. All of them were numerical variable, and they were distributed as shown in the next table:

	Min	Q1	Median	Mean	Q3	Max	sd
CD45 FITC FITC-A	1.666	1.977	2.223	2.253	2.468	3.227	0.329
PerCP-A	-1.041	0.882	1.198	1.144	1.372	4.138	0.316
HLA-DR APC-A	-0.312	0.555	0.795	0.985	1.269	3.331	0.625
CD10 APC-A700-A	-0.559	0.420	0.504	0.503	0.594	4.810	0.166
CD14 APC-A750-A	-0.224	0.452	0.689	0.961	1.014	3.347	0.820
CD16 PB450-A	-0.193	1.345	1.715	1.924	2.801	3.251	0.862
CD15 KO525-A	-1.325	0.492	1.506	1.213	1.781	2.695	0.666
CD8 Violet610-A	-0.416	0.534	0.687	0.721	0.905	2.111	0.265
Violet660-A	-0.325	0.249	0.352	0.355	0.457	1.987	0.164
LOX1 PE-A	-1.140	0.499	0.636	0.661	0.795	3.105	0.264
PDL1 PECF ECD-A	-0.461	0.328	0.801	0.849	1.336	2.691	0.563
CD3 PC5.5-A	-0.712	1.582	1.860	1.710	2.067	3.622	0.701
PD1 PC7-A	-2.459	0.653	0.884	0.893	1.130	2.923	0.335

These were the boxplot associated with every channel considered:

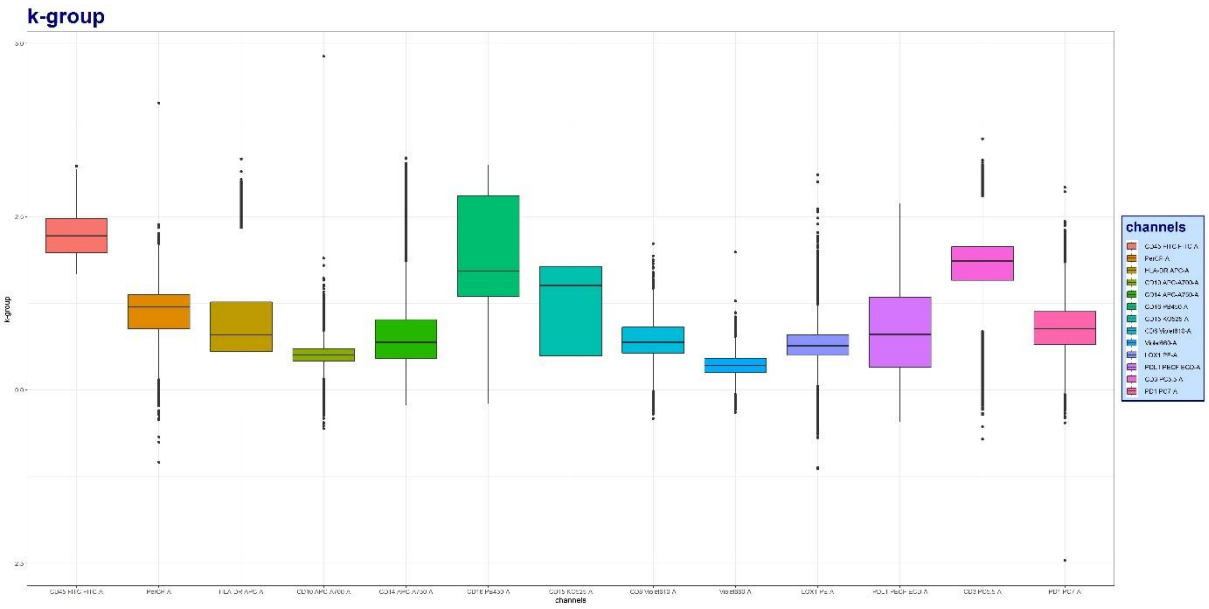


Figure 1: boxplot full dataset compensated and transformed with logicle transformation

After the first analysis on our dataset of interest, we were interested in figuring out the possible number of different metaclusters we could find applying the clustering methods suited for this work: so, we decided to visualize both a t-SNE plot and an UMAP plot, without any metacluster predetermined, to explore the possible outcome. We needed to use a sample for the t-SNE because the process was very burdensome for our devices.

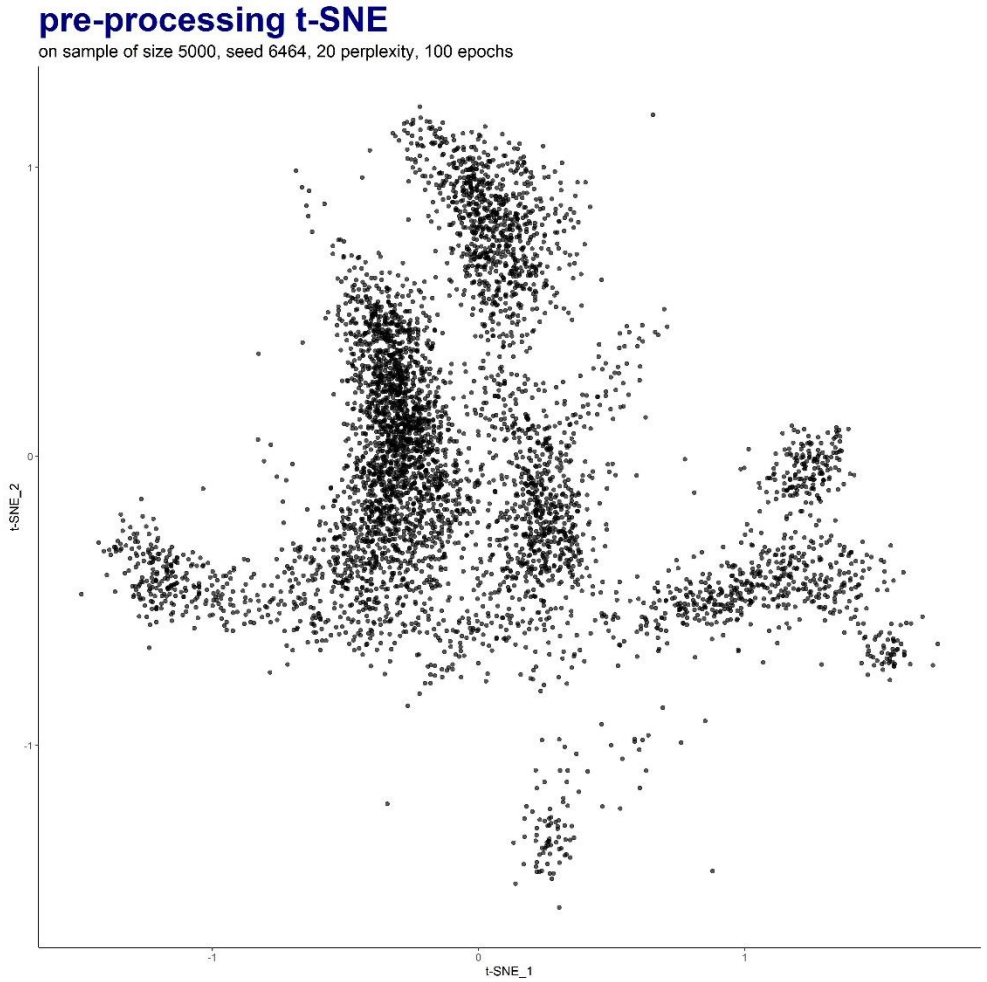


Figure 2: pre-processing t-SNE plot

We then executed the UMAP plot: we chose to use the second method, and not t-SNE, for the whole analysis, because of the greater generalizability of the procedure, the shorter computational times and the clearer division into four, well-separated groups inside the plot.

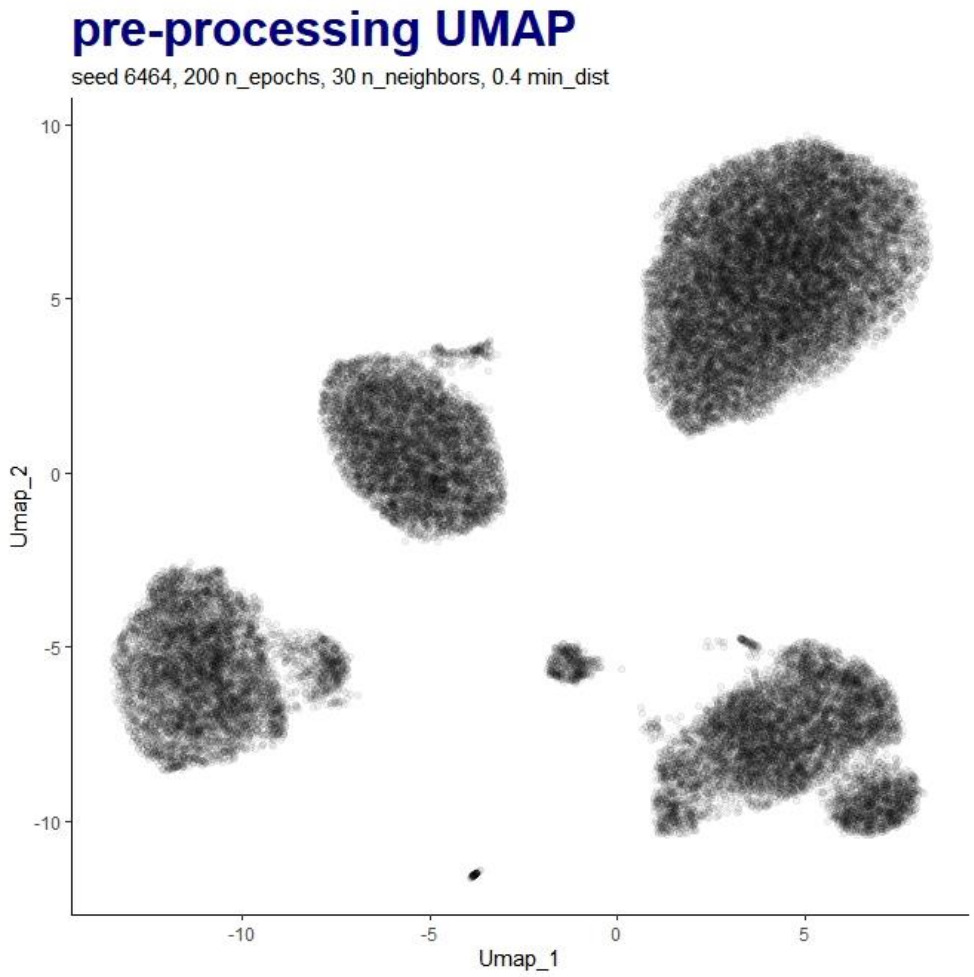


Figure 3: pre-processing UMAP plot

Looking at the pre-processing UMAP, we realized that the data collected seemed to create 4 general clusters. The other, smaller clumps may be related to few observations, left inside the dataset from the process of data preparation, as debris or stray lights.

DIFFERENT CLUSTERING METHODS APPLIED

We used standardized data for the following clustering methods, here I reported the summary and the boxplots related to the original data transformed with logicle, and then standardized:

	Min	Q1	Median	Mean	Q3	Max	sd
CD45 FITC FITC-A	-1.785	-0.840	-0.092	0	0.652	2.959	1
PerCP-A	-6.916	-0.830	0.172	0	0.720	9.474	1
HLA-DR APC-A	-2.076	-0.688	-0.303	0	0.455	3.755	1
CD10 APC-A700-A	-6.379	-0.500	0.009	0	0.549	25.878	1
CD14 APC-A750-A	-1.446	-0.622	-0.332	0	0.064	2.910	1
CD16 PB450-A	-2.454	-0.671	-0.242	0	1.017	1.539	1
CD15 KO525-A	-3.814	-1.083	0.439	0	0.854	2.226	1
CD8 Violet610-A	-4.292	-0.704	-0.128	0	0.695	5.249	1
Violet660-A	-4.141	-0.645	-0.016	0	0.622	9.941	1
LOX1 PE-A	-6.816	-0.612	-0.093	0	0.510	9.253	1
PDL1 PECF ECD-A	-2.328	-0.925	-0.084	0	0.866	3.275	1
CD3 PC5.5-A	-3.454	-0.183	0.214	0	0.509	2.727	1
PD1 PC7-A	-10.019	-0.717	-0.027	0	0.710	6.069	1

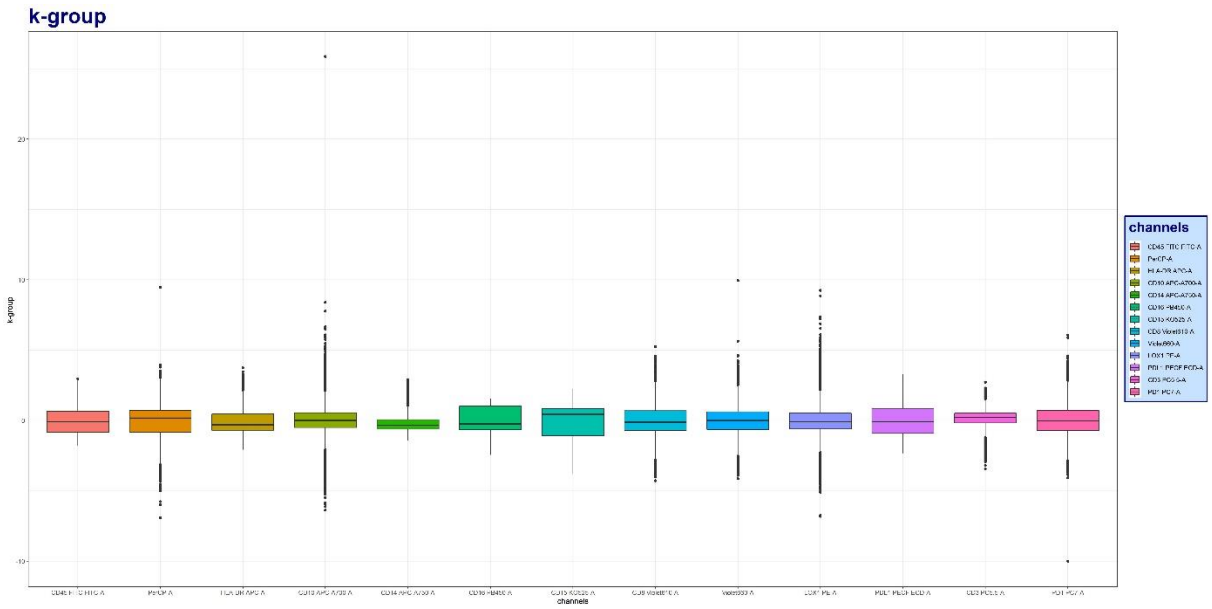


Figure 4: boxplot full dataset compensated and transformed with logicle transformation and standardized

K-means method

We applied the K-means method to the dataframe with transformed and scaled channels of interest. We decide, after observing the pre-processing UMAP, to use the K-means method with the *set.seed* at 6464, 4 centroids considered and 100 of *nstart* – which is the initial number of configurations considered as the first one to start the process –. We obtained 4 k-groups distributed like this, and this is the following UMAP plot after the K-means method applied:

k-group	1 (red)	2 (blue)	3 (green)	4 (yellow)
n	16958	7275	6366	8086

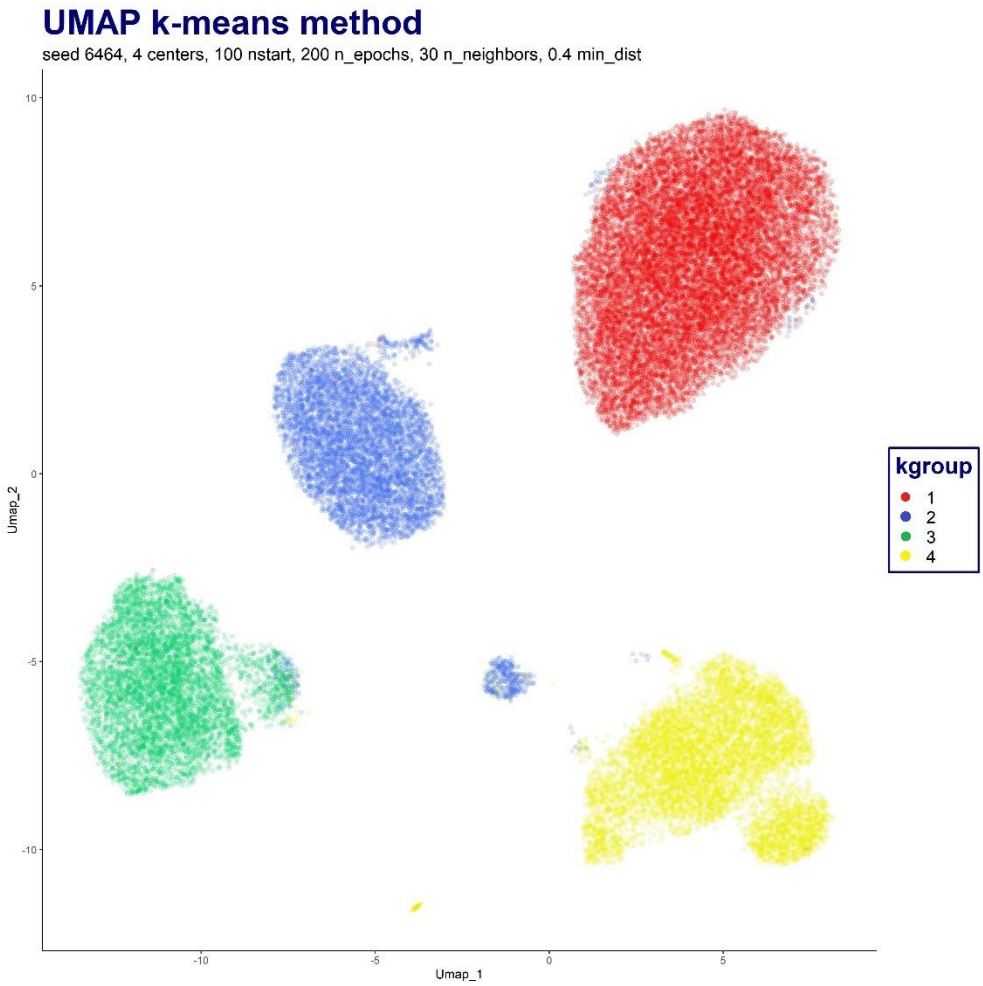


Figure 5: UMAP plot with K-means method

We can say that the clustering method worked very well. The 4 k-groups seems to be well-represented in the UMAP, with just few observations far from the big clumps with the majority of the observations: they could also be some left dust, debris or stray lights. The characteristics of each of the four, well-separated groups can be considered for further analysis, due to the results obtained: all well-separated group with just few observations linked to different metaclusters than the one of the nearby clump in the UMAP plot – mostly in the bottom-left cluster, with some observations from group 2 and group 4, and in the top-right cluster, with some observations belonging to the second group –.

flowSOM method

We applied functions belonging to the flowSOM package, produced by Bioconductor to create visualizations output utilizing Self-Organizing Map clustering methods; the procedures were used on the FlowFrame with transformed channels, considering only the channels of interest. We decide, after observing the pre-processing UMAP, to use the flowSOM method with the *set.seed* at 6464, and a number of clusters pre-chosen at the beginning of the process at 4. We obtained 4 metaclusters distributed like this:

metacluster	**1 (red)**	**2 (blue)**	**3 (green)**	**4 (yellow)**
n	23791	395	6442	8057

We noticed that, while group 3 and 4 were basically the same as the k-groups in the K-means method (6442 instead of 6366 for the third group, 8057 instead of 8086 for the forth), group 1 and 2 changed drastically: a significant part of group 2 passed to the first metacluster with the flowSOM method, significantly reducing group 2 to 395 instead of the previous 7257. This is the result obtained with the UMAP:

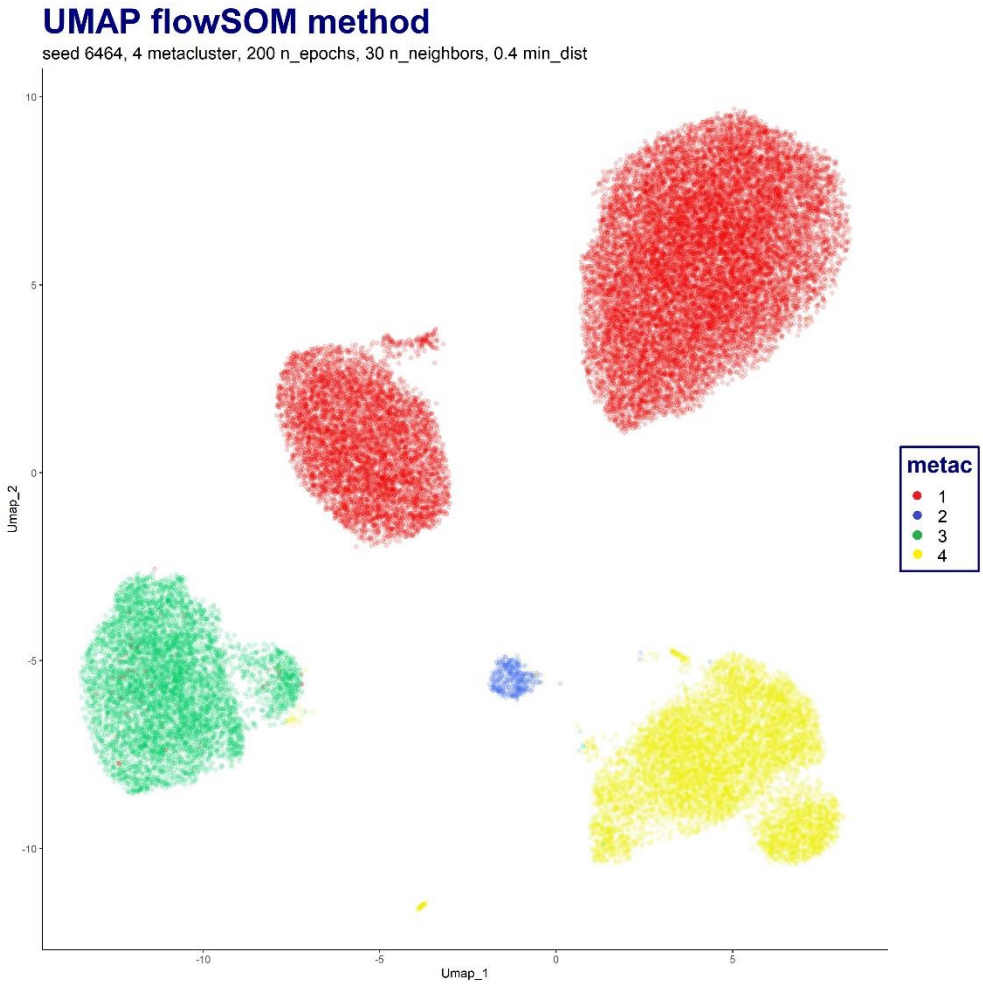


Figure 6: UMAP plot with flowSOM method (4 metaclusters)

Group 2, which in the K-means UMAP was the clump in the top left corner, now is part of group 1; the actual group 2 is just a small amount of observations, already part of group 2 in the K-means process, in the center of the UMAP plot, closer to group 4.

At this stage, given the actual visualization of the UMAP plot with 4 metaclusters in the flowSOM method, to explore a little more our data we decided to apply the same method but with just three metaclusters, to see if something would change about the numerosity and composition of the metaclusters. We decided to use the flowSOM method with the *set.seed* at 6464, and a number of cluster pre-chosen at 3. We obtained 3 metaclusters distributed like this:

metacluster	**1 (red)**	**2 (green)**	**3 (yellow)**
n	23791	6442	8452

The group 2 in the previous flowSOM was included in the previous group 4 (the yellow one, now called group 3), while the other two groups stayed exactly the same. The UMAP obtained is the following one:

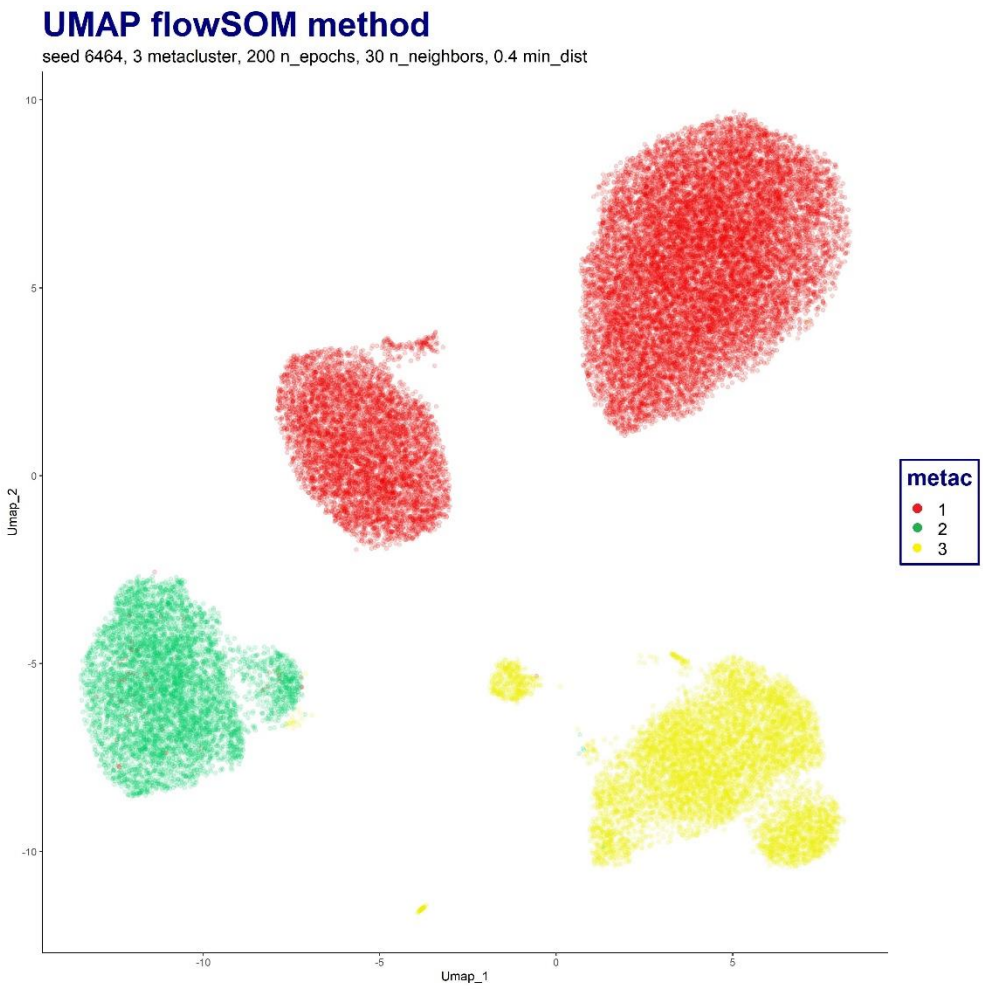


Figure 7: UMAP plot with flowSOM method (3 metaclusters)

This is the UMAP we could expect from the previous one: group 2 was included in the closest one in the last UMAP plot – the previous group 4 –, meaning that the observations of that smaller group were similar to the ones in group 4; the rest of the groups were actually the same.

The clusters obtained with this clustering method were good, with well-separated clumps; however, we need to consider the fact that the composition of the different cluster changed a lot from K-means methods to flowSOM methods. The first group in flowSOM methods was the merged composition of the first two groups in K-means.

Kohonen SOM

The third method considered after k-means and flowSOM is the Kohonen SOM: it's a method to project an input dataset into a two-dimensional grid, in a way that each grid node is associated with a single *unit_classifier*. We applied this method to the dataframe with transformed and scaled channels of interest, with *set.seed* of 6464. We built a squared grid 5x5 consisting of 25 *unit_classifier* to associate with 4 metaclusters:

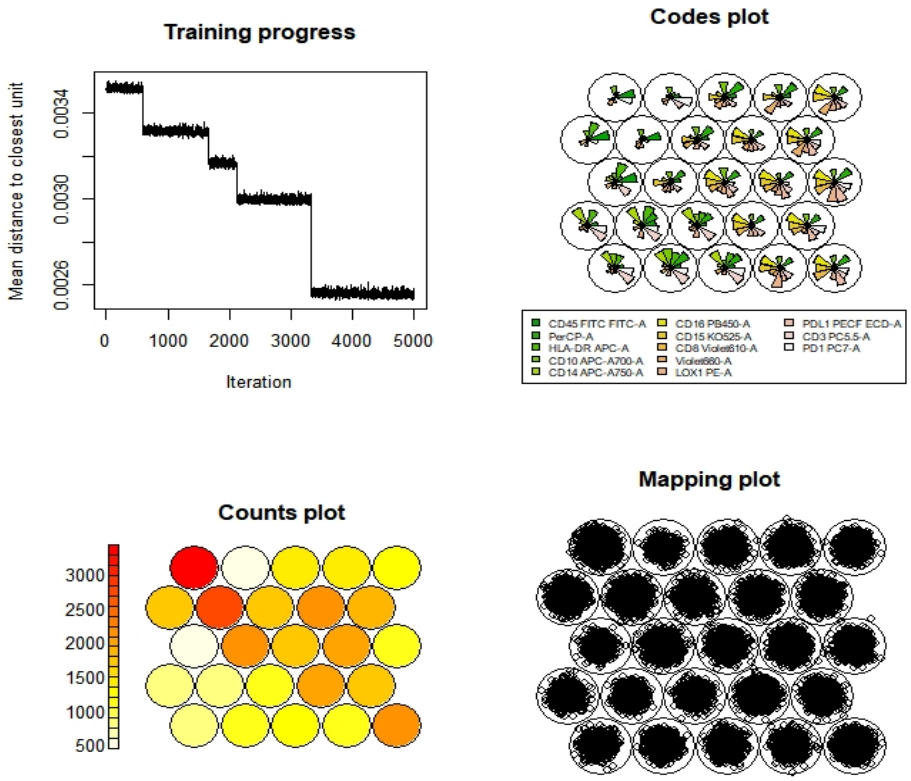


Figure 8: diagnostic plots for somGrid

The four plots reported above represent some diagnostic plots related to the grid created:

- the first plot represents the mean distance to the closest unit as the number of interactions proceeds;
- the second plot represents all the pie charts to define the magnitude of each channel inside each grid node;
- the third plot is an heatmap showing how many observations were chosen for each grid node, with red groups as the bigger and yellow one as the smaller groups

- the forth plot represents how close any point is to the representative vector of that group.

This is the division in metaclusters obtained from the dendrogram, calculating Manhattan distances for all the observations. We tried to preserve a division in 4 metaclusters, the same number of groups we used in K-means method.

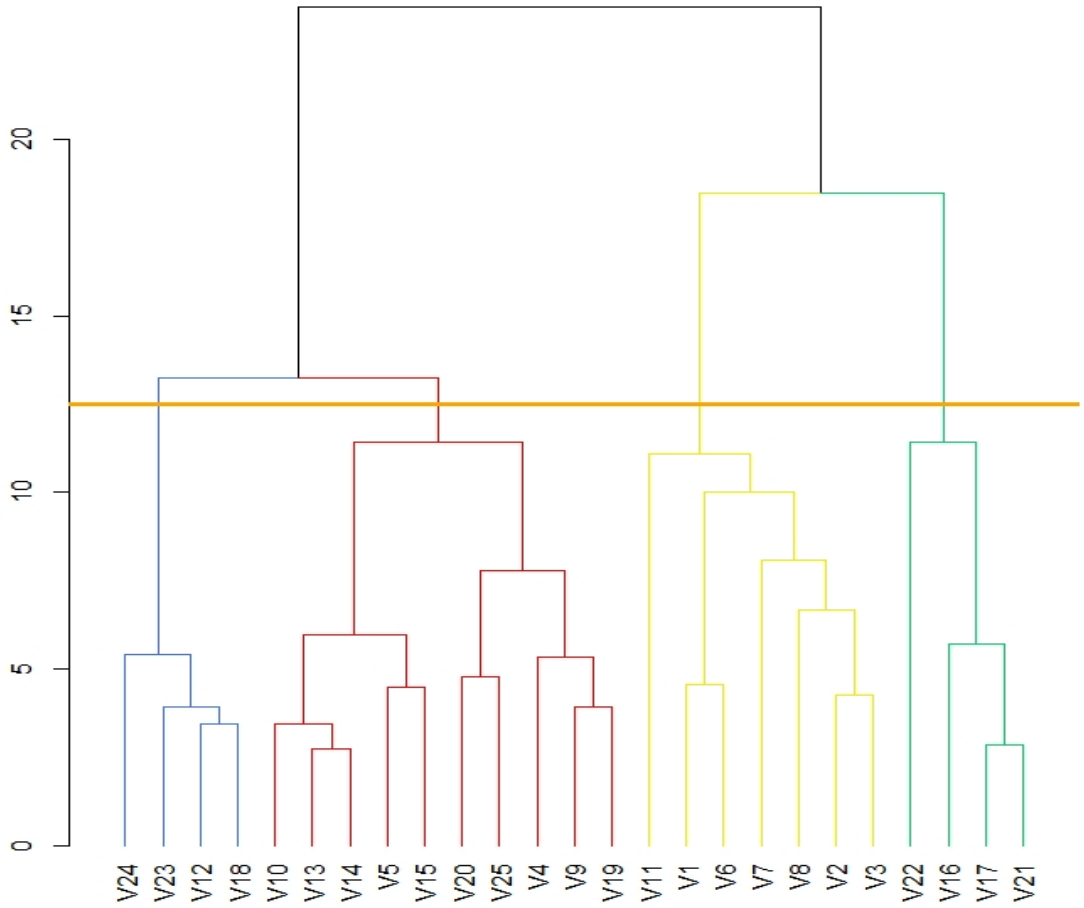


Figure 9: dendrogram somGrid

As shown in the plot, by cutting the dendrogram at 4 metaclusters, we obtained four groups composed like this:

- group 1: units 4, 5, 9, 10, 13, 14, 15, 19, 20, 25
- group 2: units 12, 18, 23, 24
- group 3: units 16, 17, 21, 22
- group 4: units 1, 2, 3, 6, 7, 8, 11

This is the composition of each group obtained with the Kohonen SOM:

metacluster	1 (red)	2 (blue)	3 (green)	4 (yellow)
n	15491	6300	4828	12066

The four metaclusters were very different from every other division found until that moment, so we could expect an UMAP plot very different from all the previous ones.

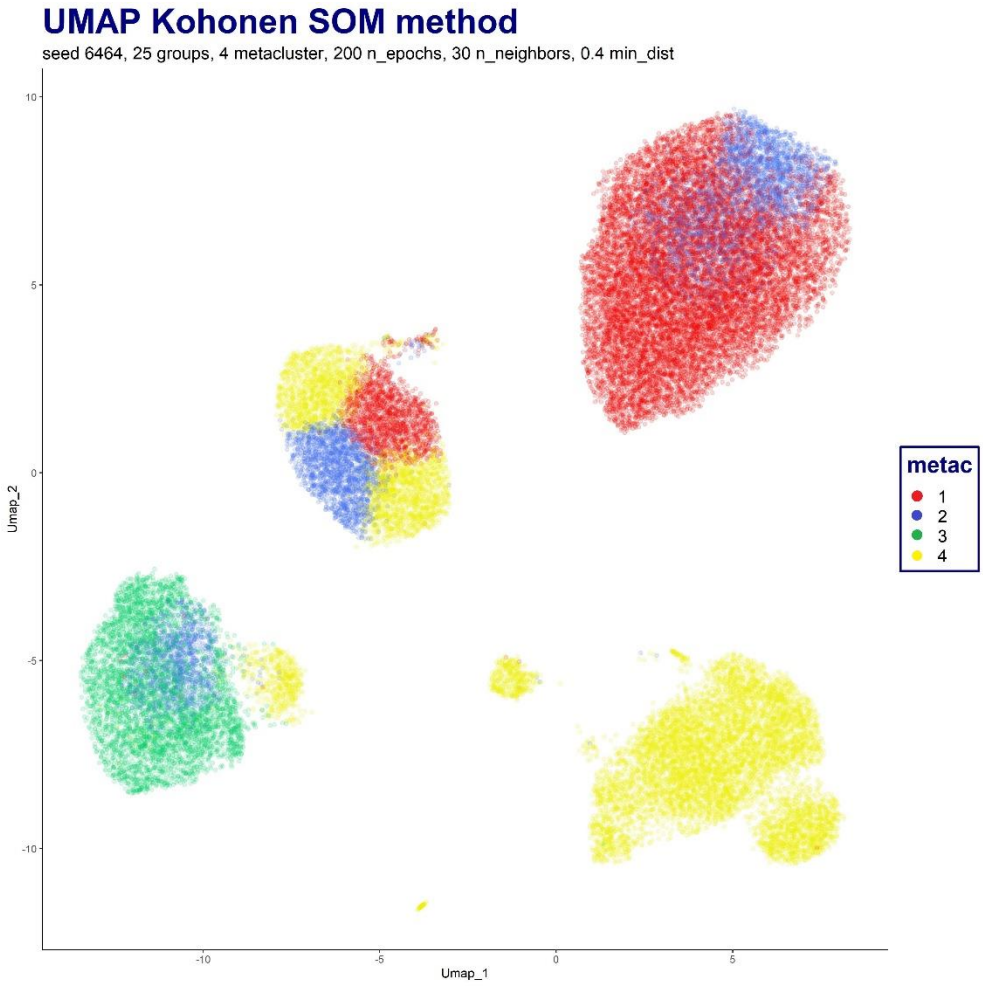


Figure 10: UMAP plot with somGrid method

Unfortunately the four groups weren't well-separated: comparing to the division achieved with the K-means method, the first group (which was just the top-right clump, the red one) spread to the previous second group, which became a mix of first (red), second (blue) and forth (yellow) group; the third group (the previous green one, at the bottom-left in K-means UMAP) had now a significant part of observations considered as part of second (blue) and forth (yellow)

groups; the fourth group (the yellow one) was well represented on the bottom-right of the UMAP plot, like in previous UMAPs, but spread all over the previous group 2 and 3.

Density Based Clustering method

Finally, the last clustering method used was the DBSCAN method: we applied the DBSCAN method, with *set.seed* of 6464, ε = 1.5 and k=180 as the MinPoints parameter in the k-Nearest Neighbor method applied. This way, we obtained 4 metaclusters composed like this:

metacluster	0 (red)	1 (blue)	2 (green)	3 (yellow)
n	22073	754	8474	7384

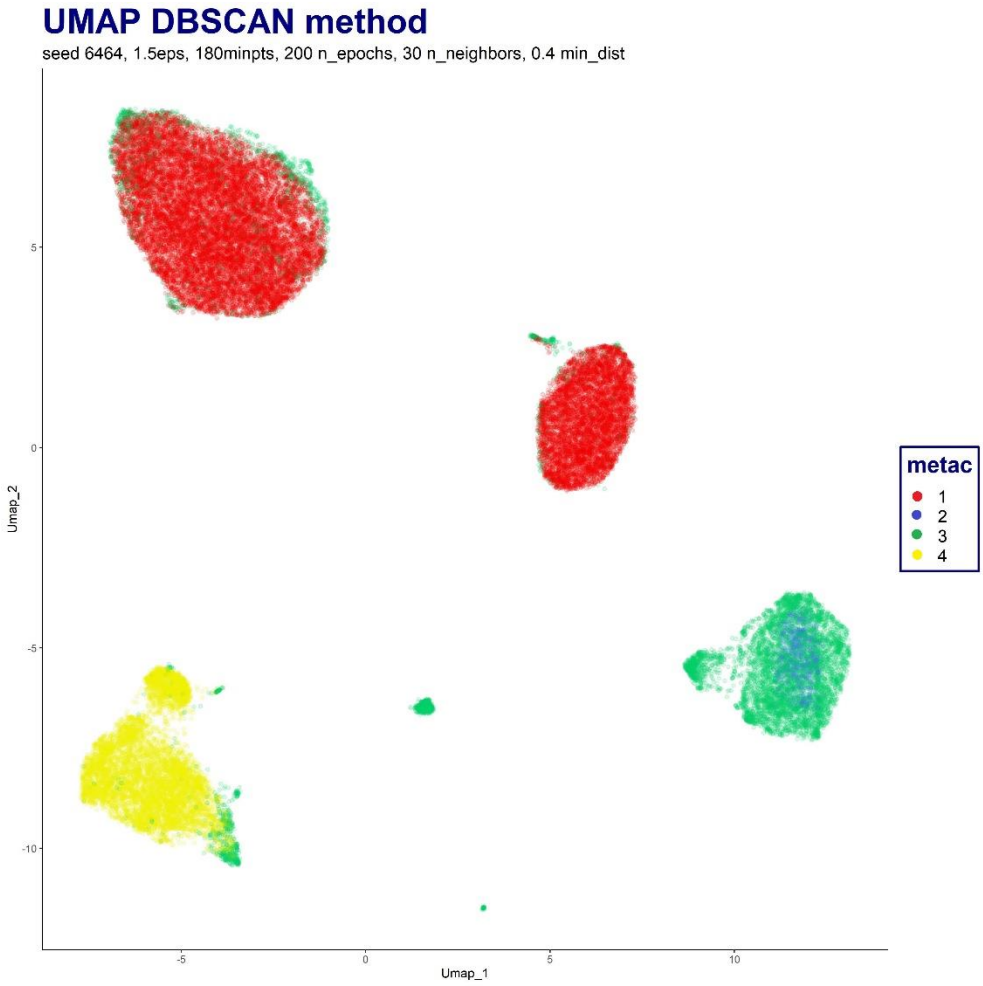


Figure 11: UMAP plot with DBSCAN method

We noticed that the four groups weren't divided like before, considering other clustering methods, so we could expect an UMAP plot different from those.

Like in the UMAP plots obtained with flowSOM methods, the two clumps on top-left and top-right were put in the same group (although there were way more observations from other clusters, in particular from the third metacluster, near the two clumps); the green and blue group were mixed in the bottom-right clump, with prevalence of observations from the green one. Finally, the clump on bottom-left of the UMAP plot had a prevalence of forth group, with some observations from group three (the green one) too.

DATA ANALYSIS FOR K-MEANS CLUSTERING

We decided to consider the K-means as the clustering method more aimed at the purpose of our work: with 4 clusters, all with considerable dimensions, all well-separated from each other. Then, we decided to analyze the different clusters obtained, considering one k-group per time, to find possible differences comparing the four metaclusters both with themselves and with the complete original data.

This was the division into the 4 k-groups:

k-group	1 (red)	2 (blue)	3 (green)	4 (yellow)
n	16958	7275	6366	8086

Group composition

The first group, composed by 16958 observations, produced these statistics and these boxplots:

	Min	Q1	Median	Mean	Q3	Max	sd
CD45 FITC FITC-A	-1.785	-1.125	-0.877	-0.885	-0.642	2.769	0.358
PerCP-A	-3.011	-0.079	0.289	0.289	0.655	9.474	0.559
HLA-DR APC-A	-2.032	-0.834	-0.567	-0.586	-0.313	3.755	0.381
CD10 APC-A700-A	-2.492	-0.429	0.039	0.066	0.532	25.878	0.745
CD14 APC-A750-A	-1.391	-0.456	-0.223	-0.227	-0.002	2.305	0.336
CD16 PB450-A	-0.166	0.957	1.036	1.029	1.110	1.539	0.123
CD15 KO525-A	-1.539	0.726	0.888	0.868	1.037	2.226	0.251
CD8 Violet610-A	-4.292	-0.104	0.705	0.546	1.336	5.249	1.090
Violet660-A	-3.905	-0.664	0.082	0.103	0.846	9.941	1.100
LOX1 PE-A	-2.842	-0.062	0.425	0.563	1.038	9.253	0.925
PDL1 PECF ECD-A	-2.145	0.547	0.974	0.925	1.361	2.889	0.638
CD3 PC5.5-A	-2.539	0.118	0.279	0.268	0.434	2.727	0.251
PD1 PC7-A	-10.019	-0.643	-0.063	-0.058	0.527	3.793	0.863

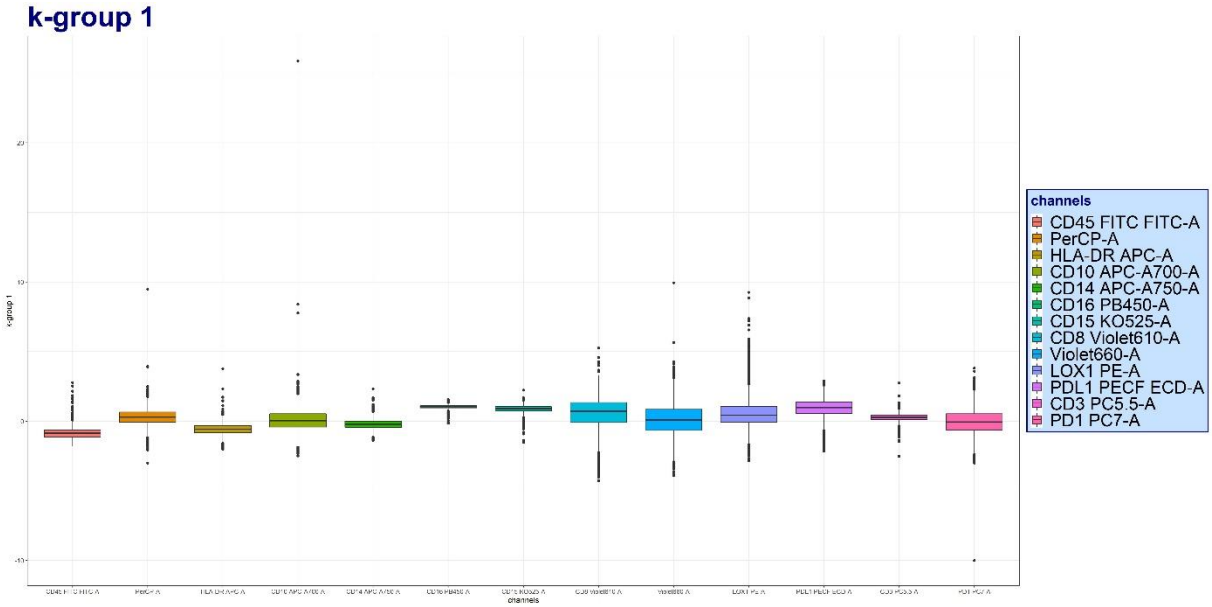


Figure 12: Boxplots k-group 1, for every channel

Compared to the complete dataset, looking at standard deviation's values, we noticed that a lot of the channels had very low value of standard deviation (in particular CD16 PB450-A with 0.123, CD3 PC5.5-A with 0.251 and CD15 KO525-A with 0.251), meaning that the data were clustered around the mean for these channels. However, other channels had significant higher value for standard deviation, with Violet660-A as the most spread out distribution with a 1.1 standard deviation.

Looking specifically at some distributions of single channels, we noticed some differences between the complete data and the observations included in the first K-means group: for example, now the first channel (CD45 FITC FITC-A) included more observations with lower value (the third quartile went from 0.652 to -0.642); the CD16 PB450-A changed a lot too, going from a distribution with half the value lower than 0 (with minimum value of -2.454 and the first quartile at -0.671), to a minimum value of -0.166 and the first quartile at 0.957; the channel CD15 KO525-A was almost the same way as the CD16 PB450-A, with a lowest value of -1.539 compared to the -3.814 from the complete summary, and with first quartile at 0.726 instead of the -1.083 for the complete data.

In general, the first group included a lot of the highest values among all the channels: in particular, the top value for all the channels except for CD45 FITC FITC-A, for CD14 APC-A750-A, for PDL1 PECF ECD-A and for PD1 PC7-A; we even realized that one of the biggest outlier for the entire dataset, the value 25.878 for channel CD10 APC-A700-A, was part of this first group.

Speaking of outliers, the vast majority of them, for the first group, were related to the channels LOX1 PE-A, Violet660-A and CD8 Violet610-A.

The second group, composed by 7275 observations, produced these statistics and these boxplots

	Min	Q1	Median	Mean	Q3	Max	sd
CD45 FITC FITC-A	-1.783	0.157	0.328	0.295	0.494	2.354	0.427
PerCP-A	-4.267	0.363	0.726	0.646	1.062	3.528	0.671
HLA-DR APC-A	-2.046	-0.621	-0.300	-0.266	0.023	2.654	0.549
CD10 APC-A700-A	-2.599	-0.144	0.343	0.361	0.845	6.653	0.775
CD14 APC-A750-A	-1.365	-0.703	-0.522	-0.526	-0.341	1.341	0.266
CD16 PB450-A	-2.125	-0.549	-0.453	-0.480	-0.364	1.116	0.244
CD15 KO525-A	-2.464	0.246	0.409	0.315	0.564	1.323	0.481
CD8 Violet610-A	-3.675	-0.805	-0.236	-0.255	0.303	4.372	0.841
Violet660-A	-2.923	-0.295	0.314	0.345	0.953	4.633	0.925
LOX1 PE-A	-4.338	-0.459	-0.149	-0.188	0.147	5.342	0.650
PDL1 PECF ECD-A	-2.328	-0.300	-0.109	-0.160	0.061	0.920	0.342
CD3 PC5.5-A	-1.564	-0.056	0.122	0.127	0.296	2.055	0.336
PD1 PC7-A	-3.134	-1.297	-0.844	-0.734	-0.317	3.574	0.840

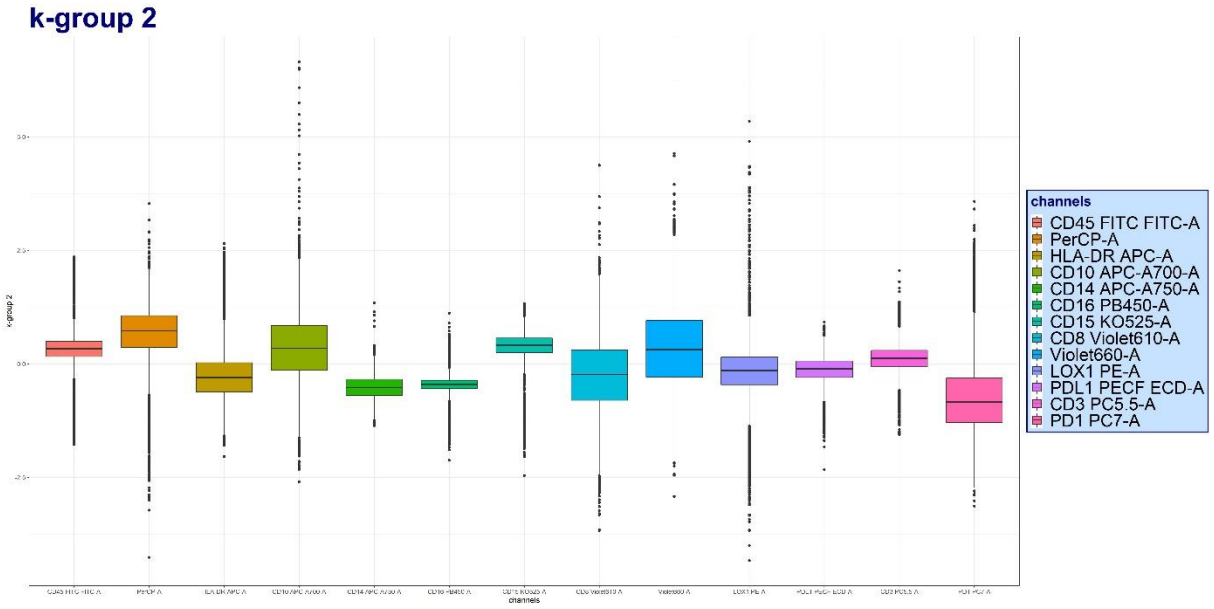


Figure 13: Boxplots k-group 2, for every channel

In general, the summaries of the second group, as opposed to the first group, had more similar distributions to the summaries of the complete dataset; however, a few differences could be found: some channels had a lot more observations with lower values (for CD45 FITC FITC-A, PerCP-A and CD15 KO525-A, first quartile passed respectively from -0.840 to 0.157, from -0.830 to 0.363 and from -1.083 to 0.246), while others had more higher values (for CD16 PB450-A and PD1 PC7-A, third quartile went from 1.017 to -0.364, and from 0.710 to -0.317).

The standard deviation in general was lower than 1, with a couple of values close to the unit, but in general it was a little lower than the first group. Also, all the values of mean were very restrained, all between -0.734 and 0.646, with even 10 channels between -0.5 and 0.5 values.

Compared to the first group, we noticed that the second group included a lot more outliers, in particular the boxplot showed a lot of outliers for channels CD10 APC-A700-A and LOX1 PE-A. However, despite having a lot of outliers, most of the highest values were still inside group 1, as mentioned before, and not in this one. Nevertheless, one of the lowest values, for channel PDL1 PECF ECD-A (-1.949), was inside the second k-group.

The third group, composed by 6366 observations, produced these statistics and these boxplots:

	Min	Q1	Median	Mean	Q3	Max	sd
CD45 FITC FITC-A	-1.782	-0.135	0.169	0.294	0.572	2.713	0.719
PerCP-A	-6.916	-0.820	0.260	0.073	1.103	3.785	1.392
HLA-DR APC-A	-1.781	1.058	1.507	1.514	2.011	3.464	0.746
CD10 APC-A700-A	-6.379	-1.540	-0.359	-0.341	0.820	6.013	1.718
CD14 APC-A750-A	-0.779	2.065	2.202	2.076	2.322	2.910	0.505
CD16 PB450-A	-1.997	-0.730	-0.429	-0.491	-0.200	1.141	0.405
CD15 KO525-A	-2.590	-1.125	-0.914	-0.877	-0.667	1.297	0.380
CD8 Violet610-A	-2.389	-0.672	-0.313	-0.319	0.028	2.349	0.528
Violet660-A	-4.141	-1.082	-0.542	-0.581	0.001	3.132	0.852
LOX1 PE-A	-6.733	-0.981	-0.318	-0.325	0.369	6.107	1.113
PDL1 PECF ECD-A	-1.949	-1.050	-0.841	-0.833	-0.626	2.097	0.334
CD3 PC5.5-A	-1.345	1.154	1.321	1.304	1.476	2.294	0.277
PD1 PC7-A	-4.081	0.033	0.854	0.640	1.415	6.069	1.096

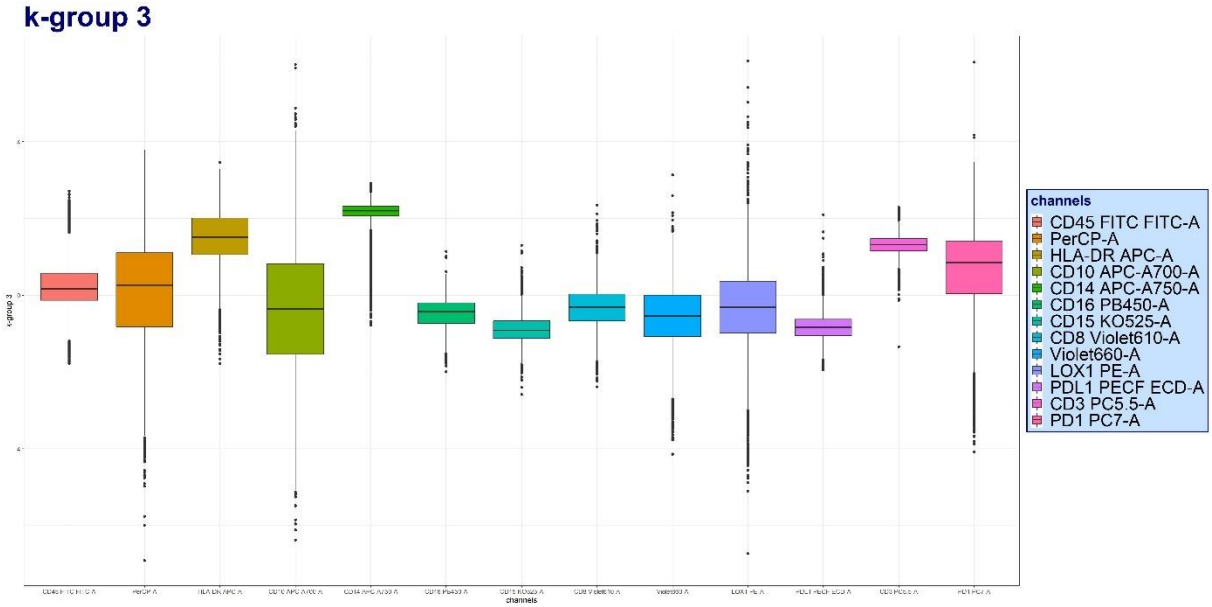


Figure 14: Boxplots k-group 3, for every channel

Just by looking at the different boxplots, we could easily realize the differences between the third group and the first and the second ones: in particular, we could expect very different values for mean and standard deviations.

In fact, we could confirm those first assumptions by looking at the summary: we had 4 values for standard deviations overcoming the unit, with a maximum of 1.718 (in the channel CD10 APC-A700-A, at this stage of the work it was the highest value of standard deviations found), and a couple more coming very close to values near 1; for mean values, while in the second group they were inside the -0.734 and 0.646 range, we had a lot of values higher than ±0.5, with even three values over the unit: 1.304 for CD3 PC5.5-A, 1.514 for HLA-DR APC-A and even a 2.076 CD14 APC-A750-A.

Like the first two groups, the third one presented some outliers too: in particular, the channels LOX1 PE-A, PD1 PC7-A and PerCP-A were the distributions with most outliers. PerCP-A, in group 3, was very odd for the outliers composition, as we were observing a distribution with no outliers for high values, but a lot of outliers for low ones.

The forth group, composed by 8068 observations, produced these statistics and these boxplots:

	Min	Q1	Median	Mean	Q3	Max	sd
CD45 FITC FITC-A	-1.784	1.112	1.405	1.360	1.687	2.959	0.555
PerCP-A	-3.091	-1.445	-1.271	-1.244	-1.081	3.114	0.311
HLA-DR APC-A	-2.076	-0.558	0.133	0.276	0.897	3.195	1.053
CD10 APC-A700-A	-4.700	-0.518	-0.159	-0.194	0.193	6.671	0.707
CD14 APC-A750-A	-1.446	-0.856	-0.697	-0.684	-0.530	0.471	0.253
CD16 PB450-A	-2.454	-1.585	-1.434	-1.339	-1.248	0.749	0.398
CD15 KO525-A	-3.814	-1.579	-1.404	-1.413	-1.242	0.964	0.248
CD8 Violet610-A	-2.652	-0.983	-0.676	-0.666	-0.362	3.902	0.478
Violet660-A	-3.644	-0.474	-0.025	-0.071	0.407	2.233	0.711
LOX1 PE-A	-6.816	-1.016	-0.734	-0.756	-0.462	3.834	0.533
PDL1 PECF ECD-A	-1.972	-1.305	-1.147	-1.141	-0.991	3.275	0.263
CD3 PC5.5-A	-3.454	-1.812	-1.701	-1.702	-1.603	1.501	0.302
PD1 PC7-A	-2.684	-0.333	0.268	0.277	0.850	5.879	0.866

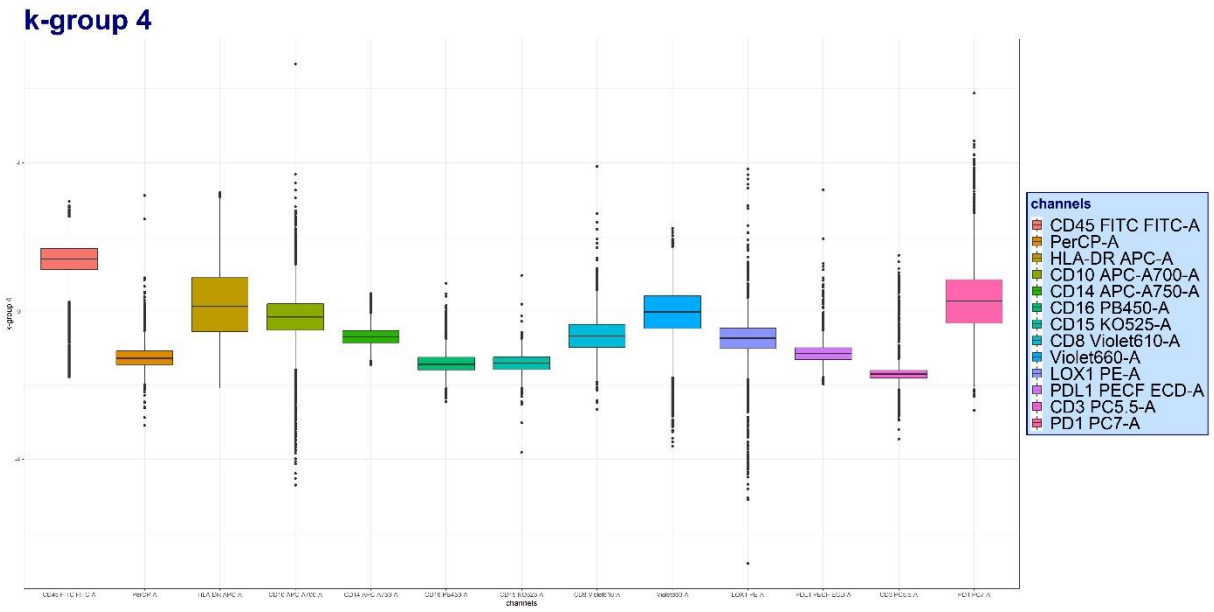


Figure 15: Boxplots k-group 4, for every channel

The forth group consisted of a lot of observations with mean values not close to 0: six channels out of fourteen had mean values over ±1, as we could see from both the boxplots and the summary. However, distributions were grouped together, as shown from both the boxplots and the standard deviations, with just one value over 1 (for channel HLA-DR APC-A, equals to 1.053), but a lot of values under 0.5.

Some channels had a very different distribution, compared to their distribution with the complete channel: the CD45 FITC FITC-A, for example, had 1.112 as first quartile, instead of -0.840, and a third quartile at 1.687 instead of 0.652; the channel PerCP-A had very low values, with both first (-1.445) and third quartile (-1.081) under 0; CD3 PC5.5-A had a first quartile at -1.812, while the complete data had a first quartile at -0.183.

All the channels presented a lot of outliers, in particular channels CD10 APC-A700-A and LOX1 PE-A, the same way as in the second k-group; other channels had a significant amount of high-value outliers too, like CD45 FITC FITC-A and PD1 PC7-A.

CONCLUSIONS

Thanks to the procedures available with the FlowCore package, we were able to work on unsupervised data utilizing automated methods: basically, the work previously made by the biologists through long and complicated procedures – the gating operations, the pre-processing steps to know the characteristics of the data involved in the study, the exploration phase previously essential to figure out the most suited number of metaclusters… – was no longer necessary, but it was all automated to lighten up the biologist work and to speed up the whole process.

After the initial phase of data preparation, we examined in depth few methods for dimensionality reduction to help us create visual output from input dataset, with a great amount of observations and variables of interest.

Between **t-SNE** and **UMAP**, the second one was clearly the more suited method to use with flow cytometry data: given the dimensions and complexity of the FCS files we worked with, UMAP helped us not only to obtain significant results in much shorter time – also with the possibility to replicate UMAP objects and to make useful comparison among all the clustering methods considered –, but also to create two-dimensional visualizations of the data with better defined metaclusters. This way our goal to clearly understand the real composition of the input data was a lot easier to achieve.

Regarding the clustering methods applied to our input dataset, we obtained few results that may be really helpful for future analysis.

The **K-means** method was without any doubt the best to work with: the four k-groups created in the UMAP plot were all well-separated, with a minimal amount of observations classified in different groups from the one they were near in the visual plots. In addition, the algorithm didn't take too much time to be processed. The only possible downside with this method was the

number of centroids required at the beginning of the process: it didn't directly derive from the algorithm but it needed to be predetermined with some pre-processing steps on data. But, apart from that, K-means was the best of all the methods analyzed.

The **flowSOM** produced some good results as well, but, compared to the K-means performances, it was a burdensome process for our devices, that took us a lot more time to produce good outputs. It worked pretty well on the clustering analysis too, except for the fact that one group was almost entirely merged with another one, leaving us with a small cluster not significant at all. But, all things considered, it was an useful method, with a lot of customizations available to work and to explore the data even more than using the K-means.

The **Kohonen SOM**, similar to the flowSOM, was a very customizable method to work with, also with great computation times needed too; unfortunately, it worked poorly on our data, merging almost all of our metaclusters as shown in the UMAP plot obtained with this method.

Last but not least, the **DBSCAN** was by far the heaviest method in terms of computational time, and surely the results weren't the best, even though the clustering results weren't as bad as the ones obtained with the Kohonen SOM: some metaclusters were represented pretty well, while others were scattered all over the UMAP plot.

After we chose the K-means as the most suited method to address the dimensionality reduction problem in our dataset, we showed the results of our procedures to the biologists of the structure, to analyze the real composition of the 4 k-groups we obtained: we were able to identify some leucocyte populations from peripheral circulating blood.

The **third group** – the green one in the UMAP plots – was composed mainly by monocytes, due to very high values for channel CD14 APC-A750-A (with mean of 2.076) and pretty low values for CD45 FITC FITC-A (mean equal to 0.294) and CD16 PB450-A (mean equal to -0.491) channels; the **first group** – the red one in the UMAP plots – was composed mainly by granulocytes, due to very low values for channel CD45 FITC FITC-A (with mean of -0.885) and pretty high values for CD15 KO525-A (mean equal to 0.868) and CD16 PB450-A (mean equal to 1.029) channels; the other two groups, the **second** (blue) and the **forth** one (yellow), needed further data, such as adding more channels to the analysis, to locate well-defined leucocyte populations.

RINGRAZIAMENTI

Prima di tutto vorrei ringraziare moltissimo il mio Tutor aziendale, nonché Correlatore, il Dott. Luca Lalli: per quanto mi ha insegnato e fatto crescere in questi mesi, non solo come statistico, con mille preziosi consigli ed insegnamenti di cui ho fatto tesoro, ma anche come persona; e per la grande pazienza e l'immensa disponibilità nel seguirmi e supportarmi in un progetto lungo, faticoso ma che alla fine ha portato grandi soddisfazioni.

Ringrazio moltissimo anche il Dirigente della struttura semplice di epidemiologia clinica e organizzazione trial presso la Fondazione IRCCS Istituto Nazionale Tumori, nonché supervisore di questo lavoro, il Dott. Luigi Mariani: per la grandissima competenza dimostrata, unita ad una profonda umanità, e per l'enorme supporto che ci ha fornito in tutte le fasi di quest'esperienza, dall'inizio dell'attività di stage fino alle fasi di ultimazione di questo lavoro.

In generale ringrazio l'Istituto IRCCS per la grande opportunità che mi ha offerto con questa attività di stage, che mi ha permesso di volgere un primo, vero sguardo sul mondo del lavoro in un contesto e ambito che ho davvero molto apprezzato.

Ringrazio tantissimo il Prof. Vincenzo Bagnardi, mio Tutor Accademico per l'attività di stage e Relatore, per la grande pazienza, gentilezza e disponibilità, e per avermi portato, con il suo corso di Elementi di Biostatistica, a fare un'ottima scelta come percorso di tirocinio in azienda.

Ringrazio l'Università degli studi di Milano - Bicocca per la formazione e il supporto ricevuti in questi anni di crescita, come persona, studente, statistico e cittadino.

Ringrazio tutta la mia famiglia, per il sostegno ricevuto costantemente in questi anni, nei momenti migliori come nei momenti di difficoltà, e per aver sempre creduto in me, a volte più di quanto io credessi in me stesso; ringrazio i miei amici, per aver rappresentato la mia boccata di ossigeno nei momenti di sconforto, e per aver gioito assieme a me nei momenti più lieti.

Infine, ringrazio anche me stesso, per aver perseverato e insistito in un percorso lungo e difficile, ma che alla fine mi ha portato grandi soddisfazioni!

BIBLIOGRAPHY

- Metodi di riduzione della dimensionalità, Marco Fattore, 2020
- Appunti corso di Data Mining e Statistica Computazionale, Pier Giorgio Lovaglio, corso di laurea triennale SGI, 2020
- Flow Cytometry Basic Guide, Bio-Rad
- UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction, Leland McInnes, John Healy and James Melville, 2020
- Reference Manual Package "FlowCore", 2021
- Update for the Logicle Data Scale Including Operational Code Implementations, Wayne A. Moore and David R. Parks, 2016
- Genetic K-Means Algorithm, K. Krishna and M. Narasimha Murty, 1999
- FlowSOM: Using self-organizing maps for visualization and interpretation of cytometry data, Sofie Van Gassen, 2015
- Extending the Kohonen self-organizing map networks for clustering analysis, Melody Y.Kiang, 2001
- DBSCAN: Past, Present and Future: Kamran Khan, Kamran Aziz and Simon Fong, 2009
- https://distill.pub/2016/misread-tsne/?_ga=2.135835192.888864733.1531353600-1779571267.1531353600