cytoChain

A web app to handle high-dimensional flow cytometry experiments

Table of Contents

1 Abstract

The availability of new cytometry systems which enables the simultaneous measurement of more than 20 different characteristics of a single cell raise the bar much higher in terms of data analysis effort and result interpretation. Definitely, the manual gating approach to select the cell of interest is not efficient if the aim is to compare different samples in different conditions.

Clustering, the well-known unsupervised machine learning technique, which is widely used in this context is just an important step of a chain of many actions (modules) to be performed in order to achieve an exhaustive and significant statistical analysis.

For this reason, more than often, the analysis should also be rearranged using many different environment and applications in order to cope with different criteria and needs, passing from one application to another, using spreadsheets and visualization tools from many different environments and developers, dealing with plenty of different formats.

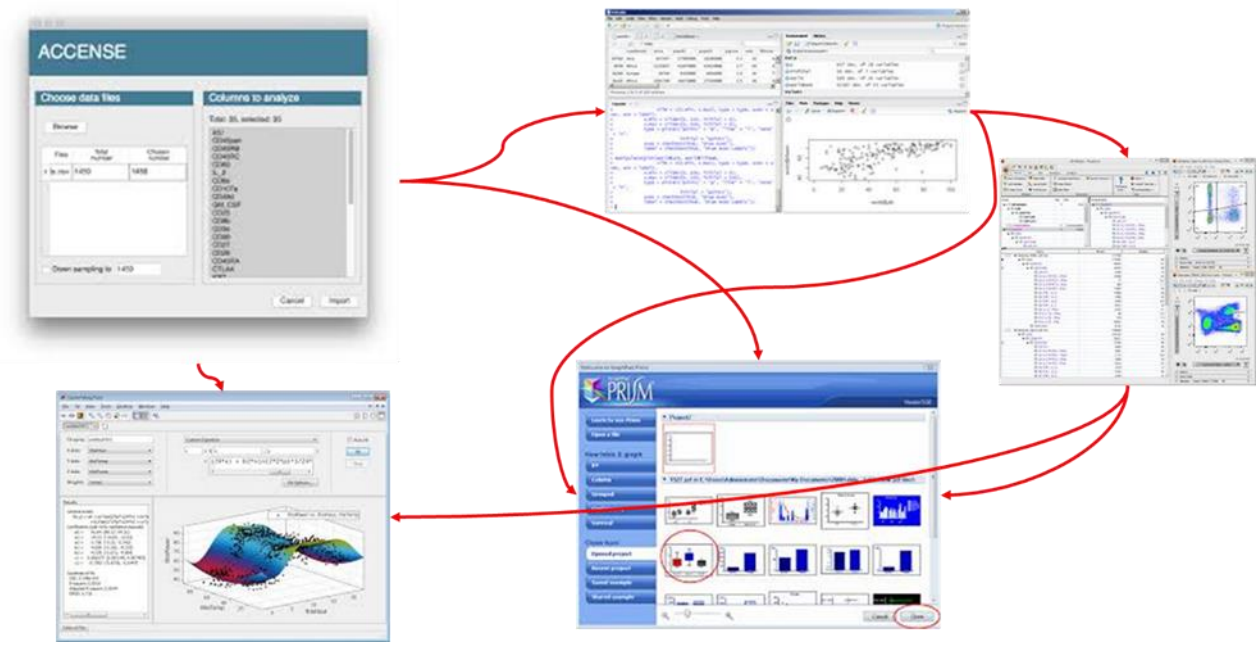


Figure 1 - A typical analysis workflow

In fact, the huge amount of resources available (think for example about the Bioconductor repository for the cytometry) could be seen as an unprecedented opportunity but as well as an overwhelming effort to integrate the various component of a meaningful, fast and comprehensive methodology.

Recently, several workflows have been exposed to suggest a meaningful set of operations to perform a complete analysis of a selected dataset. Among the others, the *CyTOF workflow: differential discovery in high-throughput high-dimensional cytometry datasets - BioC 2017* workshop and its more recent versions

available on Bioconductor seems very promising to collect the various steps in a concrete pipeline of modules.

One of the main part of these type of workflows (or *pipelines*) approach is to focus on the dataset preparation, a sort of pre-analysis sequence of actions having the scope to clean, transforming, down-sampling the dataset before to start the clustering and so the core of the analysis. A comprehensive workflow could be composed of many other analysis step or operations which could be fundamental to achieve a significant results: the way to organize such a kind of analysis is to go for a step by step chain of operations to be integrated in a single and modifiable pipeline.

Nevertheless, the difficulty for the vast reaches of researchers and biologist to implement such a pipeline in a typical IT developer's environment, suggest a more straightforward and transparent approach without messing with compilers, software packaging and computer languages.

Take into account the effort and the amount of resources also in terms of productivity that a fully operative platform would need: the language interpreter or compiler, the programming IDE, the whole packages library bunch… everything correctly installed, up and running on a consolidated platform with all the necessary memory amount, with a running operating system and of course a network to share the results.

The latter is the only aspect a web application would need for a true technology independent platform in a widely used *SaaS* (Software as a Service) deployment frame. Through a common web browser, all the analysis blocks are available in a unique integrated environment (dataset optimization; data fitness; clustering; meta-clustering; plots and data result). In fact, in this way the cytoflorimetry experts which does not have a deep informatics knowledge could access via web the cytoChain and simply loading the dataset to setup a complete analysis without the need of a special installation on a particular platform with the necessary amount of recourses that such a kind of analysis normally requires in terms of performance. Being a web-app, the user only needs a web browser to access the program: a full set of plots and results are downloadable, for further analysis or to simple be used directly on the paper or in any kind of report.

The other main advantage is in term of productivity: typically a complete analysis needs a tedious and complex process iteration with time wasting jumping from one program to another, for example, from the manual gating application, to a fully configured R compiler environment, from a spreadsheet to a graphic facility for the plot generation and so on (like shown in Figure 1). With the centralized cytoChain you can set up a full analysis with a priceless advantage in terms of effort and time.

It should be noted that the all the algorithm used in the *cytoChain* workflows are well known and abundantly used in the various published works in literature. This is a choice in order to perform every analysis with known packages which are well established and commonly accepted in the scientific community. This could help the user to compare the results and to analyze the dataset with mostly trustable criteria.

One of the main aspect of such a complex analysis frame is the reproducibility (which is one of the main aim in order to be able to compare different analysis results on the same data input). In fact, as you will have reasons to discover through the different part of the analysis workflows, there are bunch of parameters and choices to select to be able to tune your analysis.

This important achievement is also pursued with the help of the settable pseudo-random seed (a seed setting in each used algorithm – the seed in *cytoChain* is just another parameter which can be set). Since almost every main algorithm used (starting from the clustering but also the dimensionality reduction algorithms) are based on a pseudo-random number choice which can affect the analysis outcomes: most of the cytoflorimetry experts knows even too well the rather difficult journey getting some results which may be really comparable among different analysis...

With these considerations some questions are immediately raised.

- How to be free from the burden of a fully configured computing platform?
- How to make available a 360° complete, streamlined and intuitive light platform?
- How to discover new marker interactions?
- How to get through the countless packages parameters in order to tune your analysis?

The aim of ***cytoChain*** is to group a selected an expandable set of resources in a single easy and transparent environment, leveraging all the typical informatics related overhead for the vast community of the cytometrist.

For this reason rather than a stand-alone application (see for example the valuable cytofkit, which needs the whole R environment up and running in the research's computer) we present a web-app, a program which runs on a web-server and can be reached out from a simple web-browser: (see the web-app architecture). Of course it could be always possible to run cytoChain as a stand-alone package in a more traditional way, once you have the whole environment up and running in your computer and you install everything you need (R compilers with all the related packages installed) and the cytoChain source files.

2 Introduction

The advantages of a high-dimensional cytometry approach could be overshadowed by the challenge of the analysis of the large-scale datasets produced by instruments like the BD FACSymphony and many other cytometry platforms on the market. Even if the traditional manual gating, the gold-standard method for flow cytometry data analysis, is still a valuable resource in evaluating the effective amount of the various population events, it is not a practical approach for the high dimensionality dataset especially in evaluating the target cell in different conditions and in distinct data samples (or flowFrames)[1].

On top of that, the amount of possible way to group similar events in clusters explode with the amount of dimensions. There are $\frac{N(N-1)}{2}$ biaxial plots in an *N* dimensional dataset (where *N* is the number of selected markers[2] for each of the registered event and M, the number of events which defines the $M * N$ matrix expression) like the one in Figure 2. Things get worse if the events are spread along several markers and not in a Boolean distribution with all the events grouped in a Gaussian space like in Figure 3.

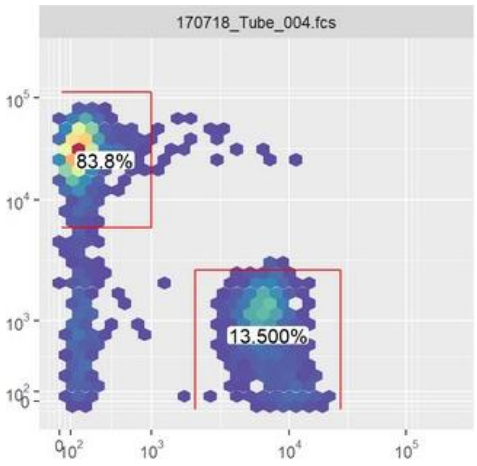


Figure 2 – Boolean event distribution type

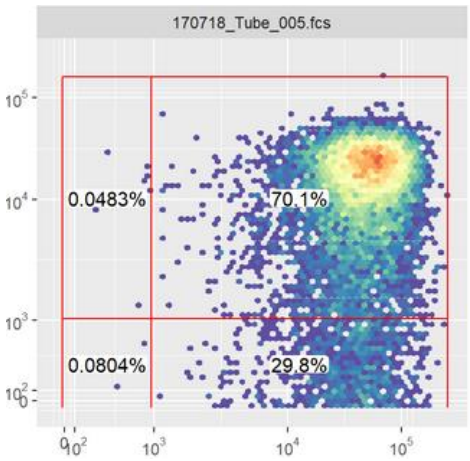


Figure 3 – Spread events distribution type

In this last case, which is the most common in a typical dataset, the number of possible homogenous group of events (clusters) could reach the values shown in Figure 4 (notice the logarithmic scale!).

[1] We follow the BioConductor convention introduced by flowCore package to call *flowFrame* the single sample formatted as the Data File Standard for Flow Cytometry, Version FCS3.0 (typically a file with the *.FCS* extension produced by the cytoflurimeter) and to call *flowSet*, the set of the selected flowFrames having the same characteristics in terms of markers name and descriptions.

[2] Marker is a synonymous of many other common words in this field. An incomplete list could be: channel, stain, dimension, signal…

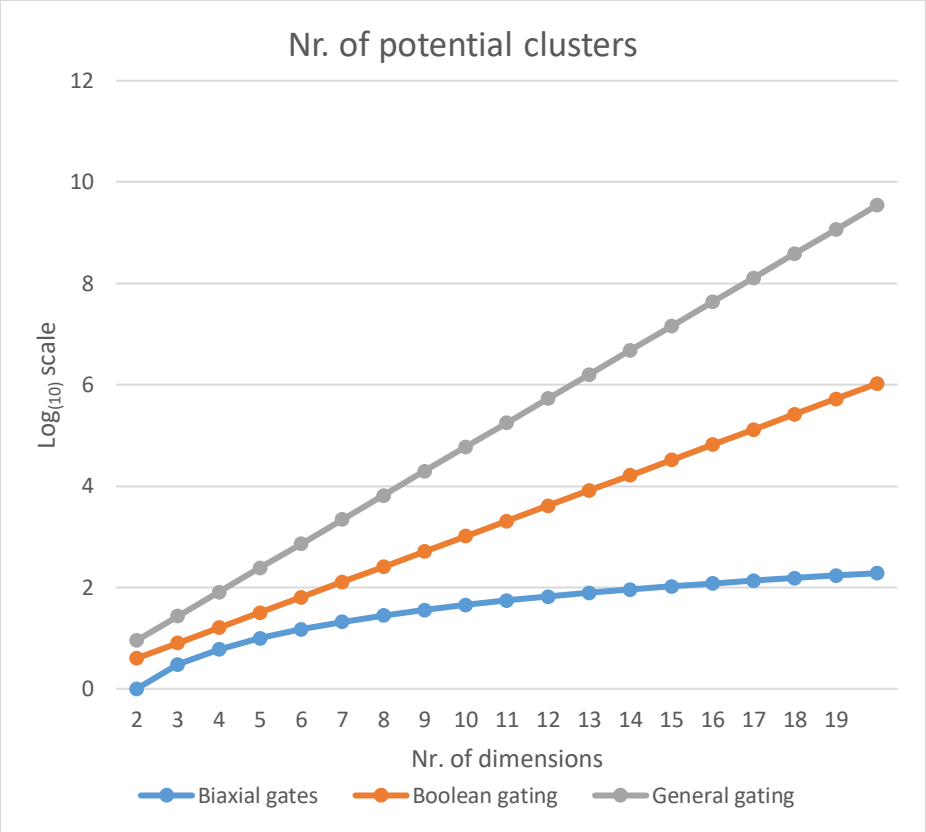


Figure 4- Clusters in different cases

Even if a much smaller amount of these theoretical values have a real counterpart in a significant biology analysis, it could be a prohibitive amount to handle.

To reduce complexity, still maintaining discovery ability, the only approach is a structured clustering with the implementation of several algorithms which could organizes data allowing to highlight similarities putting them in a space with a lower degree of complexity in a guided methodology. Besides other factors should be taken into account like:

- the flow cytometry dataset contains systemic errors
- minor differences in marker expression may have large functional consequences and vice versa
- a perfect clustering method do not exist and it's sensitive to data distribution and parametrization
- every related algorithm do not scale well with the number of events and dimensions

Another efficient and widely used method, which is complementary to the clustering method, is the dimensionality reduction. Several dimensionality reduction methods has been developed (see tSNE (Maaten 2008), UMAP (Leland McInnes 2018), ISOMAP) and many other are under studying (e.g. opt-SNE and PHATE) to implement alternative solutions in order to be able to depict in maps making it possible to visualize such a complicated space of events. Some of these are already implemented inside the cytoChain workflow. These future implementation represent the will of keeping the cytoChain web-app always updated with the best-in-breed algorithm always available and integrated along with the rest of the workflows.

2.1 The User Interface (UI)

In order to better illustrate the concept behind the cytoChain UI (User Interface) and its capability we will briefly show just the first workflow (the optimization flowSet workflow) since this holds for each single workflow implemented.

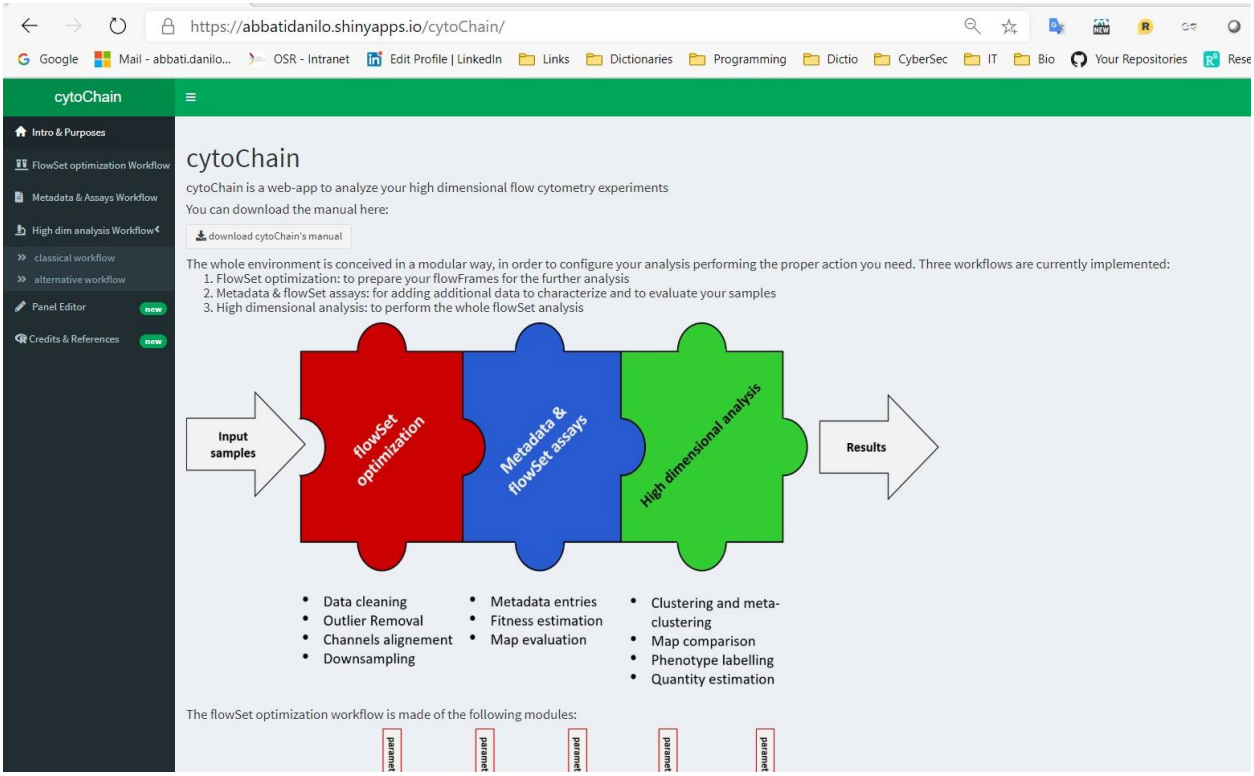


Figure 5 - The UI showing a partial view of the the optimization flowSet workflow

With the most left-side panel it is possible to select the workflows, while each tab in the related workflow page allows the user to browse and configure the related module of the chain. In every page then the whole set of the available parameter for the selected workflow is listed on the middle left-side panel.

To upload your samples, just move to the **Loading and parsing samples** tab inside the **optimization flowSet workflow** and press the **zip** button (currently the https://www.shinyapps.io/ server does not allow the upload of normal *.FCS* files). So then, compress your *.FCS* files first and then upload them as a single compressed zip file.

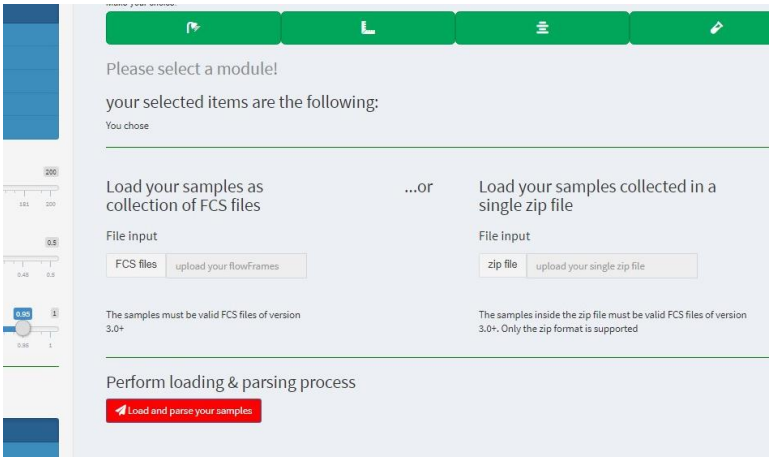
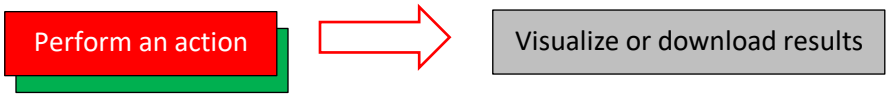


Figure 6 - The mask to upload your sample's files

After the upload is completed, let cytoChain parsing your dataset by pressing the **Load and parse your samples** red button. The operation may takes some time, depending on the size of your samples: check the console message little window on the top left sidebar and wait for the confirmation message.

When the parsing step is executed you can visualize some reports pressing the **show sample's details** grey button. In the whole cytoChain environment this is the common way to proceed:



Command to perform an action (typically pressing red or green buttons) → visualize or download data results (pressing grey buttons).

You could also upload you file directly from the second workflow and more precisely on the **edit metadata** tab of the **Metadata and assays workflow.** This may be convenient in case you have already performed all the relevant sample handling on the pre-analysis **optimization flowSet workflow** and you want skip that just uploading your files starting directly on the assays workflow to study your samples and then go for the multidimensional analysis.

2.2 The web-app architecture

The whole web-app architecture follows the *SaaS* (Software as a Service) paradigm (Figure 7): the program runs on a remote server which is accessible via a normal internet connection via the http protocol. This means the whole computational charge is left on the server side: the user control and download all the relevant analysis results via the user interface available with a normal web browser. Both the UI (the front-end) and the server side (the back-end) is realized using the shiny package.[3]

[3] The cytoChain page is currently reachable typing https://abbatidanilo.shinyapps.io/cytoChain/ in a web browser.

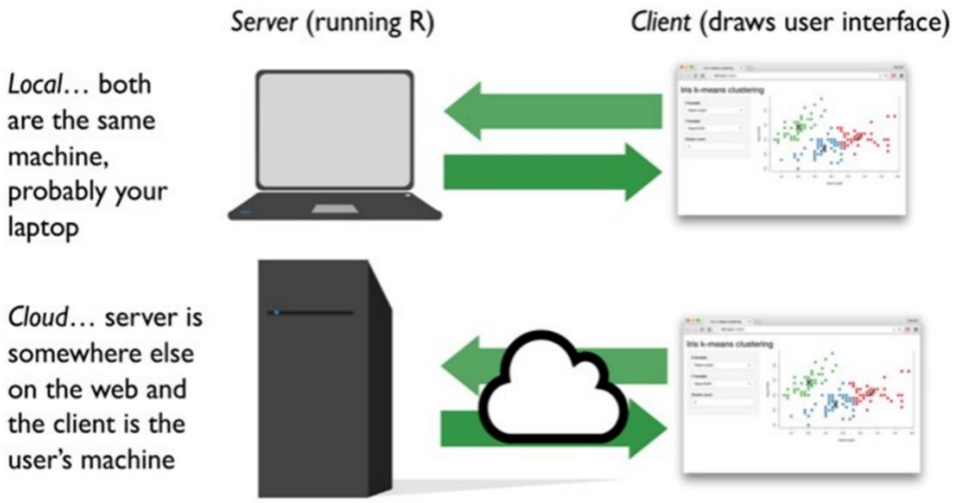


Figure 7 - The web-app architecture

Most of the current available shiny web-app are used to show (often in a run-time way) results or plots and they are rarely used to execute heavy computational tasks like in this case. For this reason some words have to be spent in order to scratch the surface of this important aspect.

The shinyapps.io server currently in use, forces the user to follow standard deployment procedures which cannot be further handled and with a few parameter to properly tune the web-app operability. A lot of fundamental configuration cannot be touched (like the maximum usable memory and the amount of GPU for multi-threading operation).

A further deployment will let cytoChain to unchain all its potential and remove most of the current limitation in terms of resources and performance. We are still looking for a good hosting server for our SaaS implementation for a better cytoChain deployment.

3 The *cytoChain* concept and its implementation

We have integrated in a single web-app three data analysis workflows which covers the essential parts of a comprehensive high-dimensional flow cytometry dataset analysis:

- The flowSet Optimization workflow
- The Metadata & Assays workflow
- The High Dimensional Analysis workflows

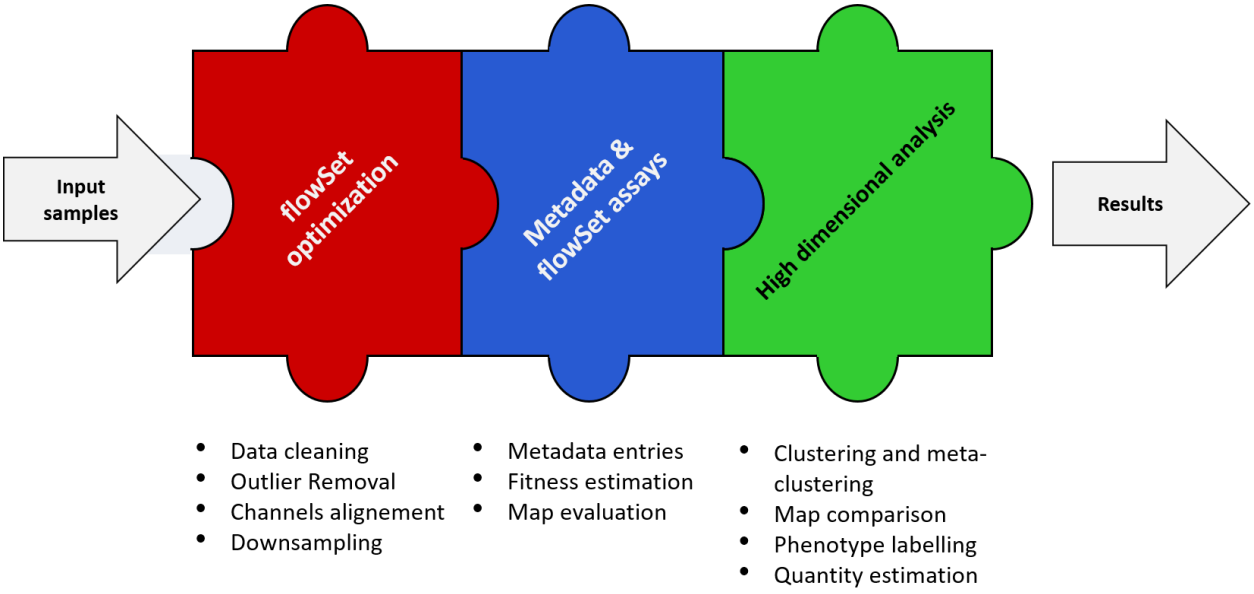


Figure 8 - The three implemented cytoChain workflows

The workflows with their parts are shown in the following pictures:

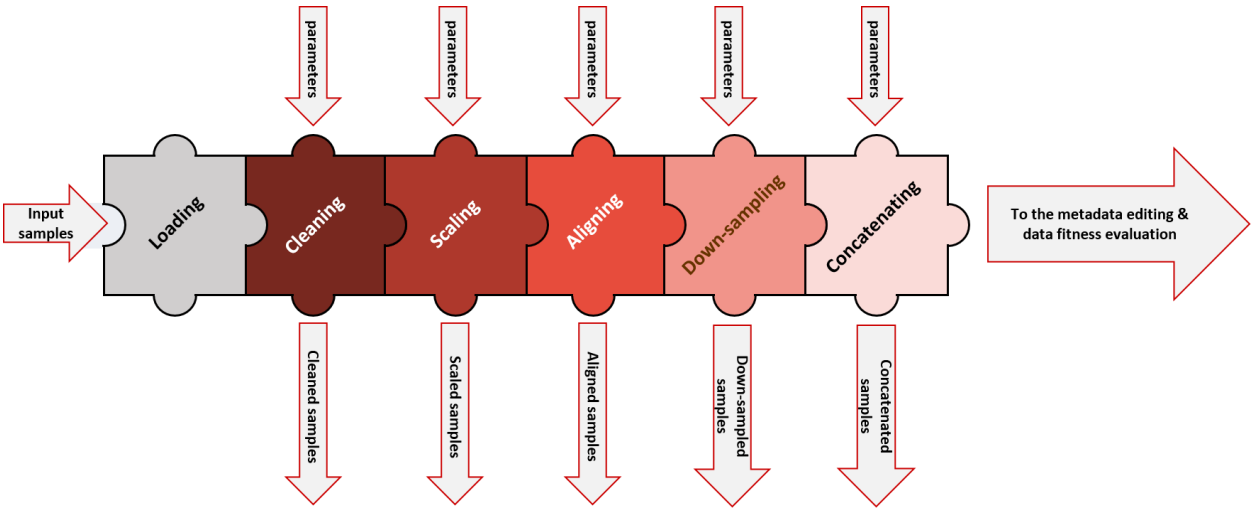


Figure 9 - The flowSet optimization workflow

The input samples are a collection of the *.fcs* files (aka the flowSet)[4] produced by the cytoflurimeter.

The expression matrix of each single flowFrame represent the row data, the input for any type of analysis, with for example the manual gating strategy, as well as with any other kind of algorithm. The *flowSet optimization* (Figure 9) is the essential part to prepare the row data for any type of algorithm. The expression matrix is composed of rows and columns. In general, the row represents one example from the problem domain, and may be referred to as an "example", an "instance", or a "case", but in case of flow cytometry is the snapshot of a cell: the *event* to analyze. A column represents the properties observed about the event and may be referred (in the machine learning jargon) to as a "variable", a "feature", or an "attribute": in our case is the marker of the sample expression matrix.

The sample loading is possible through the *Loading and Parsing Samples* tab in *flowSet Optimization Workflow* but *cytoChain* allows also this operation by using the *file input* mask in the *Metadata and Assays* workflow. This is implemented to let you load your flowSet directly without passing through the pre-clustering operations. In the latter case you cannot use the features available in the *Map evaluation* tab which is useful to compare your sample behavior before and after the pre-clustering treatment.

---

[4] A *flowSet* (or *cytoSet*) is the conventional name used in R environment in order to handle and to parse the fcs files which are the basic file format of your cytometer output. A *flowFrame* (or *cytoFrame*) is the single sample, namely the single experiment (the fcs file is, in digital format, the content of the 'tube' full of your cells loaded to the cytometer); the *flowSet* is a collection of flowFrames. Each *flowFrame* contains many relevant data about the experiment, but the main content is the expression matrix: an *n x m* real number matrix values in which each row correspond to a single event measured by the cytometer. The measure of each event (typically a single cell or event) is the coordinate of a point in the m-dimensional space of the marker expression: every point has *m* levels of expression, one column for each channel (or stain). The whole channel set defines the *panel* used for the experiment. Along with the marker expression variable, another important entry could be present: the time, which reflect the relative time acquisition, one entry for each event. For every detail about the *.fcs* format see fcs3 page. All about flowFrame parsing and content handling in this tool is based on the *flowCore* library (see the APPENDIX)

Parsing the samples means reading and interpreting your flowSet (which can be made also of a single flowFrame, a single tube) and this is borne witness to by summary tables and figures shown when you press the *Showing sample's details* button. Most of the operations in cytoChain are implemented in this two steps fashion: perform the operation first and show the results: you can always iterate the same operation, changing for example some entry parameters, until you are satisfied with what you achieve.

Notice that for every step (in every context where this could make any sense) it is possible to download the related flowSet in order to utilize the desired module along the chain and to skip the others. With this, is also possible to re-define the operation order of the chain (e.g. suppose you do not want to perform the alignment operation, is always possible to skip it; furthermore, suppose you want to perform the down-sampling after the sub-setting, you can skip the down-sampling module, download the sub-set flowSet and then reload it as a new sample input to perform the previously skipped down-sampling operation).

The importance of having the dataset available in the parsed flowSet format at each level of a structured analysis is often underestimated and rarely implemented in other tools. This capability is also useful to utilize the cytoChain analysis result as an input to other environments, even if the spirit of this tool is to keep grouping in one single tool the whole analysis.

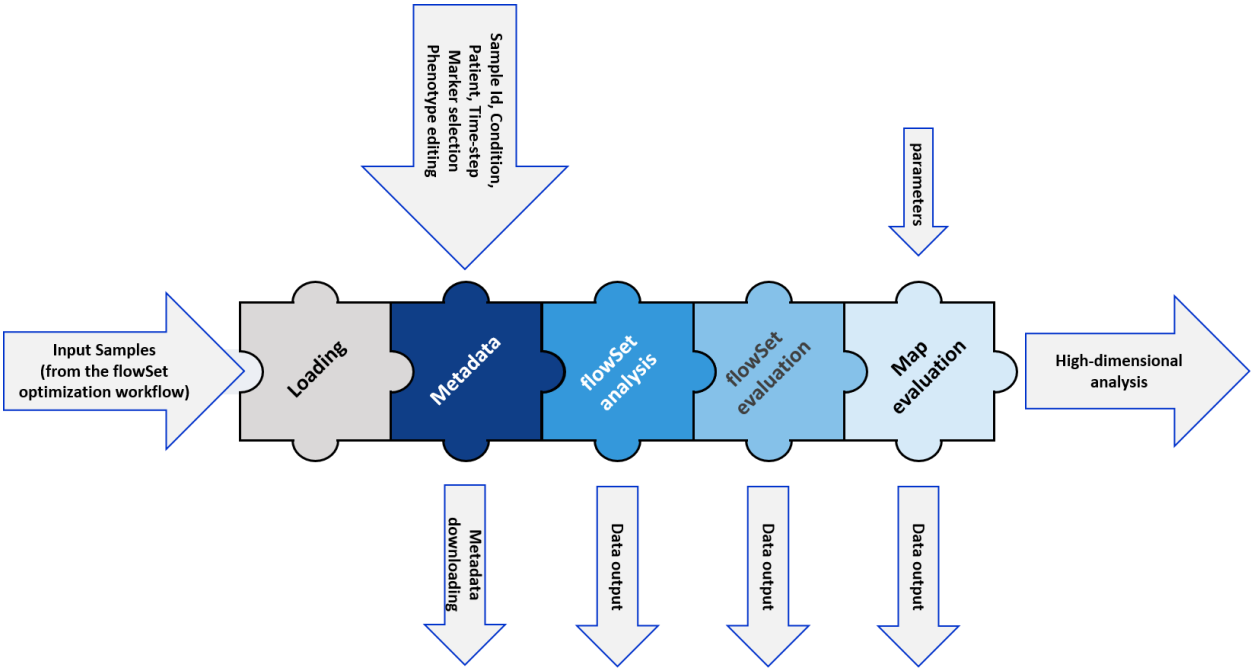


Figure 10 - The metadata & data fitness workflow

Notice also the main aspect of the module chain when it needs to change the default parameters to adapt and tune the user specific sample used: every parameter setting affect the calculus and the result of the downstream modules. For example, let's focus on the clustering workflow (see Figure 10): suppose you want to change the setting of the clustering algorithm with a new number of clusters and then to tune the selected map dimensionality reduction (e.g. a tSNE map) with a different perplexity parameter: the number of cluster, being an upstream module (the clustering module) in relation to the mapping

operation (in which the tSNE map is performed) will then affect this last operation's result because the *clustering* comes before *mapping* in the pipeline, preceding the mapping.

As previously stated, it could be always possible to ex-change the module order, since the download of the related flowSet can be performed in each module of the chain.

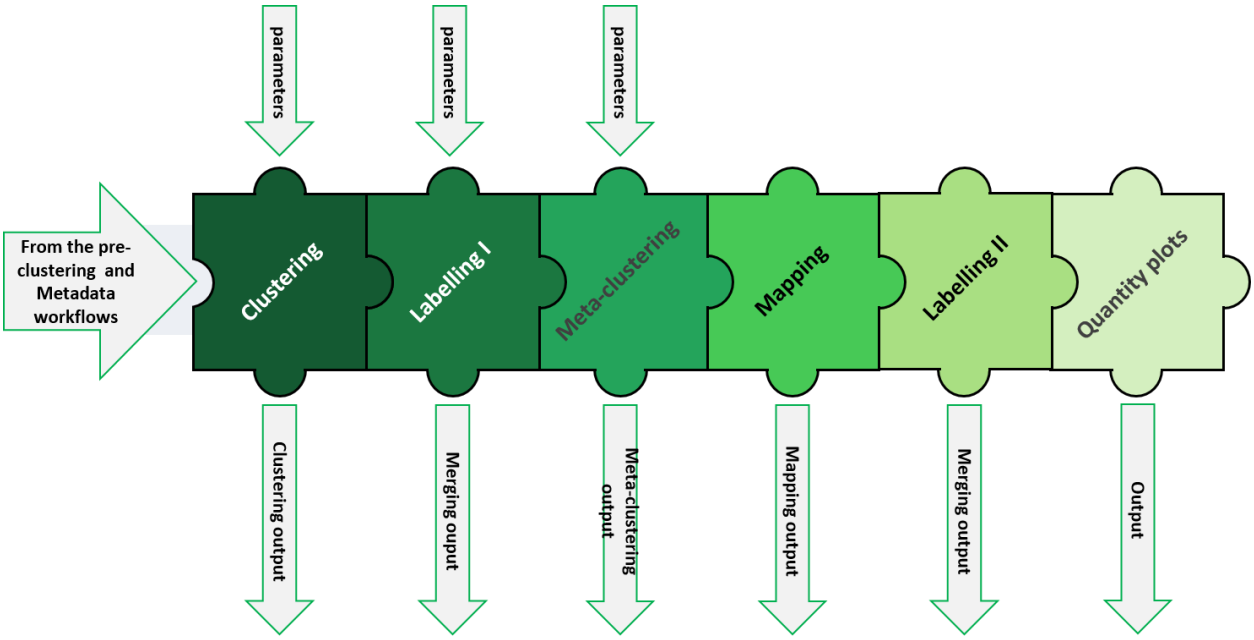


Figure 11 – The high dimensional analysis workflows

Also the tables and the figures at each level of the analysis can be downloaded and saved in order to be further integrated in the user documentation or into the publishing paper.

The high dimensional analysis workflow(s)[5] described as the "Classical" clustering workflow since it is the only one implemented in cytoChain and the only studied up to now. Other possible pipelines will be implemented soon, providing a further extension to the this tool, capable of further features which can be also used to compare the results obtained by the "Classical" version.

[5] It is plural because the analysis workflow is in turn composed of three different workflows (see 7)

4 A few general remarks on the flowSet to analyze

These are few concise notes on the samples you are going to use for these kind of analysis. Most of them are valid for any kind of *in-silico* analysis, even to be used with other tools.

4.1 Pre-gating the sample

In current implementation cytoChain does not offer any support for selecting the type of events (namely, the cells) to be analyzed. This mainly because, selecting the pertinent events of the experiments is closely related to the type of samples to analyzer. Please notice than by now, flow cytometry is not only concerned to blood cells but also with several types of tissues, coming for example, from tumor or peri-tumoral tissues. The selection of the relevant part of the experiment to analyze and thus filtering out the non-relevant events is mostly based on the experience of the researcher and it should be optimized for the whole flowSet. In every flow cytometry experiment, the following basic gating strategy should be processed first, to select the relevant events:

- **Forward scatter gating**: It is the basic gating for selecting the size (area and height) of the cell and it must be used to differentiate various cells and debris. Typically the FSC-A and FSC-H physical channels are used for this purpose. Using this information, it is possible to gate the flow cytometry data so that it is possible to detect various white blood cells and distinguish those from debris.
- **Side scattering gating**: It is related to the cell's granularity. Doublets are when the flow cytometry detects a single cell, but it was in fact two cells in close proximity. This can cause problems with data analysis, but gating by scatter height vs scatter area (typically SSC-A and FSC-A), doublets can be identified and thus removed.
- **Live Cells gating**: It could be useful to further select your interesting events, discarding the death cells. This is not an essential gate to perform (unlike the previous ones), but if you filter out the live cells (typically with 7-AAD marker which appears to be generally excluded from live cells, or with other vitality marker) remember to exclude the related marker from your analysis when you select them (see 6.1).

In all, doublets, debris and dead cells must be filtered out from FCS files prior to analysis[6].

Before supplying inputs to input to cytoChain be sure to accomplish this kind of process first.

[6] These basic gating can be implemented in a further tool deployment for the sake of completeness but no automatic algorithm could be compared to biologist and their cytometry experience.

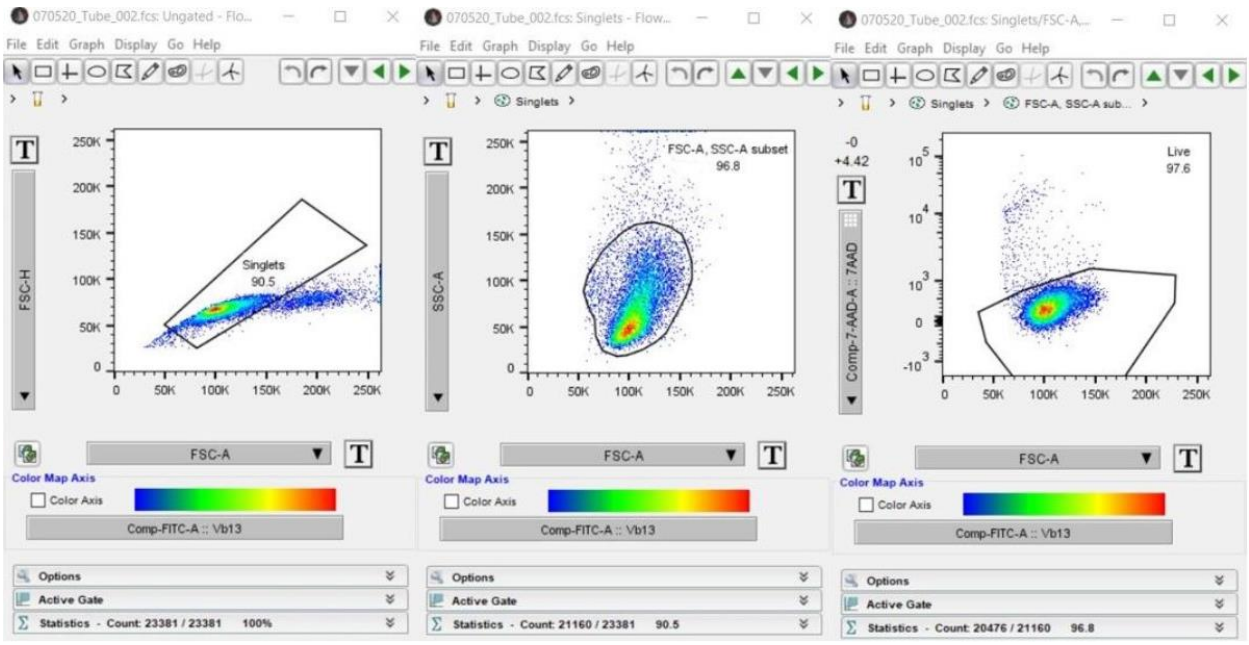


Figure 12 - The three manual gating to select the proper events

An explicative case could be the lymphocytes analysis, in which all the events will be CD3 positive: please filter out with your favorite gating strategy all the CD3 negative events in order to focus in your relevant population.

In this way remember to remove the CD3 stain from your analysis (see 6.1.2 – on the marker selection) otherwise will only add some useless noise to your analysis: the expression of your stain will be almost flat for all the event (with some fluctuation producing false positive) and it would get more all less the same density profiles for the whole flowSet: this can make the clustering difficult, hurting your deliverability, adding useless noise to a complex operation. Please notice that (as explained in the Introduction) potentially a clustering algorithm has to select among a 2^m (being *m* the number of markers) different group of homogenous events

A further gate could be to select the living cells only (with a vitality marker) in order to properly input cytoChain only with the relevant events. As before, do not forget to not selecting the related vitality markers from your analysis.

You could remove the entire dimensions from the expression matrix (with the advantage of having lighter FCS files) or you can keep it and not select them during the dataset analysis.

4.2 The sample numerousness

With most of the flow-cytometer you can easily collect samples with some millions of events. That is definitely too much even in case you want to look into some very rare population. Bear in mind that we are dealing with clustering, which is basically a type of unsupervised learning method. Even the most efficient clustering algorithm cannot work in the best way with every flowSet. In short, to look for a cluster of event of less than 0.01% of the entire population can be very complicated, even if have millions of events (as a sum of the entire flowSet) unless the group of event is a well-made cluster (a group of very

close events). If you know in advance the proportion of the rare population, please base on this value the numerousness of your flowSet and do not exaggerate the numerousness. It is always good not to exceed the 2 million[7] events. You can always reduce such amount within the tool, using the down-sampling procedure and in particular with the down-sampling by equalize procedure (see 5.4.3).

Further note about the number of events are in 5.6.3.

4.3 Notes on the small samples

Beware of the small samples!

It often happens to get from the machinery samples with a very few events. cytoChain is currently calibrated with a MIN_SAMPLE_LENGTH of 50 events: an FCS file with less of this amount cannot be part of your flowSet.

As a rule of thumb, if you deal with a FCS file of less than 10KB, consider to load it singularly (from the flowSet optimization workflow) to check weather is well formed: if cytoChain replies with an error, discard the entire sample.

A useful recommendation could be to filter your original sample thought the procedure described in 4.1, but avoid any further gating in order to leave as much as events as possible (a typical case could be the further gate the lymphocytes events (CD3+) splitting the Helper T cells (selecting the CD4+ cells) and the Cytotoxic T cells (gating the CD8+)

In any case, take into consideration that the whole high-dimensional analysis is based on a statistical approach: having samples with too few events could bring to weak results especially for comparison and trending purposes. It could be always interesting to find some rare events though: that's the only case in which could have some meaning to deal with too small samples.

[7] E.g.: if your rare population is 0.2% of the whole event amount of 10000, then a population of 200 events could be an achievable target, while a 0.02% could be a difficult task unless your rare population is a very well defined and concentrated spot in the hyperspace of you marker dimension
Currently the limit in the shinyapps.io server implementation is around 400Mbye for the whole flowSet.

5 The flowSet Optimization Workflow

As most of the analysis, the data has to be properly prepared in order to eventually subject the data to the real analysis. For this reason cytoChain implements a first pre-clustering sequence of operation, called the Optimization workflow.

The flowSet optimization pipeline is composed by (see 5):

- **Cleaning** (Quality check): the aim is to remove the bad events. The current implementation automatize the *flowAI* package (see https://www.ncbi.nlm.nih.gov/pubmed/27153628) to remove several type of suspected events which for the subsequent analysis represent a source of noise, potentially dangerous for the clustering algorithms.
- **Scaling** (transforming): to normalize the parameters. Many transformation are available with all the sensible parameters to tune your dataset
- **Aligning**: to perform the Gaussian density spots alignment of the various flowFrame. This is a beta implementation of the *flowStats* algorithm (see https://bioconductor.riken.jp/packages/3.1/bioc/html/flowStats.html)
- **Down-sampling**: the main scope here is to remove the outliers.
- **Concatenating**: for the subsequent analysis workflow. In fact, in order to allow the comparison of the various dataset for each flowFrame, that is, for each sample in your flowSet

Many articles illustrates the benefit of having a clean and well prepared dataset to analyze, but most of all our experience and the testing we have performed with several dataset.

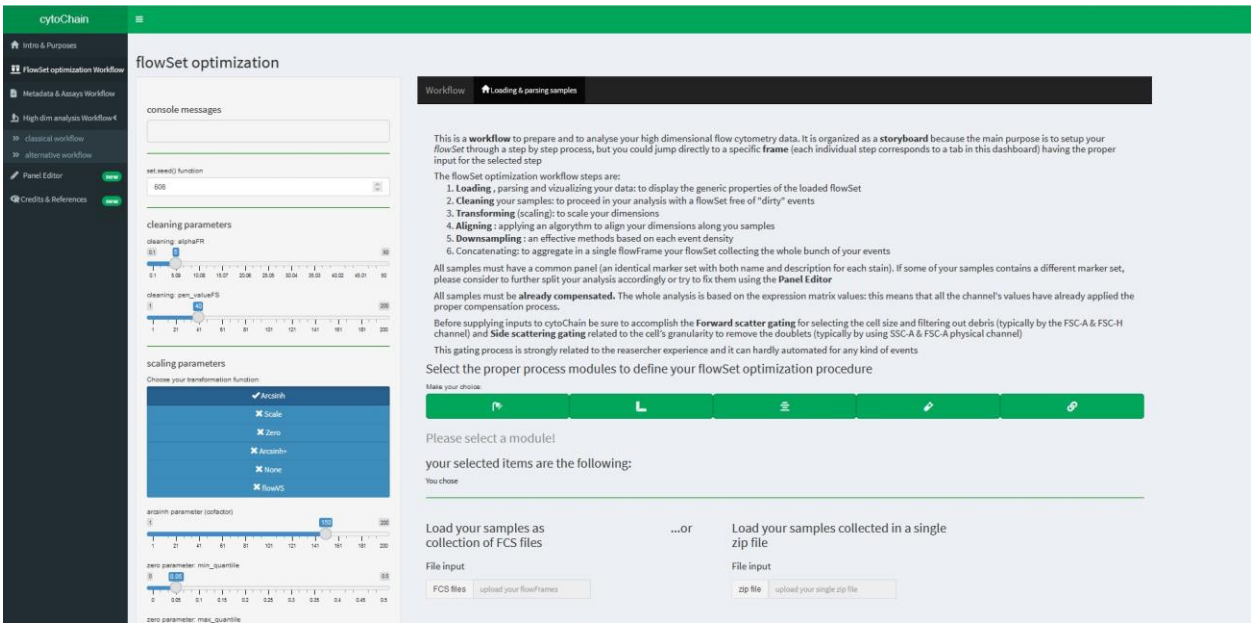


Figure 13 - The flowSet optimization workflow interface as it appears from a common web-browser

5.1 The Cleaning

The most significant part of the flowSet optimization relies in the interface between the physical data acquisition from the machinery and the *in-silico* data handling. In order to better illustrate the advantage of having a clean dataset to work on let's focus on the quality check and the down-sampling. The quality check is performed on each sample and its purpose is well depicted in the Figure 12.

Among the various pre-analysis procedures, cleaning has a major effect on the analysis' results: this means that should be performed first, before any other action.

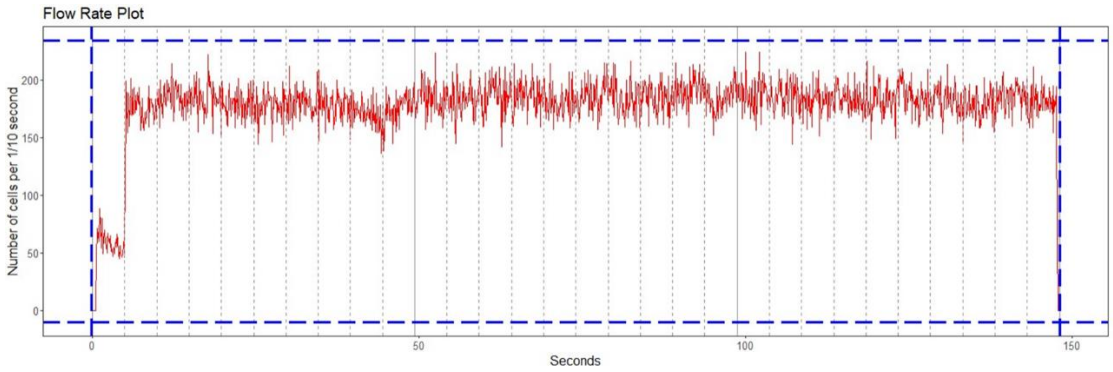


Figure 14 - the flowAI flow rate plot

Cutting the first and the last part of the total events, selecting just the most stable flow rate is important because the event related to the unstable parts it will surely clutter the whole analysis and it might tarnish every clustering method. This flow rate check and the other two methods (the signal acquisition check and the dynamic range check) are implemented in a non-interactive way inside the cytoChain web-app.

Figure 12 is taken from the flowAI vignette to represent one of the three steps of the quality control check done for each single sample:

1) Flow rate assessment: the sample quality control check is performed based on the global trend along the time event acquisition (see the 4 footnote). Its sensitive is controlled by the *alphaFR* parameter in the related sidebar.
2) Signal acquisition assessment: which is based on a change-point detection method on each single channel (marker) to verify the stability of the signal. The sensitivity depends on the *pen_valueFS* parameter.
3) Dynamic range assessment: to remove the events exceeding the upper and lower limits of the range, namely to get rid of outliers candidates.

Inside cytoChain the whole quality check (all three steps) is done automatically and can be set using the parameters mentioned.

The visualization of the outliers is offered by a dynamic plot reported in Figure 15 - the outliers after the quality check.

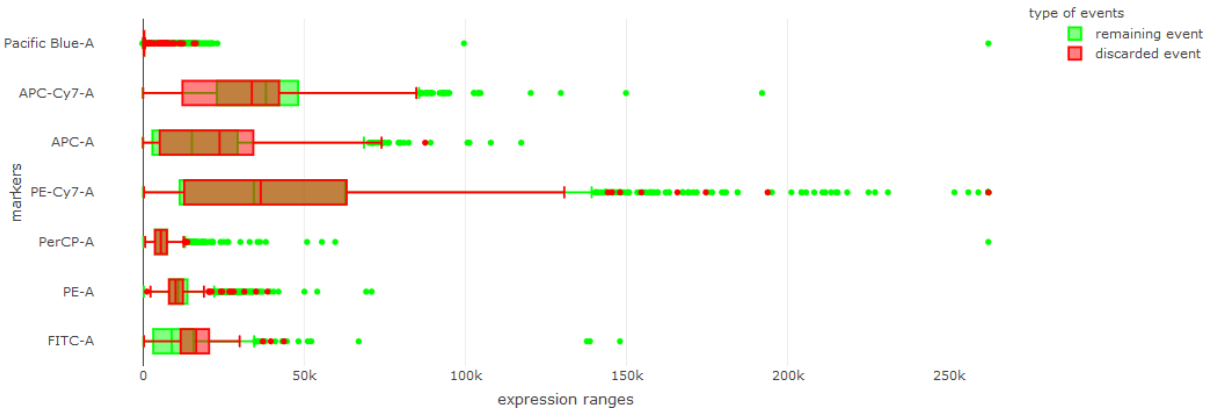


Figure 15 - the outliers after the quality check

In the on-line HTML loaded page available from a normal web-browser is interactive (thanks to the plotly package) and it can be visualized in more efficient way through the web-app. The green dots are the outliers while the red part reported the tagged events which are not discarded after the down-sampling operation.

Concerning the parameters used for the cleaning process, in general: leave them as they are (default values), but you can always tune the process especially when you deal with a lot of events and you have space to reduce them selecting the best part of your acquisition. In particular:

- alphaFR: u can lower it if too picky, especially in case of too few events (the lower, the lower is the impact in terms of event dropping). It affects the flow rate check
- pen_valuesFS: the same as alphaFR (the higher, the less events u drop). It is linked to the signal acquisition check

5.2 The Scaling (Transforming)

Since the acquisition ranges of each single marker (channel, stain…) can be very much different from each other, a range transformation has to be accomplished in order to perform any kind of visualization and comparison between these expression variables. Several different functions has been proposed in literature in order to handle the strongly skewed distributions of the expression ranges but the most used is the *biexponential* function also called *inverse hyperbolic sine* depicted in red in Figure 16.

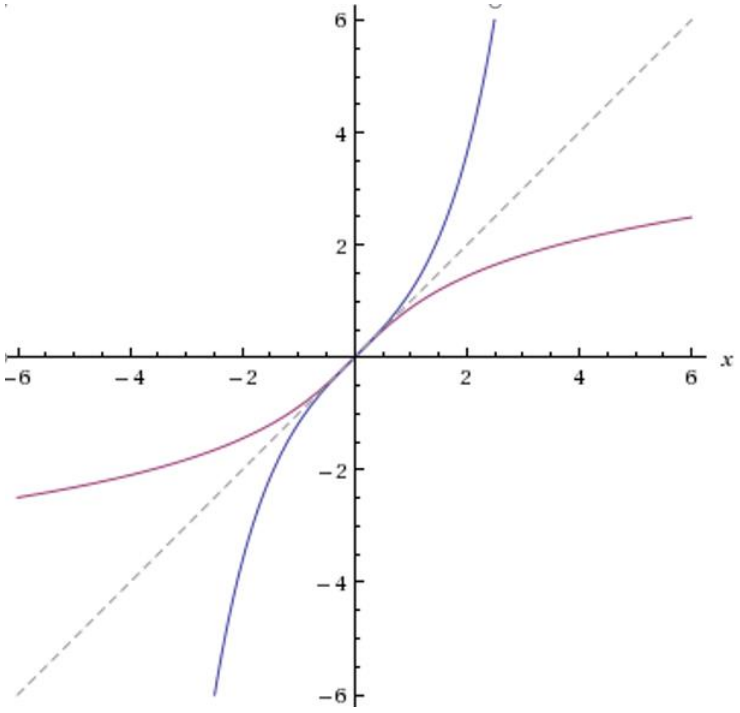


Figure 16 – The hyperbolic sin function (in blue) and its inverse function (in red)

The *arcsinh* function is the inverse of the bi-exponential function:

$$\sinh x = \frac{e^x - e^{-x}}{2}$$

Equation 1 - the sinh function

$$\sinh^{-1}(x/m) = \ln(x/m + \sqrt{(x/m)^2 + 1})$$

Equation 2 - The arcsinh used in the cytoChain

Where the *m* corresponds to the *arcsinh* parameter in the related sidebar, which is set by default to 150 as reported in (Sean C. Bendall 2011).

The *arcsinh* type of transformation is very straightforward and it has the undeniable advantage to be reversible (in fact, the arcsinh admits the inverse transformation since its domain is (-∞, +∞) but the fixed cofactor is the same for each single marker. For this reason a more flexible transformation criteria has been introduced: the *flowVS* transformation. Its name derives from the related R package (see APPENDIX) which computes for each single channel, the variance of the expression values with a method based on maximum likelihood (ML) estimation. With this every marker has its own calculated arcsinh cofactor (and not a single arbitrarily set value). The calculation could be very time consuming, especially for cases with many samples and a lot of markers. Nevertheless its performance could be for example appreciated within The flowSet analysis tab. In fact, the MDS plot could vary a lot because of the transformation choice.

The effect of this more complex scaling could add the appreciable effect of having a more clear and separated clusters as it is shown in this tSNE map

Along with this transformation another scaling is proposed in cytoChain (the simple normalization scale, the 'arcsinh+' and the Zero transformation) with the related parameters. The simple normalization scaling is based on the

$$\frac{x - min(x)}{max(x) - min(x)}$$

Equation 3 - the simple normalization scale function

While in the 'arcsinh+' function the *f(x) - min(f(x))* shifting (like in Equation 3) is performed in order to deal with the classical *arcsinh* transformation but returning only positive values (useful for certain clustering algorithms which does not allow negative values).

The "Zero" transformation is similar to simple normalization but with the possibility to assign 0 or 1 based on configurable threshold: a type of transformation described by the Equation 4 in 7.5.

Once the transformation is performed, the tool let you visualize the different marker expression for each sample in order to provide you a cross-checking of the expression trends (the Figure 17 is related to a very small panel with just few stains)

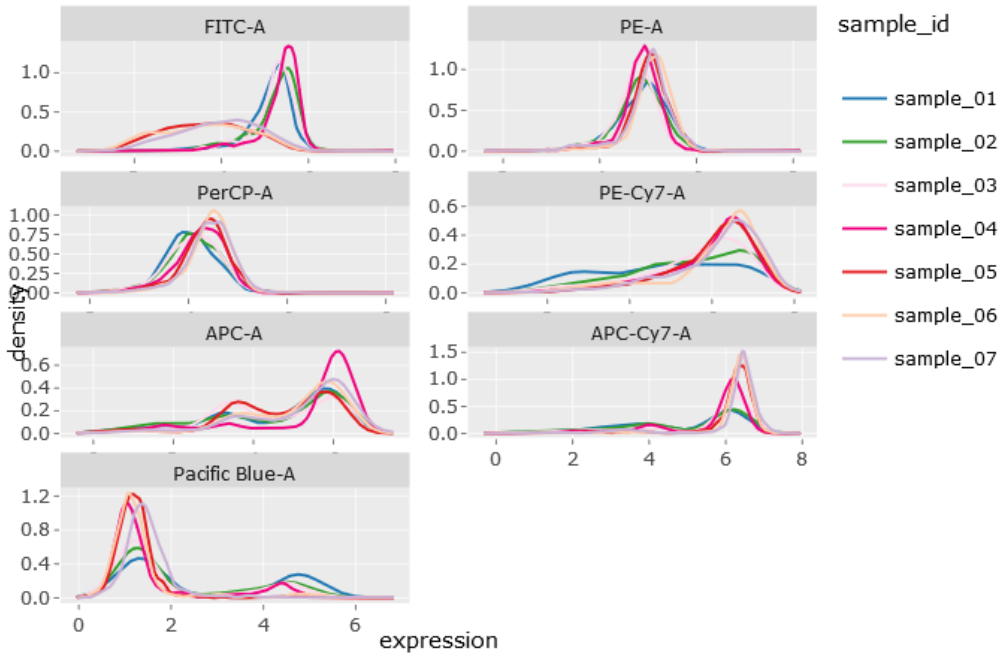


Figure 17 - The sample's marker expression

5.3 The Aligning

As you can see from the densities plots reported in Figure 15, each marker expression could vary a lot from a sample to the other. This could happen for several reasons mainly related to a natural tendency of

the fluorimeter equipment of not being perfectly stable during the different acquisition steps of the various tubes in different time, but this type of drift could also arise from the sample content itself.

For this reason the *flowStats* R package has been developed in order to re-align the marker densities along the samples to reduce these unforeseen shifts in the fluorescence intensities between samples which could affect the determination of the various biological phenotypes.

Together with the transforming operation (5.2), beware that this operation affects the expression matrix of the various flowFrames and so the whole flowSet: rather than simply filtering out some of the events, these operation change your dataset values.

This alignment process tends to adjust along all the sample set, the different population of events, in order to align them following the density distributions of the selected markers to fix this drifting phenomenon. This is performed through the *warpSet* function of the *flowStat* package. The process is based on four steps for each single selected marker:

1. Searching for the high density peaks (landmarks): each of these landmark should be related to a certain population of events
2. Number estimation of those landmarks: as it can be seen from the densities plots, some markers could have multiple peaks
3. K-means clustering: to select and classify the relevant population around the landmarks
4. Landmarks alignment estimation: to compute the best fitting coordinates to align the population over the different samples

Warning: The whole process works fine if the high density areas truly represent particular sub-types of cells. In order to align along the different samples, a certain population should be uniquely determined, which is true when the markers are binary that is the cells are either positive or negative for a particular marker (namely, the typical case of lineage markers). Avoid to align with markers which smear all along their expression! A safe and meaningful choice of the markers to perform the alignment, should be carefully selected. In particular, try to align those dimensions in which the expression of the scaled value trend, shows little differences between the sample's peaks. If the differences are unduly large, especially in those markers which have more than one peak, the alignment could be wrongly performed, trying to align peaks which are unrelated: compare the markers trends before and after the alignment to verify the results and to avoid possible misalignments.

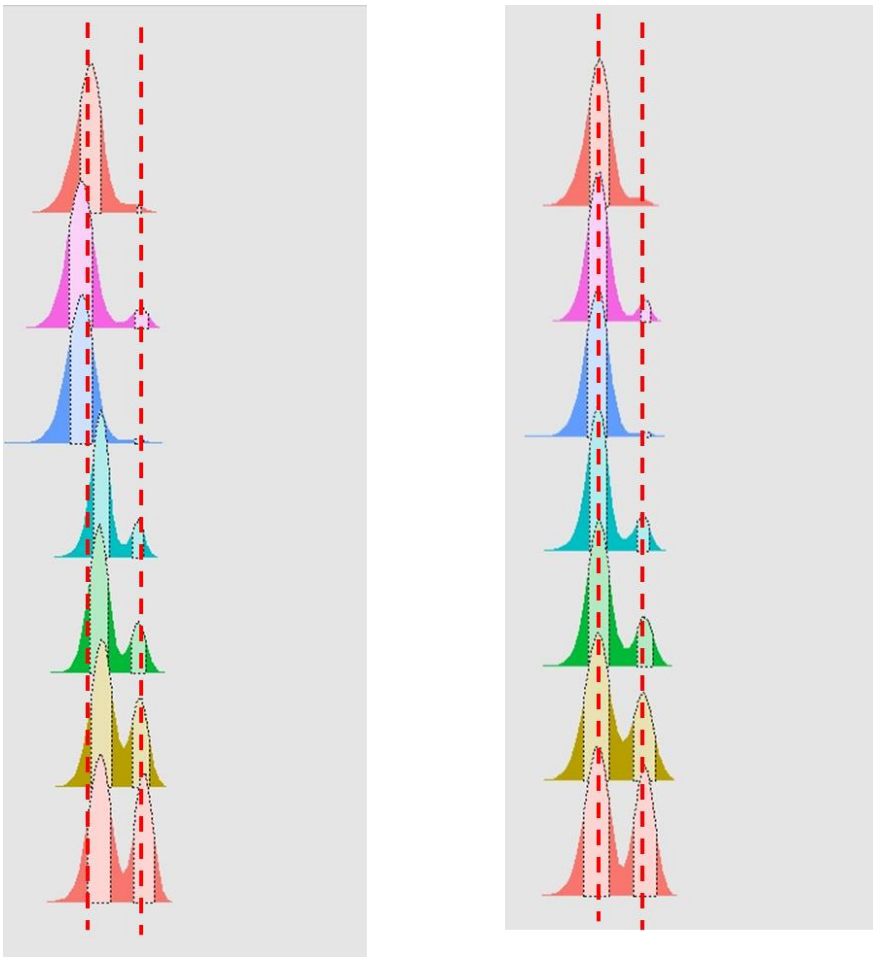


Figure 18 – Before (left) and after (right) the alignment

Finally, please also notice the sensitive nature of this type of process: together with the scaling, this is the data transformation which fundamentally alters the data input and indeed the experiment values. But while scaling is mandatory to produce something visible and comparable in scale, performing an easy and reversible function transformation, this alignment process perform a complex data manipulation, so please the advice is 'handle the alignment with care' motivating your choice. The best would be to perform the complete analysis with and without the alignment to cross-check whether there is a significant improvement in the maps production with clearer clusters and phenotypes.

This part of the cytoChain web-app is still under deployment.

5.4 The Down-sampling (the outliers filtering)

The main aim of the down-sample operation is to reduce the number of events. This could leverage the calculation effort for the cluster analysis (see 7) and the resources needed in time of memory and processing time. Nevertheless our experience, dealing with different experiments, clearly shows better fitness of a properly down-sampled dataset for the subsequent clustering analysis.

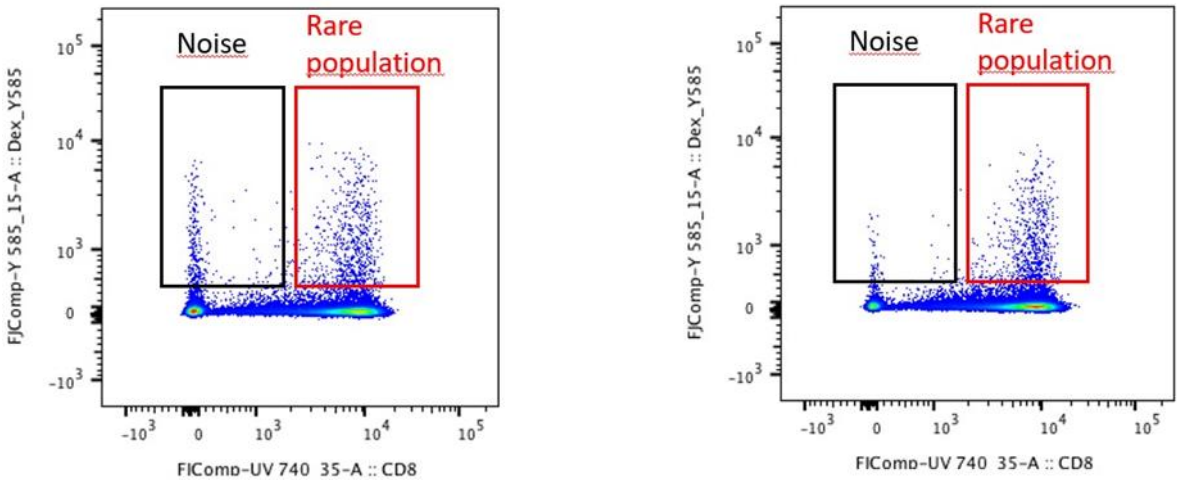


Figure 19 – Events distribution before and after downsampling

As you can see in the figures above, the real aim is to reduce the density variation across the cloud reducing noise and giving weight to small, less dense groups of cells so they won't get absorbed into the larger, denser regions as shown in the following figures. The effect is to provide a better and sharper event distribution also when clustering, namely when putting all together the same type of events in different homogeneous groups.

The overall effect is better depicted with the concatenated flowSet (the whole event bunch for every sample collected - aka *concatenated* - in a single flowFrame – see 5.5) shown in a dimensionality reduction plot.

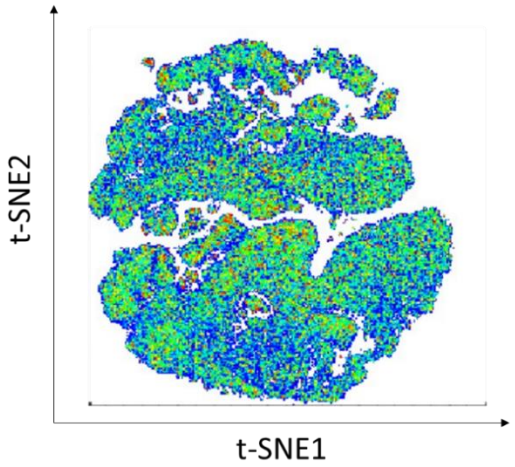


Figure 20 - t-SNE plot before flowSet optimization operations

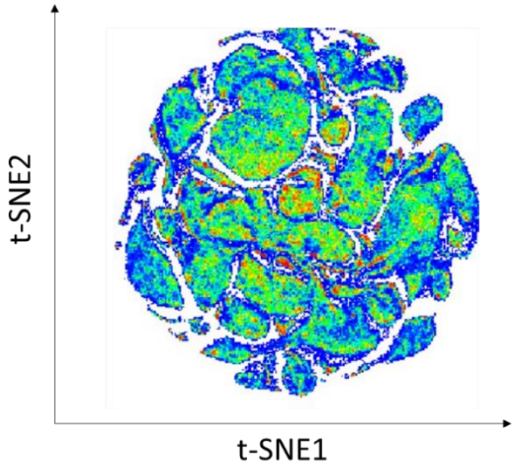


Figure 21 – t-SNE plot after flowSet optimization

The islands, which collects homogenous groups of events, are more visible in Figure 19 (with the tSNE plot having well sharp borders) than in Figure 18 in which a kind of "fog" confuses the homogenous regions and in which only the main islands are well separated. The figures are taken from a gating view from the well-known *flowJo*[8] tool. Notice that well performed pre-clustering operations and especially the down-sampling could increase and better defines the number of clusters.

In this sense the down-sampling is more an outliers removal operation rather than pure random event reduction. The already mentioned (see 8) tools also implements some kind of down-sampling but to remove possible outliers means to perform an analysis of the working dataset and to assign a type of score to compare each single event with the other.

The Figure 18 and the Figure 19 report plots related to the down-sampling algorithms evaluation. They are related to the original flowSet, before the down-sampling process (the tools also reports the same plots after the down-sampling in order to be able to compare the effect of the operation).

Another important checkpoint for the advantage of a good pre-clustering treatment is illustrated in The flowSet evaluation tab when the *kmeans* cluster projection could show its benefit (see in particular Figure 28 - the elbow method with k-means in different flowSet).

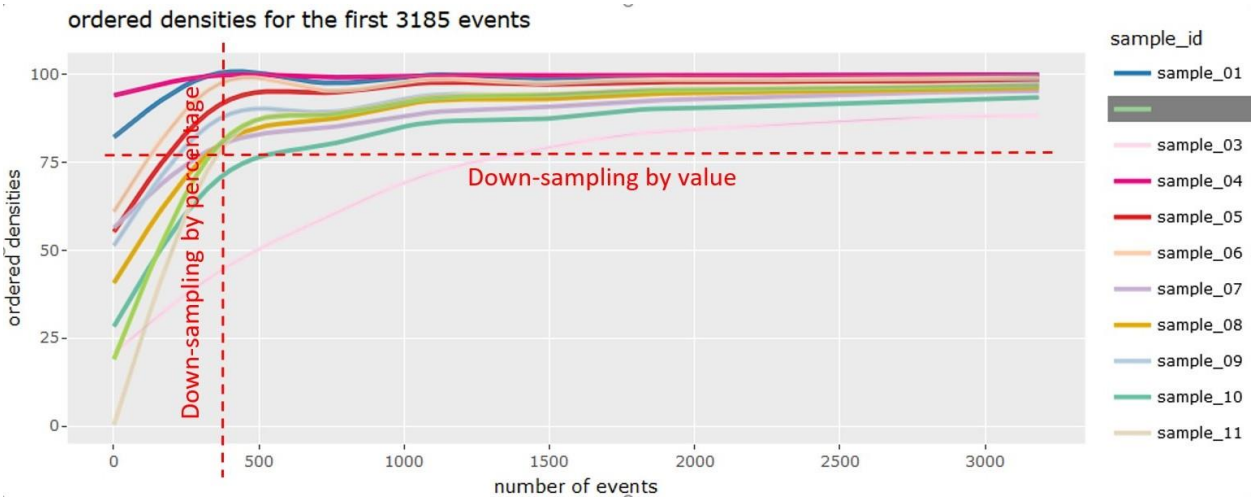


Figure 22 - The ordered score plot

The Figure 20 shows the outliers score for each of the samples for one of the available algorithm[9]: the *spade*. Each down-sampling algorithm assigns a score to every single event: in general, the higher is the score, the higher is the probability that the event is isolated from the rest, namely there is a good chance

[8] Along with the FCS Express, the flowJo application is currently the most widely used tool in flow cytometry for manual gating

[9] The current implementation collects some of the algorithms collected by the *DDoutlier* package, namely the KNN_AGG (Aggregated k-nearest neighbors distance over different k's), the KNN_SUM (Sum of distance to k-nearest neighbors), the LDF (Local Density Factor (LDF) algorithm with gaussian kernel), the LOF (Local Outlier Factor), the LOOP (Local Outlier Probability) and the RKOF (Robust Kernel-based Outlier Factor algorithm with Gaussian kernel) and the additional *SPADE.addDensityToFCS()* function provided by the spade package.

that the related event is an outlier: the greater the score, the greater the outlierness. With this, the way to evaluate this score/densities values assignment is to order them starting from the higher scores (or the lower densities).

Notice than the ordered scores/densities plot restricts the view taking into consideration the amount of events of the smallest sample. That is to better focus on the most interesting part of the trend: the highest score (or the lowest densities) for every single sample.

While the ordered scores/densities plot is mostly useful to find a possible valuable cut-off point to set your percentage amount or the cut-off value, the normalized scores/densities plot is just to have a global view and to compare the trends of the scores/densities calculated by the chosen down-sampling algorithm for every single sample. Since all the values are normalized both in terms of scores - on the abscissa - and in terms of density amount - on the ordinate - the graph allows a comparison of the score trends of the for each single sample. Take this into account when you deal with experiment with samples collecting a very different number of events.

You can filter the outliers in three different ways and the first two are: by percentage and by value. As a rule of thumb, if you want to get rid of the outliers but your goal is to search for some specific rare type events, you better perform down-sampling for a very little percentage of events. If this is not the case but you want to get rid of the outliers only, it is better to set a value threshold for your down-sampling: in this case you may cut a different percentage of events for each sample. The last possible choice could be to equalize the number of events: with this you probably cut out the outliers but also a lot of significant events, but you may achieve to get the same amount of events among the samples. Let's see in details these three types of down-sampling.

5.4.1 Down-sampling by percentage

To better explain the thresholds choice, in Figure 20 two dashed lines are reported: the vertical one is to show the cut-off for the percentage selection.

When you select by percentage, the *exclude_pctile* is the threshold to filter out the events (e.g: with a threshold of 3% you filter out the 3% of the events with the higher score). In this case every single sample will be reduced by an amount which is related to its total number of events.

The best way to set properly the percentage threshold is to carefully study the score plot (see Figure 20) and to compute the following ratio

$$\frac{number\ of\ event\ after\ the\ elbow\ shown\ in\ the\ score\ plot}{number\ of\ event\ shown\ in\ the\ score\ plot}$$

This will give you a meaningful percentage of what you could cut, removing the outliers. Keep in mind that the amount of events shown in the score plot is just the number of event of the smallest sample (and not the total amount). This is a conservative approach, in order to not affect the smaller samples. With this you could also exceed this ratio, but at risk you could remove events from the smaller samples which actually are not considered outliers. For this reason we introduce also the following cutting criteria, which is useful especially when you have a flowSet with very different amount of events along the samples.

5.4.2 Down-sampling by value

When you choose by value you cut out those events which are lower than the threshold value. Of course in this case you are surely selecting only those events above the threshold but without knowing exactly their amount (the advice is to perform iteratively the selected algorithm with different thresholds.

If the goal is just to remove the outliers, the best would be to not go over the point where the score trend reach the changing peak: after that peak the slope and the score trend remains nearly constant, meaning that the related events are in the bulk and not "isolated" from the rest. The horizontal dashed line shows a possible threshold which will cut-off mostly the events in some samples only. Typically this could be handy when, like in this case, the two samples over the thresholds are the smaller ones (the ones with less number of events). With this you kill two birds with one stone: you down-sample the outliers without affecting the smaller samples and so harmonizing the event quantity between the samples.

In any case, keep in mind that the score values expressed in the plot are normalized (they are not the ones directly calculated by the selected algorithm - this to be able to compare values on different samples). There are some exceptions. The first is obvious: a Random algorithm does not compute any score: it is possible to filter randomly only by percentage and not by value. The other exception is the spade algorithm (see the APPENDIX) that, as a score, it computes the event density: in this case the lower is the density the higher is the outlierness as depicted in Figure 20.

The last exception is related to the LDE algorithm. The greater is the Local Density Estimate score, the greater the centrality. This last algorithm is used to downsample the most dense zone of the expression hyperspace and it can be used to perform a downsampling which allows the reduction the total amount of events. This can be useful for very big samples set, to reduce the amount of resources needed for the clustering analysis. The drawback is that you do not filter out the outliers but those part of events which are in the most dense zone of the hyperspace. Usually these are the most representative events, but in case you would like to focus on the rare population, downsampling the denser zones means to focus more on those events which are more isolated, namely by selectively favouring certain events, the quantities are distorted.

If you can find any elbow, or a marked discontinuity along the ordered scores (or densities) plot, this could be a point of considerable interest to set your cut-off percentage for your down-sampling, especially if your goal is to remove the number of outliers, rather than simply down-sample the amount of events. In this case, just compute the related percentage (recall that the ordered scores/densities plot focus on the first number of events, taking as a reference the smallest sample) to set your cut-off percentage. The same applies to a cut-off criteria based on values instead of percentage. In this case you set the value around the elbow point as reported by the order scores/densities. The only drawback is that you cannot infer a priori the amount of events you would filter out: iterate you down-sampling process until you find satisfying result.

Not every algorithm is able to provide a sharp discontinuity, namely a clearly shaped elbow in order to be capable of guide you through this analysis: the big question "*which are the outliers in my dataset?"* is one of the most complex and controversial you can find in statistical literature because (like for the clustering algorithm) the results strongly depends of the data to analyze, its complexity and even its magnitude in terms of number of element, namely an algorithm that suit for every occasion does not exist. For this reason cytoChain provide a list of algorithm to choose from.

If you do not notice any relevant difference between the plots reporting the trend before and after the down-sampling process, this means that you do not cut a sensible amount of events, which could be perfect choice if you just do not want to strip-off a considerable amount of events to preserve the original relative percentage when you proceed with the analysis. The problem is that, most of times you have to consider the down-sampling process to simplify and improve your dataset in order to safely perform any clustering process. More properly, to get rid of the outliers could be a must in order to deal with a qualitative treatable dataset. A straightforward way to evaluate this treatability and the related rendering process ability is to go through The Metadata & Assays Workflow and in particular, to select the flowSet evaluation tab. The first analysis there, is to produce the scree plot of the eigenvalues from the PCA algorithm: this is the easiest way to evaluate the sparsity and the complexity of your dataset. A treatable flowSet is the one having the two principal components capable of handling at least the 70% of the relevant information. Again, this is not apodictic but it provides you a rule of thumb to know what is what and most of all, a quantitative index to compare different dataset.

5.4.3 Down-sampling by equalize

A common problem, in this type of analysis, is to keep the number of events among the samples more or less the same in order to quantitatively compare the percentages of the various clusters or phenotypes.

Furthermore, the analyst should take into account the common amount of events among several groups of samples when these are considered of the same type when you categorize them within the metadata definition (see 6 on the metadata). This is the only way to compare them and to have some significant values in your results.

Of course, if your goal is to qualitatively compare the various samples, or to find some specific phenotype, this is not relevant, but if you want to obtain some numbers, it is difficult to get some meaningful quantities if the number of events among samples are very different. For this reason we introduced another type of down-sample filtering: the equalization. Selecting “equalize” you can filter out all the events which exceed a fixed number which can be set: with this, the outcome could be a whole set of samples with exactly the same number of events.

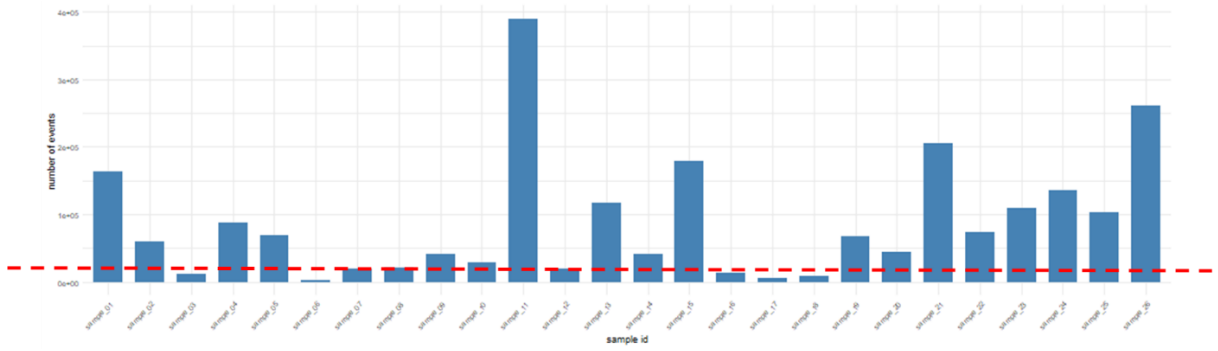


Figure 23 – A typical case of uneven distribution among the sample’s events

In Figure 21 you can see a typical case of uneven distribution among the samples. This is very frequent especially when the samples are biological replicates from different treatments or tissues and in which the amount of meaningful events could spread from some hundreds to millions.

Of course, the operation here is not just a down-sampling: if, for some of the samples, there is a lot of events to discard, you may get rid of cells which are not outliers since they belongs to the bulk of the events in the hyperspace and so could be among the most significant ones. This could happen despite of the down-sampling algorithm selected, but this is the price being paid for getting the samples totally equalized. It must be noticed that, in order to avoid any kind of biased outcome, **we strongly recommend to select the *Random* algorithm** with the *Equalize* type of down-sampling: select the amount of event you want to preserve, download your new samples and then, if you would like to remove the outliers, just perform a new down-sample by percentage or by value with your preferred algorithm. This is also useful when the total amount of events exceeds of 1 million. In this case the system could run out of memory and to avoid this, the best is to select the down-sampling in the *equalize* way as a first step in order to reduce the overall sample's weight.

A final advice. More than often several machines are capable of producing tons of event per sample (even more than 1 million!) and adding together the all bunch of samples in your experiments you end up with an unmanageable data set. The recommendation is to not to exceed ≈ 1.5 million of events. For this reason this random down-sampling comes in your help to flat your samples and make there size more reasonable. For this reason, you may try to perform this even before the cleaning in order to overcome computational problems

5.5 The Concatenating

Concatenation, in the cytometry analysis jargon, consists of putting the entire flowSet into a single flowFrame collecting every single event. Within a concatenated flowFrame every single event is tagged with a Sample_id value[10]. Since this type of operation is the cornerstone for every subsequent cluster analysis (see The High Dimensional Analysis Workflow) it is implicitly performed in any case. The concatenating process tab is implemented in cytoChain only for checking reason and to let you export the related *.fcs* file in case you want to analyze the handled concatenated flowSet in other environments.

As a matter of fact the concatenating, with the related expression matrix extension, allows the comparison between events when they are properly mapped within a dimensionality reduction representation (whatever type of mapping algorithm used, be it the tSNE, the UMAP in two or three dimensions – see The High Dimensional Analysis Workflow). This could be extended by any other type of dataset characteristic derived from the sample_id itself (like the *patient_id*, the *time_step* or the *condition* metadata as introduced within the 6 - The Metadata & Assays Workflow) or by other analysis process like the *clustering_id*, the *meta_clustering_id* or the mapping coordinates chosen. In the current implementation the complete concatenated flowSet with the complete set of additional channels embedded in the expression matrix can be downloaded within the mapping tab of The High Dimensional Analysis Workflow.

[10] The expression matrix of the concatenated flowFrame collecting all the events is extended with an additional column: the Sample_Id channel in this case.

5.6 Some hints on the pre-clustering workflow

A few hints in order to deal with the flowSet optimization workflow and in general with the rest of the workflows:

- Recall that the workflows are pipelines with no "tap in the middle": it's a cytoChain. So then, the latest action is the one inherited on the following workflow and saving flowSet also inherits the action pipe.
- In general, feel free to start where you want to start and the same applies on every workflow and on the workflow's chain (do not forget to re-load the table on the 2nd workflow). To start directly from the second workflow is a good tip especially for the bigger flowSet (save time and memory). In fact, uploading and parsing your samples from the first workflow, let you visualize all details about the flowSet before to use it in the second workflow to edit the various meta-tables. Just save (download) your handled flowSet first
- Be patient, please wait (some calculation really need a long time, even tens of minutes). Check the console window on the top left. keep calm & let it computes
- Wait for the full execution of the last action and wait also for the reports (some pictures are huge or heavy to transmit)

5.6.1 Saving the handled flowSet

Concerning the download of the handled flowSet, please notice that two possible alternatives are available:

1. Download your handled flowSet
2. Download your handled flowSet with both the handled expression matrix and the original one

If you want to save your flowSet to be further handled by cytoChain in the further processing steps, please consider the first option. The second flowSet in fact, collects the expression matrix with original values + the handled values (the named marker will be with "< >" sign): you will get two 2 times the markers and a two times bigger flowSet. This is to save your original data set along with the handled one (complete mode), useful in case of publication where all the original experiment material should be available.

Please notice that the original expressions will be saved in complete mode with the original dimensions names enclosed in **< >** brackets.

5.6.2 On the sample's naming

On flowSet uploading, the samples are loaded as you select them (ordered by name, or by size, or by date…) on zip format uploading (single file – the files are always ordered by name). In this case the best advice: always order by name (namely alphabetically) because it is important for further workflows and to assign the simplified sample_xx as expected (see 6).

Notice that the flowFrame handling during the pre-analysis phase add some meaningful prefixes (history description) on your sample's name.

E.g.: for a sample with name: ***170720_tumor_001*** the name assigned could be:

DKNN_AGG_value21.8A_T_arcsinh_C_p1_5_p2_40_Id_01_170718_tumor_001

Where for each handling process you will have an additional string concatenated form left to write in the proper order. So for this example cytoChain adds:

Id_01 – the uploading and parsing action always add an Id

C_p1_5_p2_40 – the cleaning action with the relevant two parameters (see 5.1)

T_arcsinh – the scaling action with the transformation type (see 5.2)

A – the alignment action (no parameters – see 5.3)

DKNN_AGG_value21.8 – the down-sampling action with the algorithm used and the related parameter (see 5.4)

5.6.3 On the sample's events and quantities

The overall high dimensional analysis process is statistical: in this perspective, the more, the better. In general, you are dealing with some heavy calculation and not always the amount of resources could be sufficient to face the various part of the process. On the other end, a typical event distribution could be the one depicted in Figure 21, where there is a definite imbalances between the samples. In this case a proper down-sampling to equalize is necessary in order to deal with the statistic to compare in the best way the population in the various samples. In this sense the question is: how much do I cut? What is the meaningful amount of events per samples to deal also with rare populations of events?

If you roughly know the rare-events percentage a rule of thumb could derive on the minimum amount of events cytoChain could distinguish.

If cytoChain can find ν events on a small cluster, the total amount of events we need to find this rare group of Π (in percentage) should be:

$$\mathrm{N} = \frac{\nu}{\Pi}$$

E.g. if Π = 0.01% and ν = 10 → N = 100000 total events

The value of ν defines the sensitivity of the clustering algorithm and a value 10÷20 could be a reasonable assumption. Of course you should have a value for Π which could be inferred by a manual gating on some known population. In general an amount of 2000÷5000 events per sample should be fine for most of the analysis. Some guidelines may result also from the following flowchart:

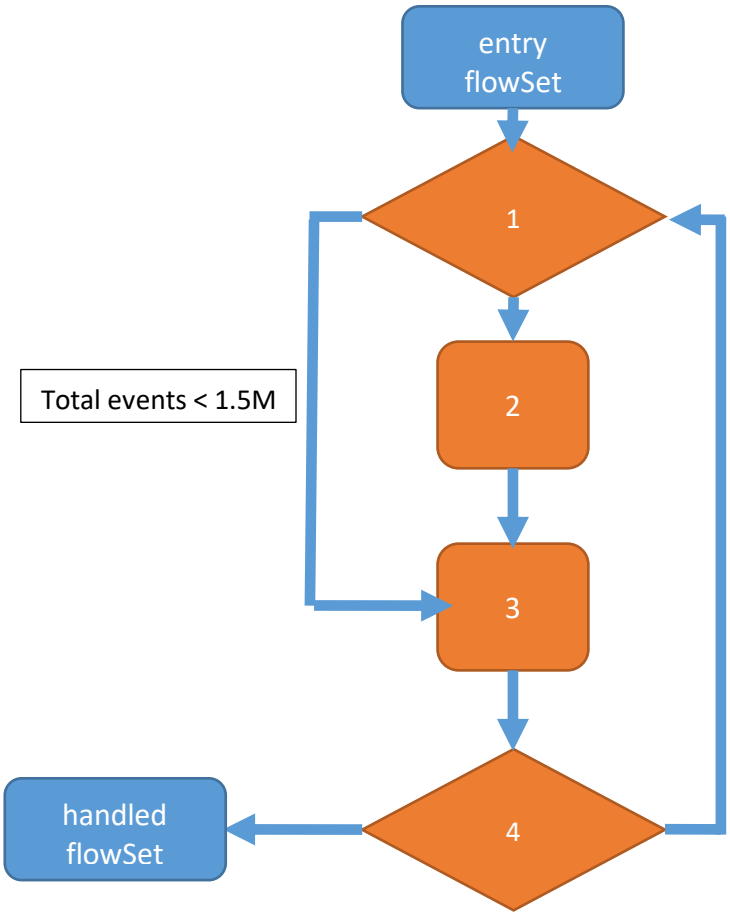


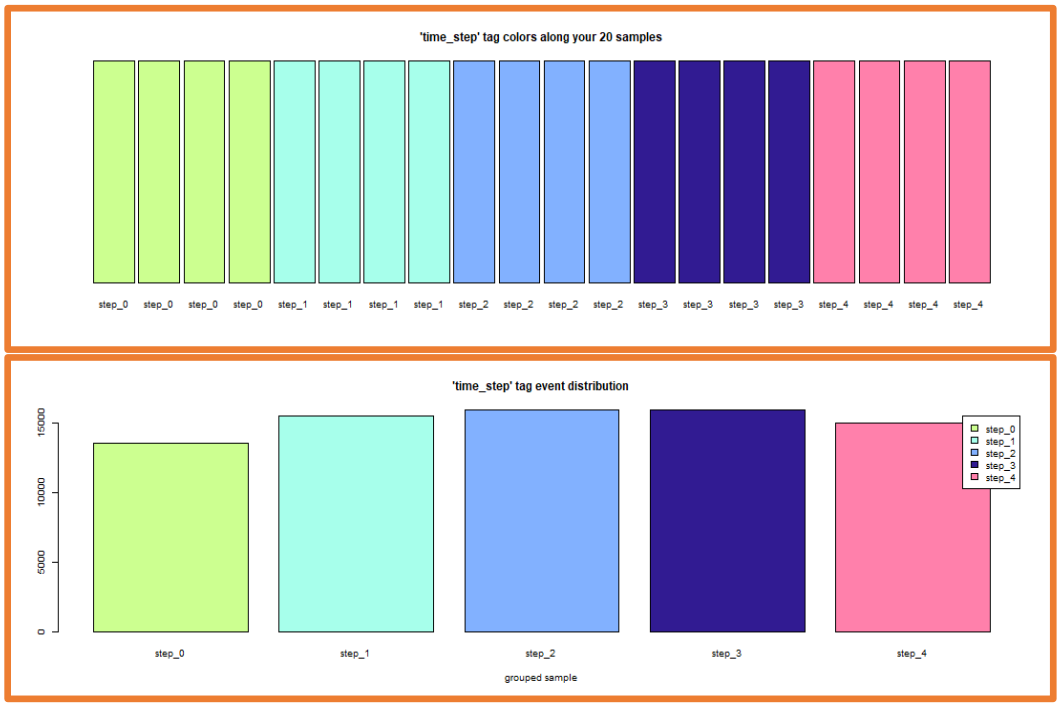
Figure 24 - a flowchart for the sample preparation

Where the steps are:

1. In case of total number of events > 1.5M, try to equalize your sample setting a common value (2000÷5000 events per sample) that fit the required amount of event + some spare amount (+ 10%÷20%) and save your equalized samples. In case of total amount < 1.5M go directly on step 3
2. upload your saved equalized samples
3. Produce your new cleaned & down-sampled flowSet (if you have a lot of spare events, clean more punctiliously to select the most stable acquisition parts). it is always better to clean the original – try to move the equalization at step 1 if there is no memory problem
4. Check if the new flowSet collects a sufficient amount of events: if yes, proceed with the further workflows; if not, re-iterate until the total number of events reach the expected amount (act on cleaning and down-sampling result)

From the consideration above, a recommendation should be taken: try to remains under the amount of 1 million of events even with a considerable number of samples.

The equalization should also take into account the various samples characterization with the various tag (see chapter 6 for the metadata).



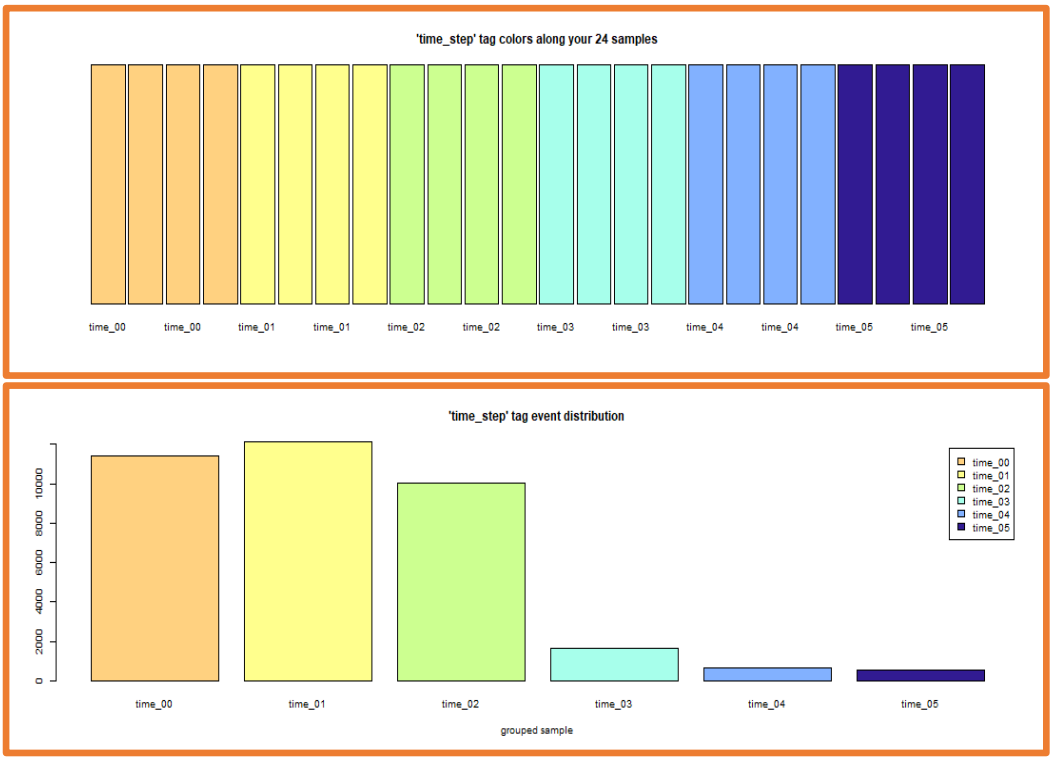


Figure 25 - event distribution on a time-step characterization with 6 steps and a total of 24 samples

The two event distributions shown in the bar plots above clearly depicts two different distribution: on the first one the event distribution is almost flat: the result could be meaningful in term of quantities as well; while the second bar plot shows an extremely unbalanced distribution of event between event in the various steps: the results could be only of qualitative type.

These are the type of figures you obtain when dealing with the meta-data tables (see 6).

To recap, the pre-analysis actions of the first workflows are listed and compared in the following Table 1

action	**impact on the high dimensional analysis**	**acting on amount of events**	**acting on expression matrix's values**
cleaning	very high	YES	NO
scaling	depends on parameter setting	NO	YES
aligning	depends on the dis-alignment on the original flowSet	NO	YES
down-sampling	high	YES	NO

Table 1 - A comparison of the flowSet optimization workflow actions

A last recommendation: since its importance, try to perform the cleaning before any kind of down-sampling: cleaning algorithm performs better on original samples with no holes in time dimension.

6 The Metadata & Assays Workflow

The first part of the second workflow (see Figure 9) is essentially to set additional variables to combine with the *sample_id*. In the subsequent analysis these new entries could help in distinguish the various population types and to compare the different sub-population based on these additional parameters. With this is possible to enter some advanced characteristic of the sample and to associate them to the subsequent cluster analysis. You can appreciate the benefit of these meta-data entries in the plots generated of the clustering workflow.

The mechanism works essentially as depicted in The Concatenating process: every metadata is mapped as an extension of the expression matrix with the additional related columns added exactly with this purpose. In the subsequent analysis these new entries could help in distinguish the various population types and to compare the different sub-population based on these additional parameters. The scope of the rest of the tabs (see the ***flowSet analysis***, the ***flowSet evaluation*** and the ***map evaluation***) is to provide some high level information about the samples, without entering into the cell's details.

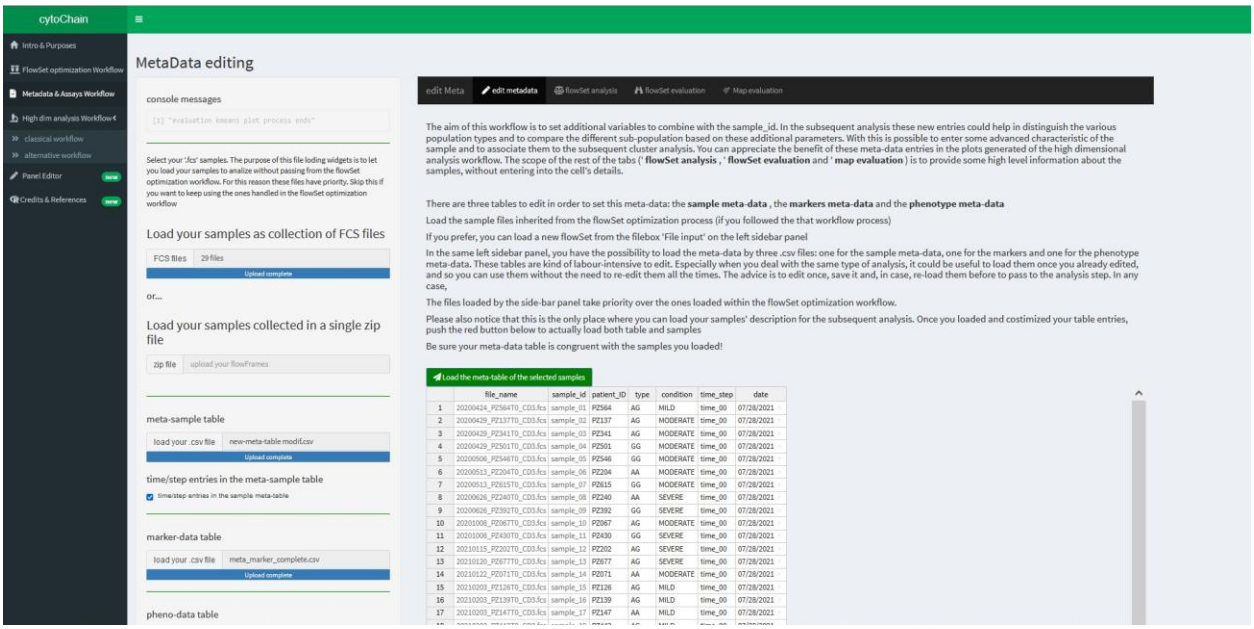


Figure 26 - The Metadata interface as it appears from a common browser

Edit Metadata is a totally separated workflow, a best practice could be to save the latest version of the data set you produced within the flowSet optimization workflow. In this case you can start directly within this workflow saving resources on the server (for example, the server will need a smaller amount of memory). You simply load your sample set within the “Load your samples as collection of FCS files” widget. Otherwise you can stick with your last sample production in the flowSet Optimization workflow and simply press the “load the meta-table of the selected samples” green button. If no meta-sample table is ready, a

"dummy" meta-data table will be produced for you. Just handle this table if you want to distinguish your samples, characterizing them tag by tag reflecting your experiment design.

6.1 The Edit Metadata tab

This section is devoted to the editing of the metadata. There are three tables to edit in order to set this tables:

- **The sample meta-data**: The first table collects the sample's characteristics to check sample type differentiations on the common concatenated flowFrames. It is mandatory for the rest of the flowSet analysis. If you do not have any particular differentiation among your samples, you can just use the automatic editing and leave the table as it is or you can load a table with the proper format by file selection widgets in the sidebar
- **The markers meta-data**: The second table is to select the markers to be used for the analysis. Select only the relevant channels. It is mandatory to select at least two markers: the rest of the analysis and The High Dimensional Analysis Workflow will be based on this selection
- **The phenotype meta-data**: The third table is the only optional on. To fill this table or not depends on the type of analysis approach you want to follow: without this you can only perform the workflow type 1 analysis

You can load the sample files (*.fcs*) inherited from the pre-clustering process (if you followed the flowSet optimization workflow process) or, if you prefer, you can load a new flowSet from the filebox 'File input' on the left sidebar panel). This choice would facilitate the sample selection you already saved from a previous session without passing by The flowSet Optimization Workflow.

In the same left sidebar panel, you have the possibility to load three comma separated values (*.csv*) files: one for the sample, one for the markers and one for the phenotype meta-data. These tables are kind of labour-intensive to edit. Especially when you deal with the same type of analysis, it could be useful to load them once you already edited, and so you can use them without the need to re-edit them all the times. The advice is to edit once, save it and, in case, re-load them before to pass to the analysis step[11].

6.1.1 The sample meta-data table

The sample meta-data table in the **edit metadata** tag of the **Metadata and assays workflow** collects the sample's characteristics to check sample type differentiations on the common concatenated flowFrames. Currently the classification is based on up to 3+1 = 4 (3 tags + time phase) characteristics. To handle more characteristics, please split the analysis by loading further tables with the rest of tags for the same flowSet.

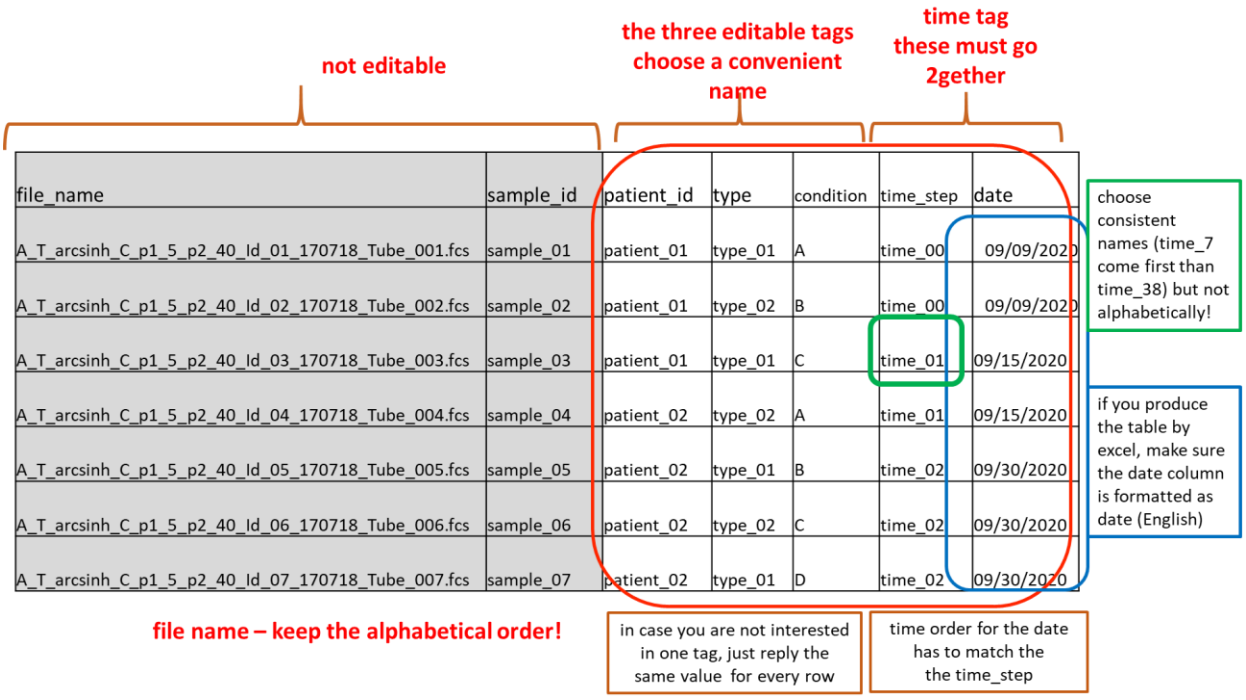
[11] Warning! The samples should be edited in time order: the oldest dates must come first and selected as 'time_00'. Since each single sample is loaded holding the order in your directory, the best would be to name your files according to the order of the various steps. On the other hand, if you are not interested in the time levels, you can skip this, holding the list automatically entered.

Please, avoid special characters (even the "+" or "-"). This to avoid the risk of data mismatch within the cytoChain algorithm's process.

In order to be sure to have the correct format, one possibility is to modify table on-line (always download it when you have finished, to store a version of your meta-data analysis). This on-line mode could be too more laborious and error prone, especially for big flowSet and the related table with tens of rows. A further way is to download the automatically produced table, save it and manipulate this one. The automatically produced table is obtained by pressing the **Load the meta-table of the selected samples** green button. This generated table version assigns one different value for each sample and for each tag.

Once correctly handled upload it under the same tab, check if the table is correctly loaded before proceed (to be sure check the console message)

The following Table 2 is one of the entry table in which a user could edit the three yellow columns, namely, the *condition*, the *patient_id* and *time_step*[12].



file_name	sample_id	patient_id	type	condition	time_step	date
A_T_arcsinh_C_p1_5_p2_40_Id_01_170718_Tube_001.fcs	sample_01	patient_01	type_01	A	time_00	09/09/2020
A_T_arcsinh_C_p1_5_p2_40_Id_02_170718_Tube_002.fcs	sample_02	patient_01	type_02	B	time_00	09/09/2020
A_T_arcsinh_C_p1_5_p2_40_Id_03_170718_Tube_003.fcs	sample_03	patient_01	type_01	C	time_01	09/15/2020
A_T_arcsinh_C_p1_5_p2_40_Id_04_170718_Tube_004.fcs	sample_04	patient_02	type_02	A	time_01	09/15/2020
A_T_arcsinh_C_p1_5_p2_40_Id_05_170718_Tube_005.fcs	sample_05	patient_02	type_01	B	time_02	09/30/2020
A_T_arcsinh_C_p1_5_p2_40_Id_06_170718_Tube_006.fcs	sample_06	patient_02	type_02	C	time_02	09/30/2020
A_T_arcsinh_C_p1_5_p2_40_Id_07_170718_Tube_007.fcs	sample_07	patient_02	type_01	D	time_02	09/30/2020

Table 2 - The dataset detailed table

With this is possible to enter some advanced characteristic of the sample and to associate them to the subsequent cluster analysis. In this case the dataset is composed of just 7 samples related two patients in three different condition (A, B and C) with 2 steps in time. Along with the sample_id, these additional

[12] The date is linked to the time_step: every time step should have the same date from the step_00 (with the date being the origin of the time for the experiment) to the subsequent steps to mark the evolution of the experiment along the time.

characteristics will be integrated as an additional dimension in the expression matrix and saved inside each the concatenated flowFrame collecting all the flowFrames. The strings “condition”, “type”, “patient_id” are just tags you can change. In fact, it is possible to download the meta-sample in Table 2 and edit the related .csv file. Feel free to change the entries choosing the best and most significant tag strings to characterize your samples.

The last tag is the “time_step” which must be consistent with the “date”. With this you can analyze the evolution of the selected samples. Keep in mind that the “time_step” tag (together with the date) should be kept in consecutive order: organize your sample list to reflect this order. The advice here is to save your file names alphabetically ordered in order to not to mess up with the different tags.

It is also possible to skip the time related entries in case you do not have a typical step by step experiment set: simply un-check the time/step entries in the sample meta-table widget inside the parameter mask of the MetaData editing page. With this it will be possible to deal with all 4 tags, having no time step characteristic at all.

6.1.2 The marker meta-data table

The next metadata to edit is for the selection of the markers to be considered in the various analysis (Table 3). As a matter of fact, even if the entire panel may include even more than 30 different markers (aka stains), it is certainly not recommended to use more than 10÷15 elements at least as first selection[13]. In general it is not advisable (at least on an initial stages of the analysis) to select more than 20 markers

[13] Just think about trying to distinguish more than 20 colors in a tSNE map or in any other quantification plot… In any case there are no theoretical limit of the number of stains to be used for the analysis, as well as there are no limit on the amount of samples to be loaded.

especially to find a specific population with a certain phenotype. For this reason it is mandatory to select a subset of markers.

	Id	dimension_name	marker_description	selected
1	1	FJComp-B-530_30-A	CD57	☐
2	2	FJComp-B-670_30-A	2B4	☐
3	3	FJComp-B-710_50-A	SIRPgammaT	☐
4	4	FJComp-R-670_30-A	TCRgammadelta	☐
5	5	FJComp-R-730_45-A	LAG3	☐
6	6	FJComp-R-780_60-A	CD45RA	☐
7	7	FJComp-UV-379_28-A	HLA-DR	☐
8	8	FJComp-UV-515_30-A	CD4	☐
9	9	FJComp-UV-580_20-A	CD25	☐
10	10	FJComp-UV-670_25-A	TIM3	☐
11	11	FJComp-UV-735_30-A	CD39	☐
12	12	FJComp-UV-810_40-A	CD62L	☐
13	13	FJComp-V-450_50-A	TIGIT	☐
14	14	FJComp-V-525_50-A	PD1	☐
15	15	FJComp-V-605_40-A	CD3	☐
16	16	FJComp-V-677_20-A	CD8	☐
17	17	FJComp-V-710_50-A	GITR	☐
18	18	FJComp-V-750_30-A	CD27	☐
19	19	FJComp-V-780_60-A	KLRG1	☐
20	20	FJComp-YG-586_15-A	Vbeta13	☐
21	21	FJComp-YG-610_20-A	Viability	☐
22	22	FJComp-YG-710_50-A	CD28	☐
23	23	FJComp-YG-780_60-A	CD95	☐

Table 3 - Marker selection

Always select the relevant markers for a consistent analysis. Typically you will deal with panels with more stains inherited from the original samples. Sometime you already gated the samples with the relevant channels. E.g. you already filter out the death cells (the Viability marker in Table 3 - Marker selection) and the non-CD3 (aka the CD3- events, to work on the lymphocytes events only) as a preliminary step in the pre-gating (see 4.1). In this case, please avoid to select the related markers because all the events will be positive for those marker and, most of all, the related expression value within the expression matrix could jeopardize the analysis results since their values are not significant for the selected flowSet.

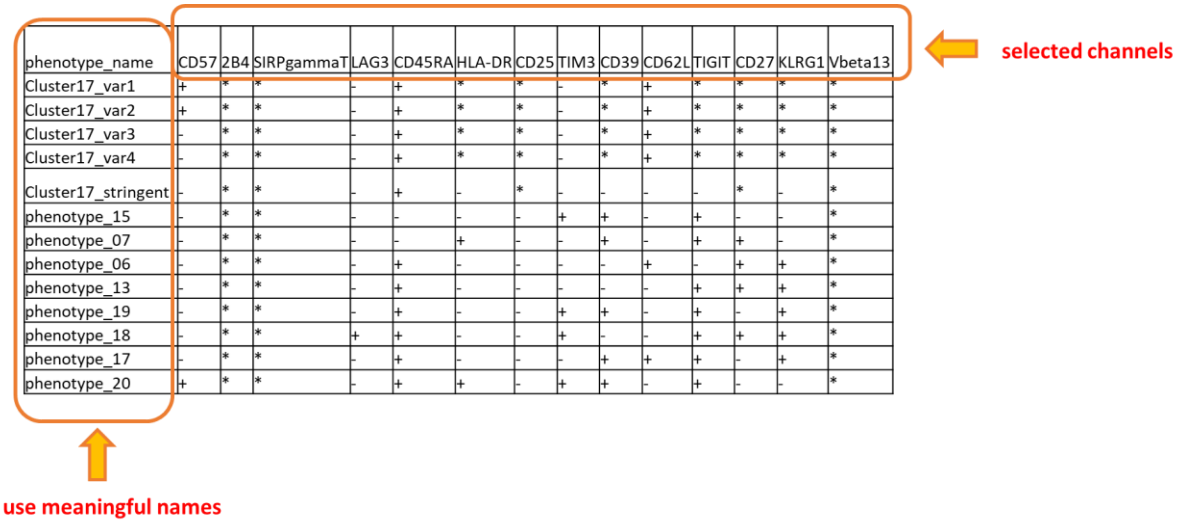
In terms of analysis strategy the channel selection is an iterative matter: you can kickoff you analysis selecting a subset of the panel's marker with the most known lineage and some other channels and adding more and more markers to increasing the result complexity. Or you may start selecting all the relevant markers you have but this could be challenging with more than 15÷20 markers.

Please notice than if you load your flowSet from a previously saved set (typically handled by the flowSet optimization workflow), in case you saved it in complete mode (see 5.6.1) you marker list will be twice as long as the original one: the dataset will contains both the original and the handled expression matrix with all the related named dimensions (the original values will be the one with the "< >" brackets). Please, in this case, select just the handled markers reflecting your optimization procedure.

6.1.3 The phenotype meta-data table

It is possible (but not mandatory) to edit the *phenodata* Table 4 - The phenotype table for the cell phenotype definition.

In this case, the whole table reports in the columns the marker names which set correspond to the one selected in Table 3 - Marker selection



phenotype_name	CD57	2B4	SIRPgammaT	LAG3	CD45RA	HLA-DR	CD25	TIM3	CD39	CD62L	TIGIT	CD27	KLRG1	Vbeta13
Cluster17_var1	+	*	*	-	+	*	*	-	*	+	*	*	*	*
Cluster17_var2	+	*	*	-	+	*	*	-	*	+	*	*	*	*
Cluster17_var3	-	*	*	-	+	*	*	-	*	+	*	*	*	*
Cluster17_var4	-	*	*	-	+	*	*	-	*	+	*	*	*	*
Cluster17_stringent	-	*	*	-	+	-	*	-	-	-	-	*	-	*
phenotype_15	-	*	*	-	-	-	-	+	+	-	+	-	-	*
phenotype_07	-	*	*	-	-	+	-	-	+	-	+	+	-	*
phenotype_06	-	*	*	-	+	-	-	-	-	+	-	+	+	*
phenotype_13	-	*	*	-	+	-	-	-	-	-	+	+	+	*
phenotype_19	-	*	*	-	+	-	-	+	+	-	+	-	+	*
phenotype_18	-	*	*	+	+	-	-	+	-	-	+	+	+	*
phenotype_17	-	*	*	-	+	-	-	-	+	+	+	-	+	*
phenotype_20	+	*	*	-	+	+	-	+	+	-	+	-	-	*

Table 4 - The phenotype table

The one reported is a quite complex table in which the user defined 16 phenotype (or signatures) entries listed as single population of interest. The phenotype description is defined describing the type of marker expression that could be positive (+ means “well expressed”) or negative (- means not expressed) or non-relevant (with the symbol *). The latter choice is meaningful for those cases in which for a particular phenotype is indifferent whether the related marker is expressed or not; in all, not every marker selected for the phenotype differentiation are relevant for the definition of the single phenotype.

Usually, when multiple combination of marker is defined, not every phenotype will be mapped in the meta-cluster set: in this case some of the meta-cluster will be “unassigned”. In short, only some of the meta-clusters could be assigned to the defined phenotypes, while some of them will remains unassigned. The app will let you download a concatenated flowFrame with as a subset of events, only those being unassigned to let the user further analyze.

It could also happen that, the same phenotype could span along more than one cluster or meta-cluster, but also it could be the case in which more than one phenotype should be assigned to the same set. This is the typical case in which the phenotype is defined with one or more “*” (the wildcard that means indifferent). In this case the various selected An analysis of the heatmap in the “meta-cluster” page tab (and the pheno_full_label.csv file) should help to reconstruct the correct list. The various plot and maps produced will clearly show the presence of these undefined (OVERLAP) events and the UNASSIGNED events: the UNASSIGNED events (are always in grey in the various plots and maps) while the black dot events are the signature OVERLAP events.

While the unassigned are those event that are not covered by any entry in the phenotype table, the overlap are the ones which are not uniquely assigned because they belongs to at least two different signatures entered.

In general, the more specific, is the signature to be found, the more easily you will have bunch of unassigned with very few or none overlap events.

There is no limit on the possible choices and this could be an extremely powerful tool to analyze the phenotype of each single event following how the related proportions change overtime or correlating the population by the three chosen tags, which by defaults are *condition*, *patient_id* and *time_step* (see the Table 2 - The dataset detailed table).

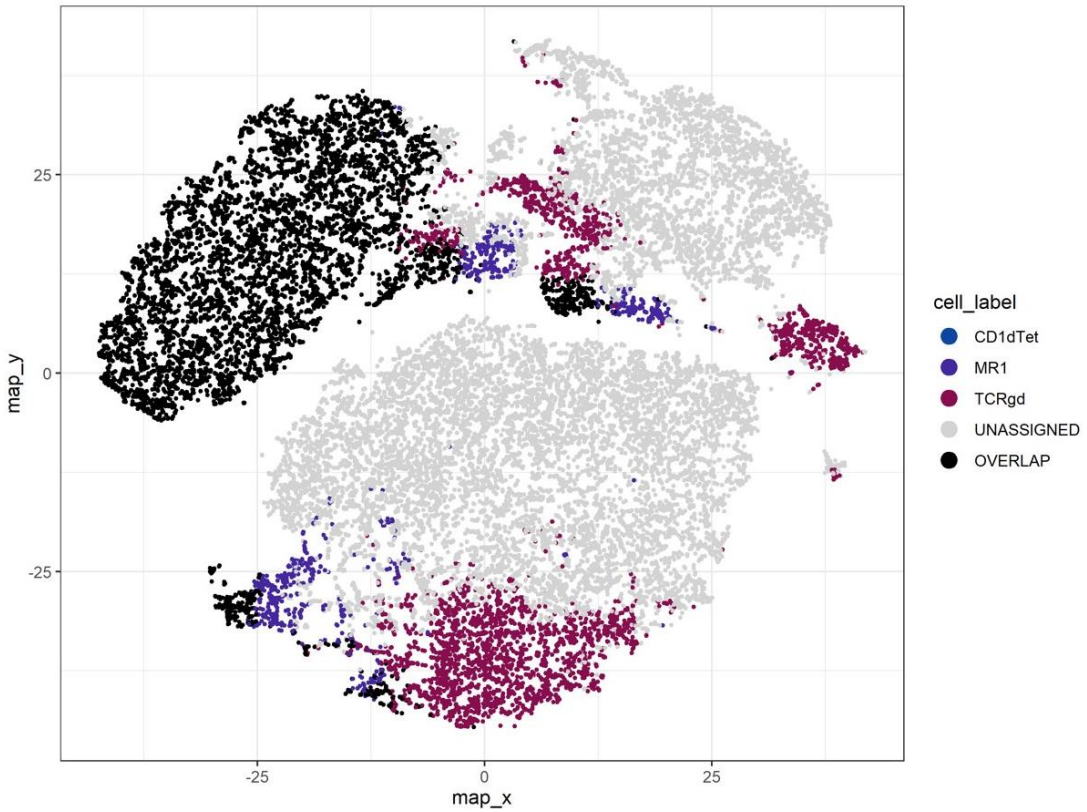


Figure 27- A tSNE map example with some found phenotypes, the black undefined (signature overlapped) events and the unassigned events

6.2 The flowSet analysis tab

Another important aim of this second workflow is to show the dataset evaluation through the MDS (Multi-Dimensional Scaling) plots (we use the limma package) which could depict the summary differences between the samples (take as a reference Table 2 - The dataset detailed table).

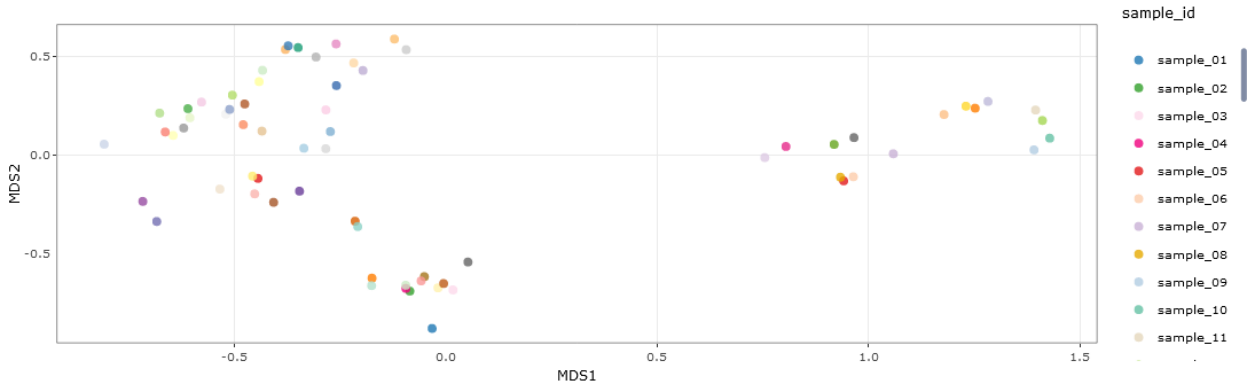


Figure 28 - The MDS plot showing the distance between samples

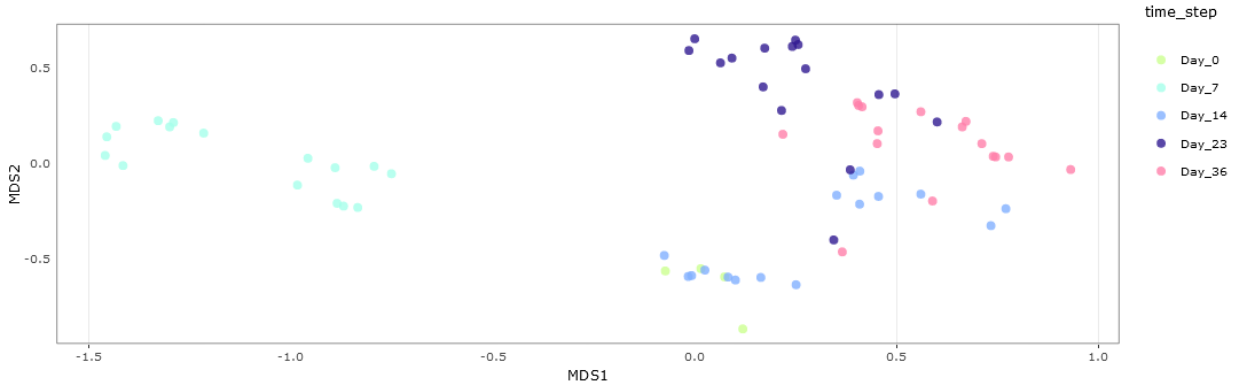


Figure 29 - The MDS plot focused on the sample time_step (tag4)

MDS projects n-dimensional data points to a (commonly) 2-dimensional space such that similar objects in the n-dimensional space will be close together on the two dimensional plot. Multidimensional scaling provides a visual representation of distances or dissimilarities between sets of objects. The objects in this case are the selected samples: samples which are more similar (or have shorter distances) are closer on the graph than objects that are less similar (or have longer distances). MDS can also serve as a dimension reduction technique for high-dimensional data. If the sample metadata are entered, the MDS plots will let you figure out the calculated distances in term of similarities between the samples itself (Figure 24). The same kind of plot can be visualized by conditions (like in Figure 25), patient_id (tag2) and time_step (tag3 – aka the three levels of sample differentiation defined in The Edit Metadata tab) helping the user to verify the effective calculated distance between the different samples.

In brief, the MDS is calculated through a sequence of steps:

1. Assign a number of points to coordinates in an n-dimensional space (in our case the dimensions are the selected markers). The axes orientation and the coordinates themselves are arbitrary and they do not have a physical meaning.
2. Calculate the distances for all pairs of points (namely, the straight-line distance between two points x and y in Euclidean space. It's calculated using the Pythagorean Theorem ($distance^2 = x^2 + y^2$) but for *n-dimensional* space. Other types of metric for computing the distances

between object are available and they are indeed recommended especially in case of high dimensional dataset. The reason for this is the “curse of dimensionality”: a jargon introduced by mathematicians to emphasize the un-intuitive phenomenon of the variation of the distances in the multi-dimensional hyperspace for the coordinates points (you can find more on https://www.datanovia.com/en/lessons/clustering-distance-measures/ and also on https://pages.mtu.edu/~shanem/psy5220/daily/Day16/MDS.html and again in https://en.wikipedia.org/wiki/Curse_of_dimensionality).

3. The distance results in the similarity matrix.
4. Compare the similarity matrix with the original input matrix. This is done to fit the data to be represented in a 'two dimension' space (which may be represented by a figure, like the plot that will be shown in this case).

6.3 The flowSet evaluation tab

An important aspect already introduced in The Down-sampling (the outliers filtering) is the possibility to evaluate the fitness of your dataset. This is extremely useful to compare the outcome state of your dataset after The flowSet Optimization Workflow. So this part of the workflow is devoted to the evaluation of the selected flowSet in terms data characteristic, like sparsity of the points in a multidimensional space, the grouping of the points in possible clusters, and even the shape of the group of points.

These evaluation algorithms take as an input the single concatenated flowFrame with the whole set of events present in each of the original sample (see The Concatenating).

First we propose a very easy method to evaluate the quality of the set of events. Definitely the Principal Component Analysis (PCA) is not suited for the typical flow cytometry dataset, because of the non-linearity of the event distributions in a typical sample. For this reason it is always preferred to utilize a tSNE-like representation, instead of ISOMAP or PCA. In any case, PCA is the most straightforward and light algorithm for dimensionality reduction (the concept behind PCA is very much similar to the MDS) and it can provide a good indication of how much spread are the events in the sample, with equally spaced set of events being the most sparse and difficult to reduce set, while in an event space with a perfectly linear point distribution, the point can be trivially represent in one axe only. On the other hand, in an extremely distributed set of events, the complex tSNE-like map could also not converge and this is the reason why we introduced this easy evaluation with PCA. In quantitative terms, the percentage of variance for each principal component can be a useful way to characterize the event space.

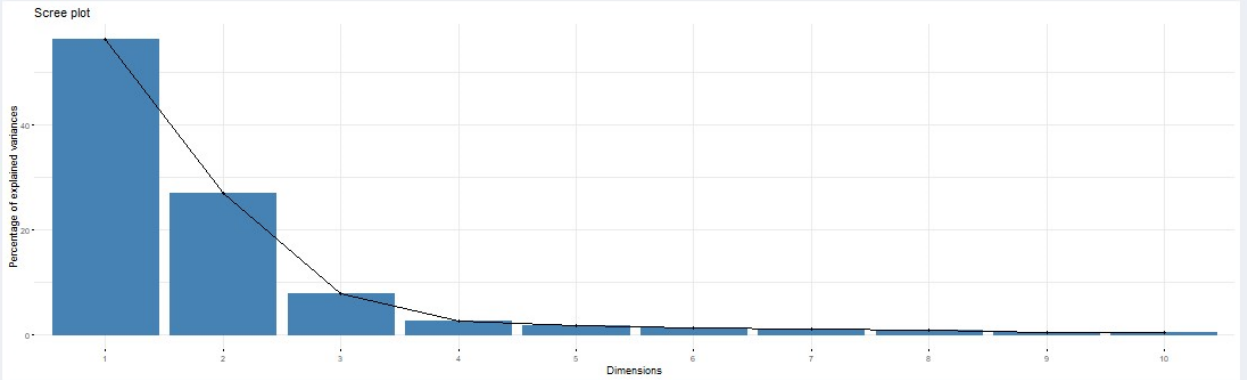


Figure 30 - The eigenvalues (scree plot) showing the percentage of variances expressed by each principal components

In the case spotted in Figure 26, the first two principal components are well above the 75%. When the trend shows a drastic reduction after the two first principal components, and the cumulative proportion of PC1+PC2 is well above than 70%, other more complex dimensionality reductions algorithm like t-SNE could be performed with no problem with a good clustering performance. On the other hand, if you are far from this safety situation, it is common to face some difficulties because the events are too sparse or too much equally distributed. In this cases the dimensionality reduction algorithms may also not converge and fail to process. In these cases, at worst, the dimensionality reduction algorithms may also not

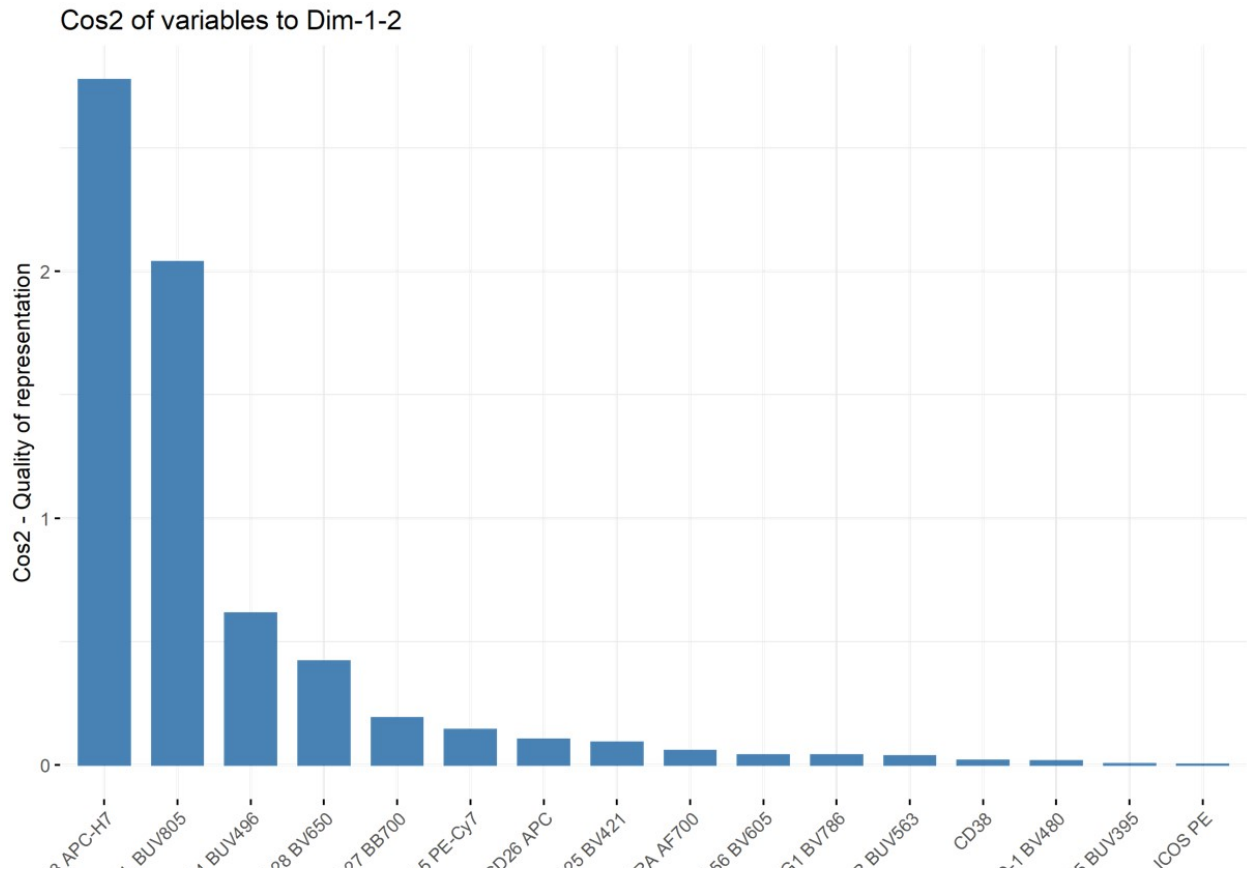


Figure 31- The quality of representation for the various selected markers

converge and fail to process and, at best, the maps could be not very efficient in differentiating the various clusters (clusters could spread along the map, there are many overlapping clusters and so on...).

Along with the histogram plot of the eigenvalues it is possible to get a deeper knowledge of the dataset in order to score the most influential channels. This is done by the graph above. This is important mostly to individuate those channels which are less representative. If you are not satisfied with the clustering results, most of the time this is due to the less representative markers which could jeopardize the clustering algorithm, providing poor quality results. Try to re-execute the clustering process eliminating the less representative channels.

Together with the mentioned graph, the PCA provides another important view of the dataset.

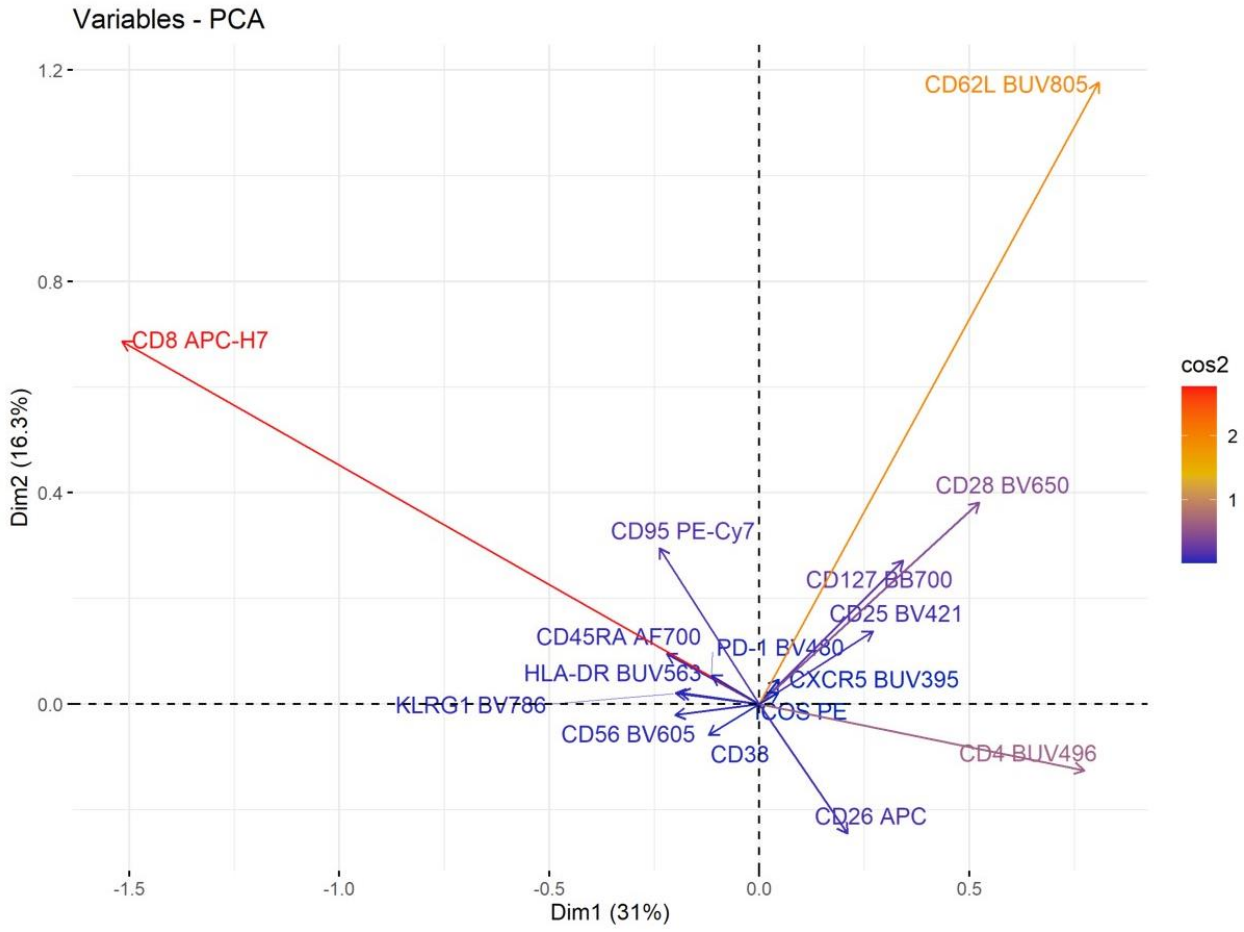


Figure 32 - The global view of the dimensions and their orientation as eigenvectors

This last plot provides not only the level of representativeness of each channels (as in the previous plot) but also the eigenvector orientation. This provides a comprehensive vision of the various channels. See for example the CD8 and the CD4: firstly the arrow length gives you an idea of the most heavy leverage of the first compare to the latter, secondly their orientations is a confirmation of the fact that where the events are CD4+, at the same time they are CD8- (as normally expected for the CD3 lymphocytes, unless

you have a case with a lot of double positive end/or double negative events for these two markers. This is the meaning of the tow arrows pointing almost at 180°.

Another important graph can be used as an entry point for the subsequent analysis. In fact, unlike supervised learning, clustering is considered an unsupervised learning method since we don't have the ground truth to compare the output of the clustering algorithm to the true labels to evaluate its performance. Since we only want to try to investigate the structure of the data by grouping the data points into distinct subgroups, the clustering only needs the number of clusters as a main input for each type of algorithms.

For this reason, one of the most used expedient in order to evaluate the number of cluster, is the elbow method with the k-means algorithm. The k-means in fact, is a fast enough and not too much resource demanding algorithm in order to be executed several time varying the number of clusters, computing the centroids for the clusters by taking the average of the all data points that belong to each clusters. The elbow method gives us an idea on what a good k number of clusters would be based on the sum of squared distance (SSE) between data points and their assigned clusters' centroids. As you can see from the figure below, from the 20 to 30 number of cluster, the curve remains almost constant: a number around 25 could be used as an entry to the clustering algorithm.

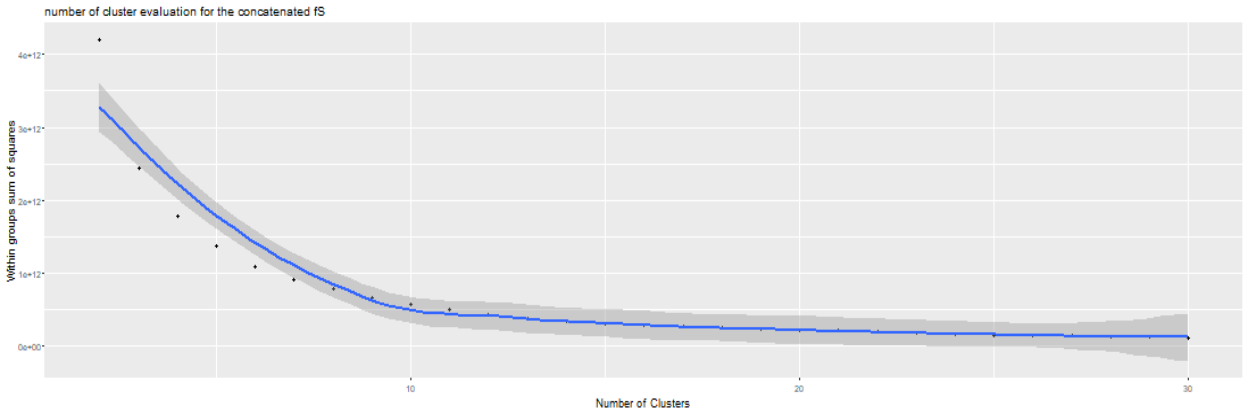


Figure 33 - the kmeans cluster projection

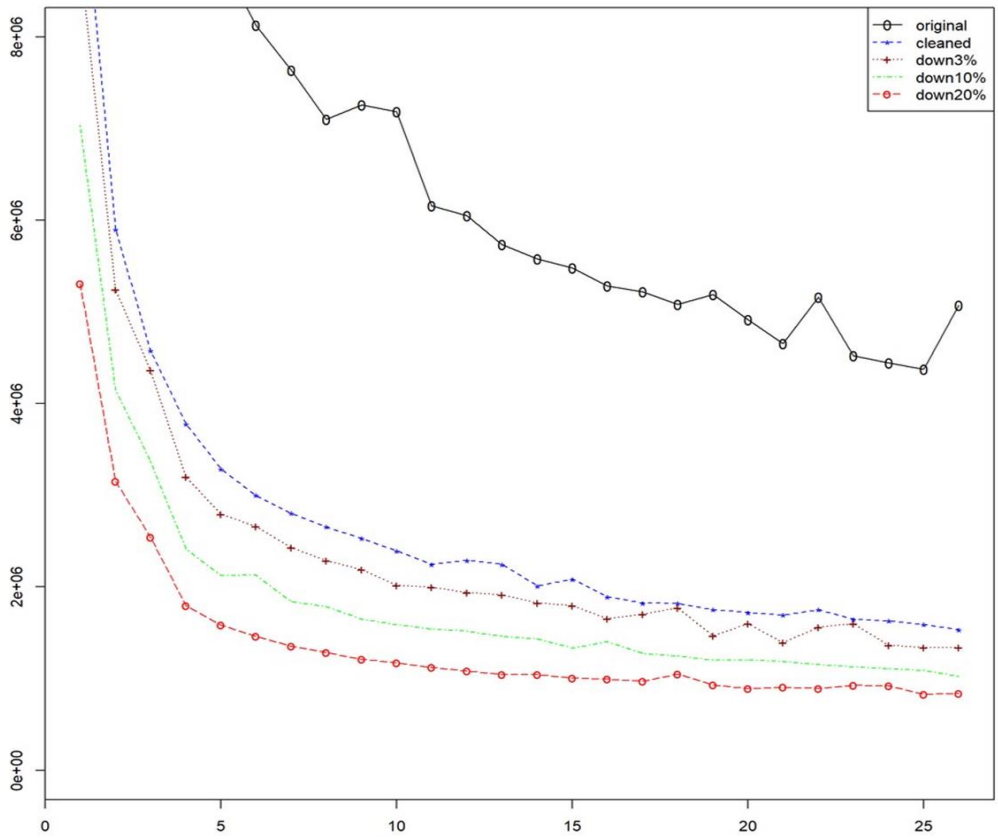


Figure 34 - the elbow method with k-means in different flowSet

It is also interesting to notice the different curves in Figure 28 - the elbow method with k-means in different flowSet. The upper side curve is related to the original flowSet in which no optimization has been performed. As you can see, even the number of cluster evaluation is not as easy as in the curves below in which a different level of down-sampling was applied.

Other additional techniques could have been implemented to evaluate this important data, but the main drawback here is due to the algorithm complexity and their bad scaling behavior: depending on the amount of events the results may take an excessive amount or resources in terms of memory and time.

Nevertheless, a more precise method has been introduced due to its different algorithm kernel. In fact, while the *kmeans* algorithm is kind of *data agnostic,* several other clustering algorithms are based on the statistical assumption that reflects the type of event distribution composing the dataset. As already noticed in the Introduction, the flow cytometry event distribution is Gaussian. That's the case of the *mclust* R package which collects several model-based algorithms for clustering, classification, and density estimation based on finite normal mixture modelling. It provides functions for parameter estimation via the EM algorithm for normal mixture models with a variety of covariance structures, and functions for simulation from these models.

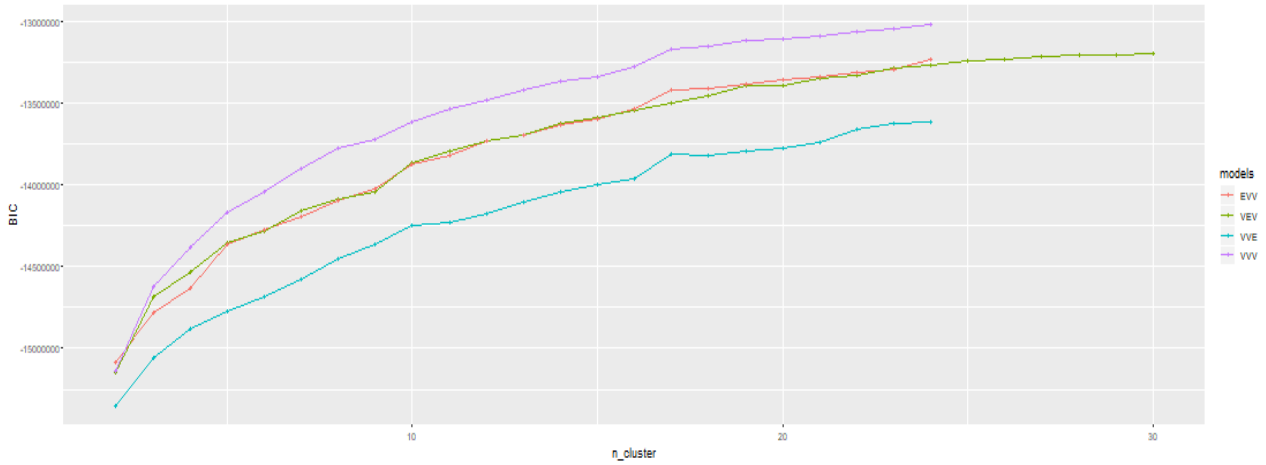


Figure 35 – The various m_clust evaluation trends

The Figure 29 shows al together several different models. You should choose the one with the best estimation. Also in this case (like in the previous based on *kmeans* algorithm) the optimal number cannot always be unambiguously identified. To better understand the output you can follow the mclust references or the related vignette).

In short, the *mclust* package comprise a set of algorithms which are based on the Gaussian finite mixture model fitted by EM: in particular it implements three ways to classify the distribution: shape, volume and orientation. The three capital letters refers to those distribution typology[14]. Depending on the entry dataset volume and on the max number of cluster set, the calculation may takes a very long time!

Other details about finding the number of clusters can be found on

6.4 The Map evaluation tab

This last tab of The Metadata & Assays Workflow is to test and compare the tSNE mapping algorithm where you can produce and download the tSNE[15] maps related to the flowSet handled in the 'Pre-clustering workflow' tab. This process works only if the flowSet was handled through the optimization flowSet workflow (you should select at least one of the optimization flowSet action if you want to analyze the tSNE map)

You will have an 'initial' and a 'final' plot related respectively to the entry flowSet and the manipulated flowSet with the last performed of the following process in the chain: cleaning, scaling, aligning and down-

[14] E.g: 'VEV' means ellipsoidal, with equal shape, 'EEE' means ellipsoidal, equal volume, shape, and orientation, 'VVV' which means ellipsoidal, varying volume, shape, and orientation. This is the most generic type of Gaussian distribution in which each single cluster could have different volume, different shape (spherical, diagonal or ellipsoidal) and different orientation, which is the case for most of our experimental samples

[15] cytoChain implements the fast Barnes-Hut version of t-SNE (an $O(N \log N)$ algorithm) which let you generate an approximation of the original tSNE (an $O(N^2)$ algorithm). There is the possibility to generate the original tSNE setting *theta* to 0 (*theta* should be between 0 and 1, and its default value is 0.5.

sampling. The aim here is to compare the two maps, before and after the flowSet handling. The two different tSNE plots are related to the original flowSet and the one related to the flowSet handled after the latest relevant action you performed within the optimization flowSet workflow. With this it is possible to compare the two tSNE maps: initial and final.

Algorithms like tSNE are more complex than other dimensionality reduction technique like the linear PCA. As a matter of fact, these non-linear technique for dimensionality reduction are particularly well suited for the visualization of high-dimensional datasets performing a kind of clustering because they calculate the probability of similarity of points in higher dimensional space and calculating the probability of similarity of points in the corresponding lower dimensional space. Once putting similar events together they implement a probabilistic strategy to spread these clusters of events apart to visualize them in a proper map.

Along with the visualization feature another possible tSNE purpose could be to use its map as a basis for a successive clustering algorithm which will act on the mapped events and their mapped islands in different clusters. This method could be useful in case of very complex samples with a lot of different clusters in which the events seems to be very much spread without much aggregating point, in other word in those cases in which a statistical clustering methodology which rely on Gaussian distribution of the events does not perform well. In such cases it could be useful to color the various clusters starting from a tSNE map collecting all the events in a concatenated flowSet to further study them with other techniques like in .

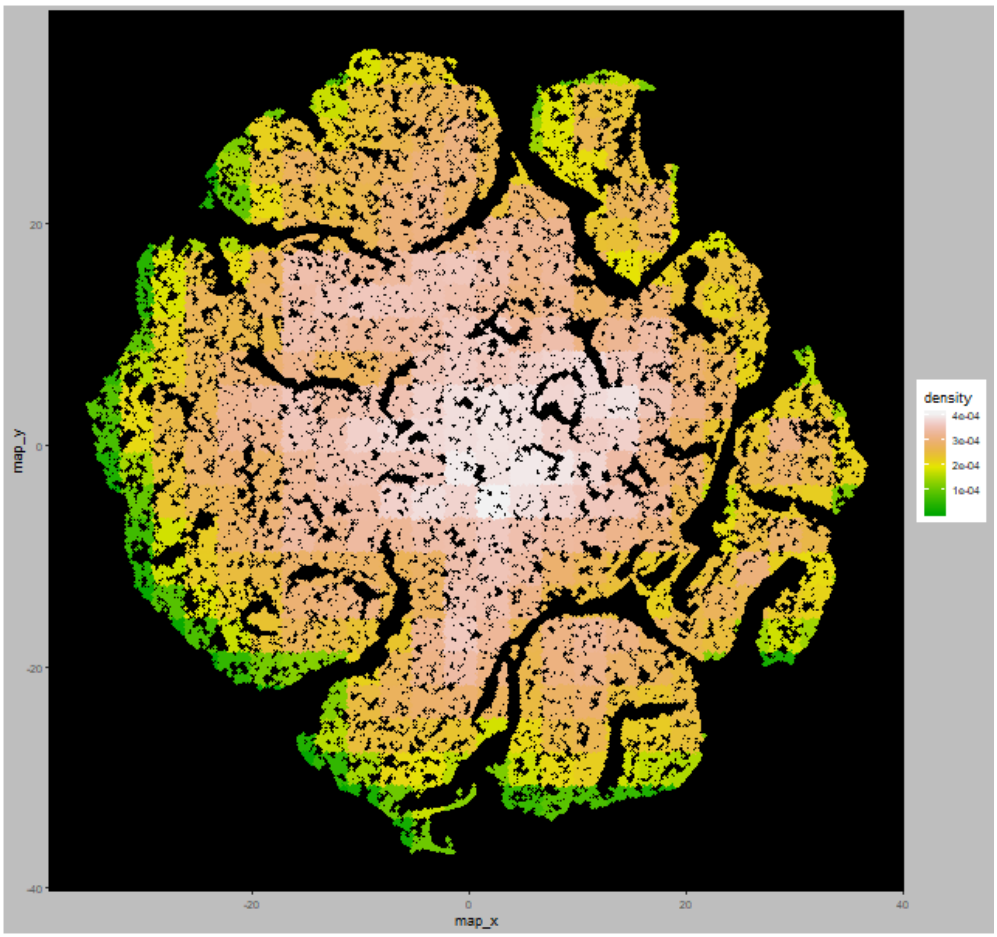


Figure 36 - tSNE map

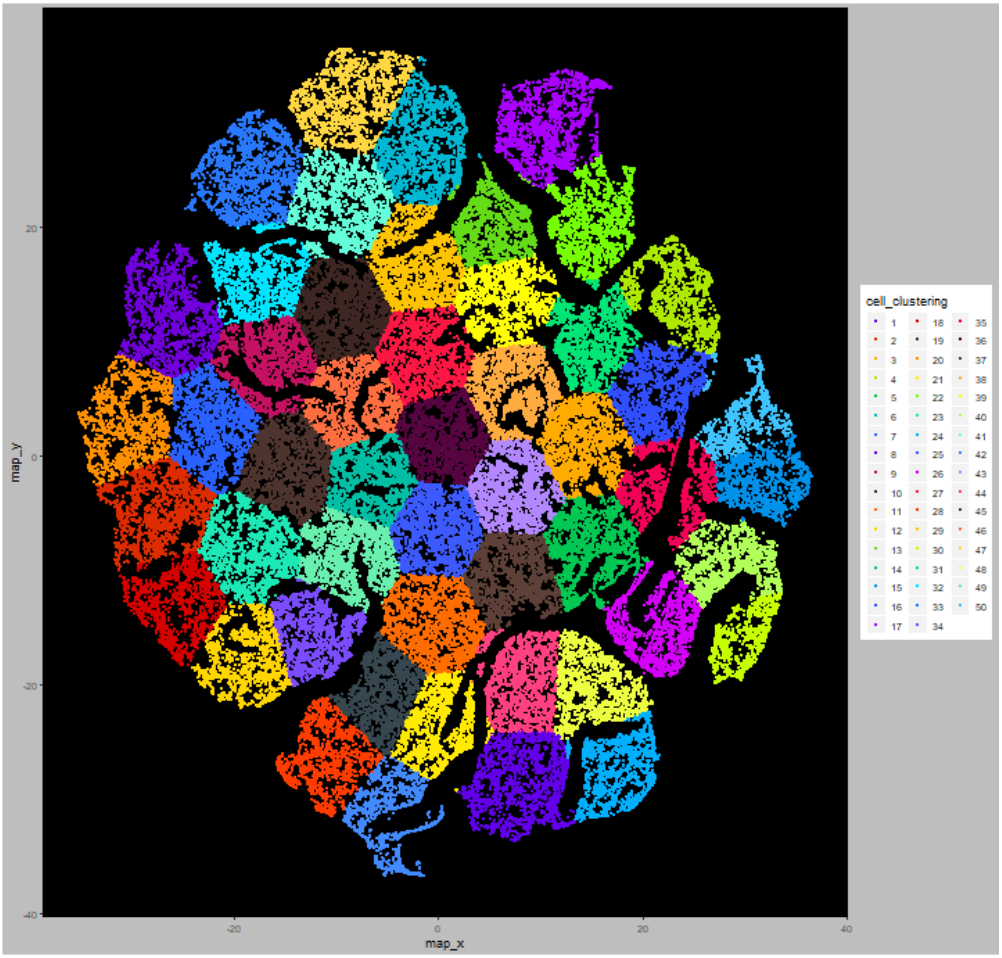


Figure 37 - tSNE map clusterd with K=50 centroids

Interesting link: https://stats.stackexchange.com/questions/238538/are-there-cases-where-pca-is-more-suitable-than-t-sne

7 The High Dimensional Analysis Workflow

The final workflow can be seen as the final goal of the whole analysis: to compare the various population by means of plots and tables allowing the user to evaluate the results (even if cytoChain could be just used for example to prepare the samples in the optimization flowSet workflow and use them as an input to other environment and other available tools).

There are plenty of different clustering methods and a perfect method do not exist since it depends very much on the nature of the dataset to analyze. A flowSet in which most of the events can be easily grouped in well-defined Gaussian plots without too much spreading across different markers can be better "clusterized" using a statistical type of clustering (like for example immunClust or flowMeans) weather in case of a very spread event distribution in the expression hyper-space a data-agnostic algorithm (like the many implementation of kmeans, like flowPeaks) or again a hierarchical clustering algorithm (like RchyOptimyx) could perform better.

One of the aim of the cytoChain implementation is to collect the most used and popular algorithms to allow an analysis comparison letting the user the choice of the best algorithm to choose. But to cluster is often non-sufficient since for the most cases in the high-dimensional flow cytometry, a good grouping performance is based on dozens and dozens of clusters. In order to facilitate the analysis, is good to work with a smaller amount of clusters. For this reason the meta-clustering module allow the user to specify a smaller amount of groups using a consolidated algorithm (the ConsensusClusterPlus typically used in genomics to reduce the amount possible gene's expression).

7.1 The implemented clustering algorithms

One of the most used algorithm is flowSOM which is widely used in other context (like in the imaging recognition field) and seems to be the most promising and one of the best in terms of performance (see (Lukas M. Weber 2016)).

kmeans is by fa the most commonly used unsupervised machine learning algorithm for partitioning a given data set into a set of k groups (i.e. k clusters) algorithm even if it is not the best in terms of performance for our typical cytometry dataset, especially when the number of dimensions are more than 10 (for a good reference for kmeans and its algorithms implementation see https://www.tqmp.org/RegularArticles/vol09-1/p015/).

A valid alternative to the flowSOM (see https://www.nature.com/articles/s41596-021-00550-0), is another important algorithm currently implemented in cytoChain: phenograph is a hierarchical clustering algorithm (see https://pubmed.ncbi.nlm.nih.gov/28865188/).

Here in Table 5 a summary of the pros and cons between the currently implemented algorithms is reported:

FlowSOM	Phenograph	Kmeans
Fastest	Much more slower	Fast
Number of clusters = 100 (fixed – cannot be set)	Number of cluster undefined	Number of cluster can be set
No parameter	Number of number of nearest neighbors	Type of implementation algorithm (Hartigan-Wong, Forgy, MacQueen)
Rich and significative feedbacks	Poor feedback	Poor feedback
More suitable for finding trends in complex experiment	More suitable for finding rare populations and values	

Table 5- The comparison between the current implemented algorthms

FlowSOM implementation in cytoChain does not allow the setting of the number of clusters: it needs a meta-clustering algorithm to reduce the clusters of events from 100 (the initial fixed amount) to the number of meta-clusters. The amount of meta-cluster can be inferred from the elbow method on kmeans clustering algorithm (see 6.3) but an alternative way could be to look at the MST plot produced by the FlowSOM algorithm itself. FlowSOM produces a lot of information along with the main outcome of the "clusterization" process which consists of the assignment of each single event to a clusters.

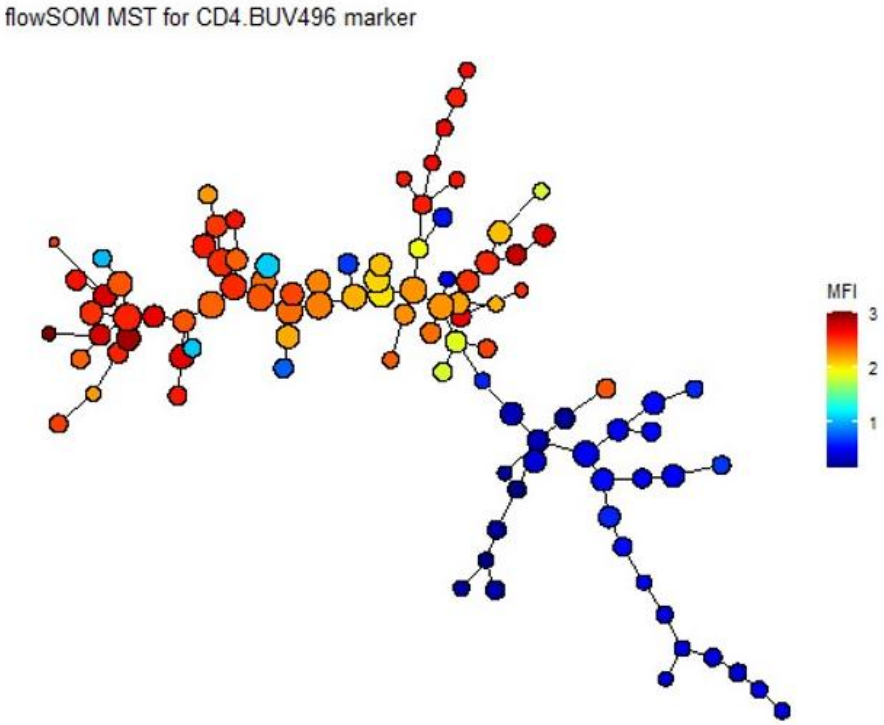


Figure 38 - An example of MST for as single marker

The MST (Minimal Spanning Tree) is an efficient way (an alternative to the dendrogram of the heatmaps) to provide a representation of the various clusters. The two MST plot shown here referrer to the same dataset. Normally a good rule of thumb for infer the number of clusters (or meta-clusters) is to count the main branches of the tree.

Also with Phenograph you cannot set the number of clusters. In fact, Phenograph does not accept as an input the number of clusters: the K parameter only fixes the number of number of nearest neighbors. Thus, the larger is K, the lower is the amount of clusters and the higher is the algorithm depth to converge finding the best way to group the K events. On the other end, the smaller is K, the easier for the algorithm to find out groups of K events, but also larger is the amount of possible clusters fitting the K events.

Rphenograph performs particularly well for finding rare populations (maybe for some markers not particularly brilliant, or for critical dextramers). In this case evaluating in advance (via the manual gating process) the amount of cells for that particular signatures could help defining the minimum amount of of nearest neighbors.

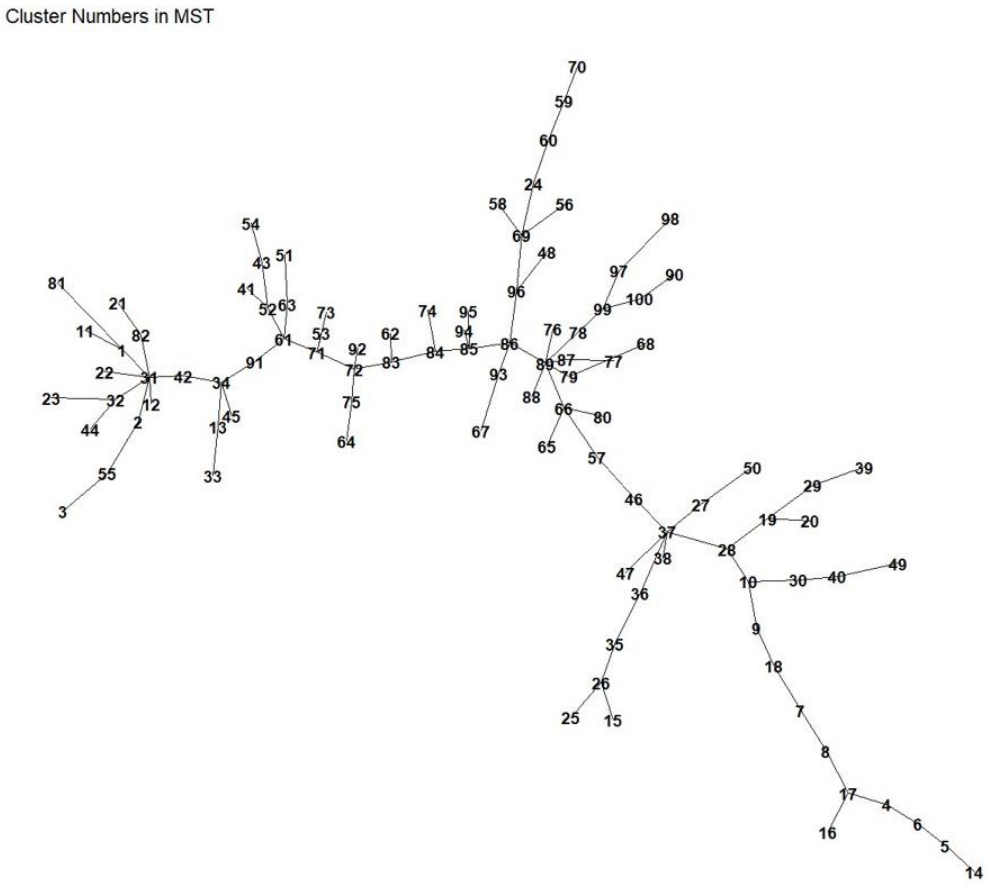


Table 6 - The STAR version of the MST with all the numbered clusters

7.2 After the clustering process

After the grouping/differentiation operation is ready by the clustering and meta-clustering, it needs to map the results on a visible event representation. In fact, with more than three dimension is not possible to visualize the expression matrix with all the single event represented as a point in this multi-dimensional space. For this, the dimensionality reduction algorithms come in handy. As a matter of fact, in many machine learning context, the dimensionality reduction is one of the most consolidated way to reduce the whole amount of data knowing exactly the amount of information lost in the step. In our search field, the dimensionality reduction is mainly used to visualize and represent a complex amount of data which otherwise could only be possible through the binomial 2x2 plots like in Figure 2.

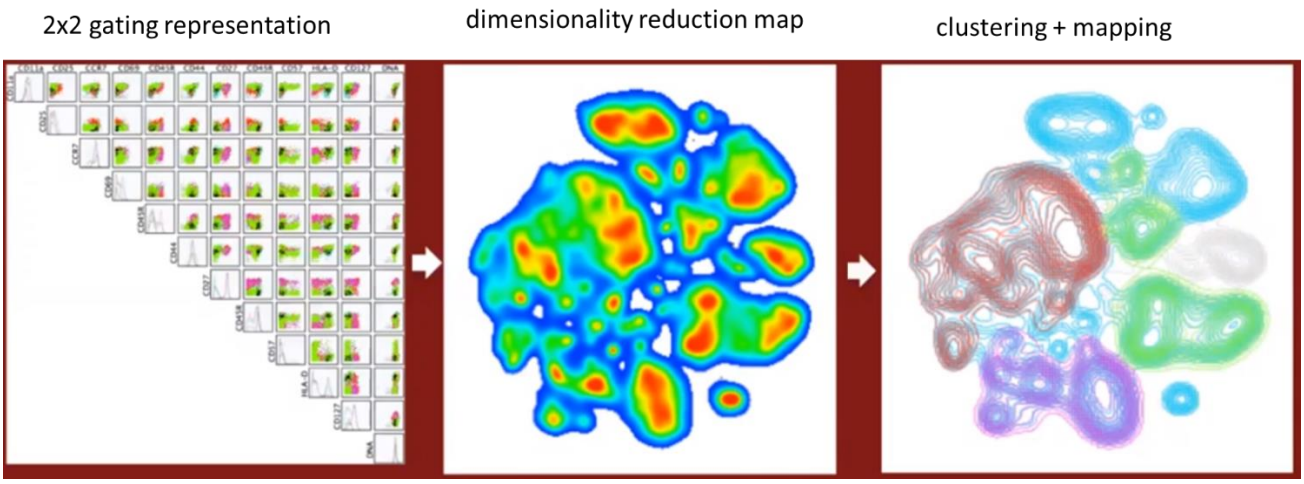


Figure 39 - from the 2x2 gating to the clustering & mapping

The visualization is not trivial since the most commonly used algorithm like PCA and ISOMAP do not perform well with the spread and nonlinear dataset which typical in cytometry. The first dimensionality reduction criteria that really helped with this limitation was the t-SNE (t-Distributed Stochastic Neighbor Embedding) map (and its various algorithmic implementations).

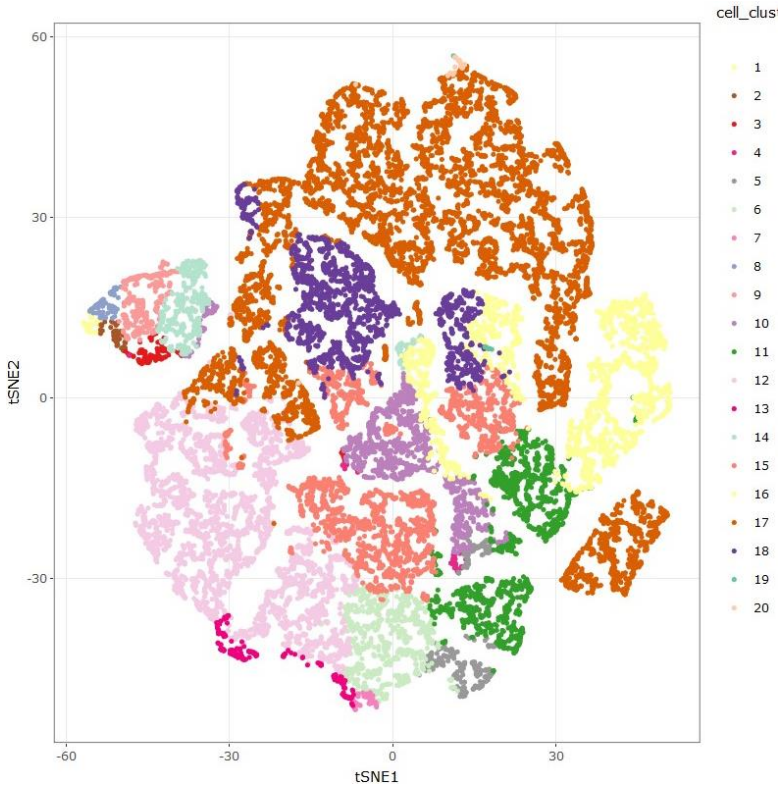


Figure 40 - a tSNE map with 20 clusters

While the tSNE seems to be the best in isolating the events in homogeneous scattering areas with sharper contours, and also reflecting better the amount of events and the related quantities, the more recent UMAP's often performs better at preserving aspects of the global structure of the data. This means that

it can often provide a better “big picture” view of the data with the more similar expression profile closer than the one more different expression as well as preserving local neighbor relations like in tSNE case.

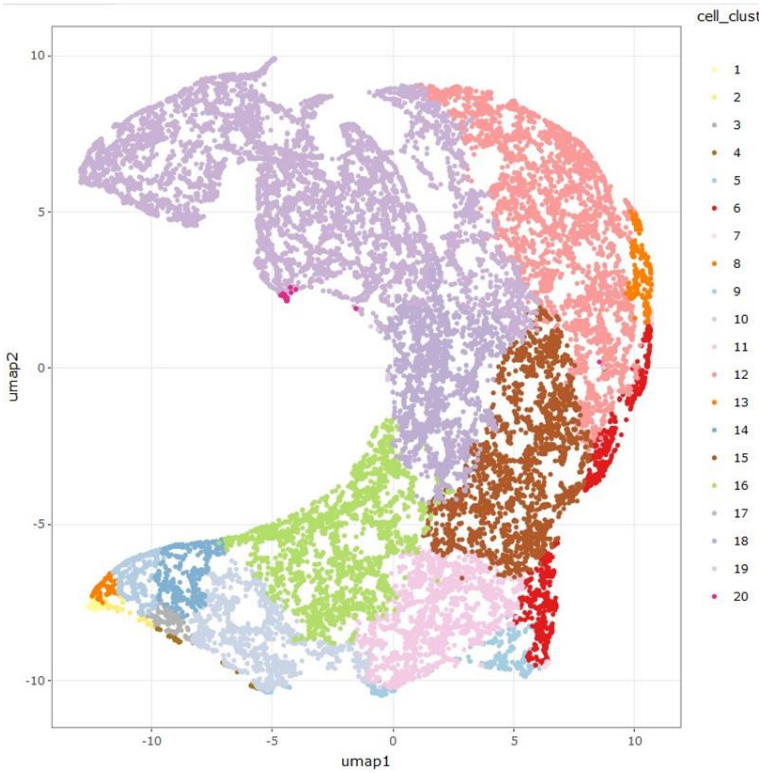


Figure 41 – the related UMAP representation

It is important to empathize that when the clustering classification, the meta-clustering classification and the subsequent mapping with the generation of the various dimensionality reduction coordinates are performed, these new elements are added in the expression matrix along with each single events, composing in this way an extension of the original flowSet. The related flowSet for each operation can be downloaded leaving to the user the capability to compare the result with the ones of other environment or from other analysis conducted on the cytoChain itself.

A further step is to assign a name to each cluster in the map trying to compare the different marker expression or to assign each cluster a phenotype following the already introduced like in Figure 35.

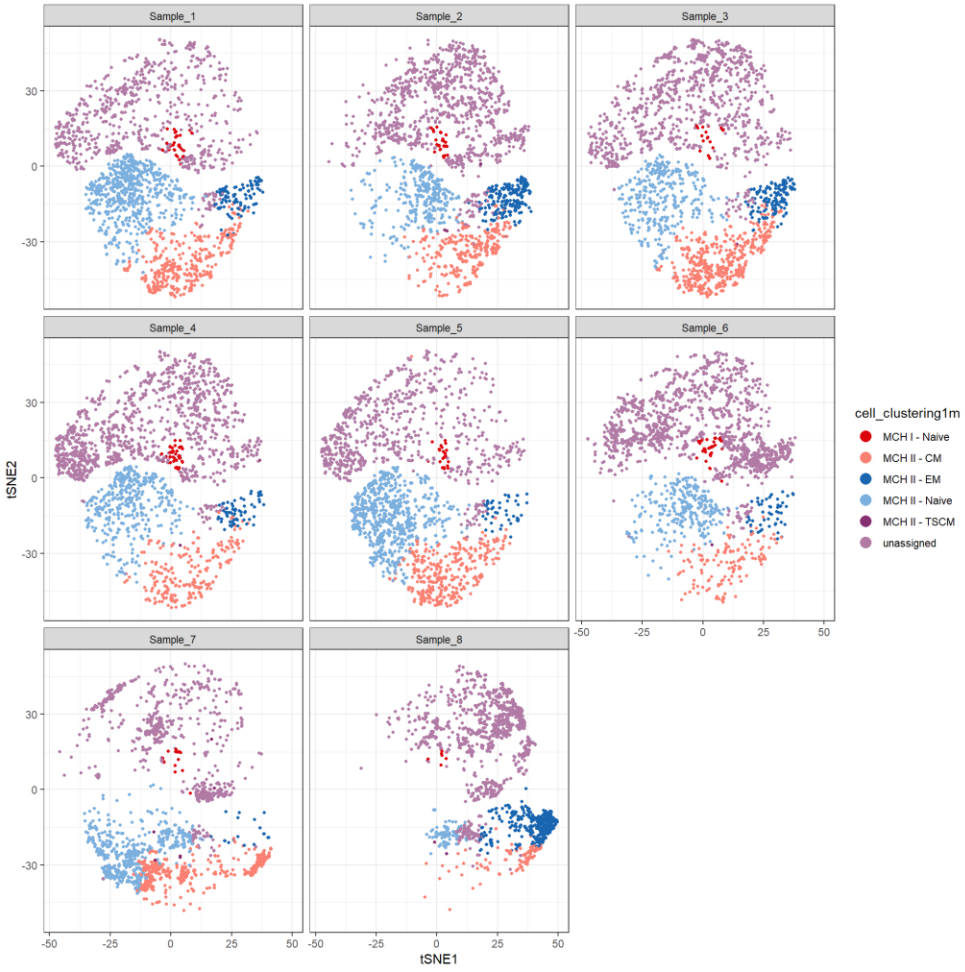


Figure 42 -mapping the phenotype table entries on the t-SNE map referred to each single sample

Of course mapping is not the only way to show the events. To better grasp the magnitudes involved for each clusters and to better organize their main differences, an heatmap and its related table is surely more indicated (see Figure 36)

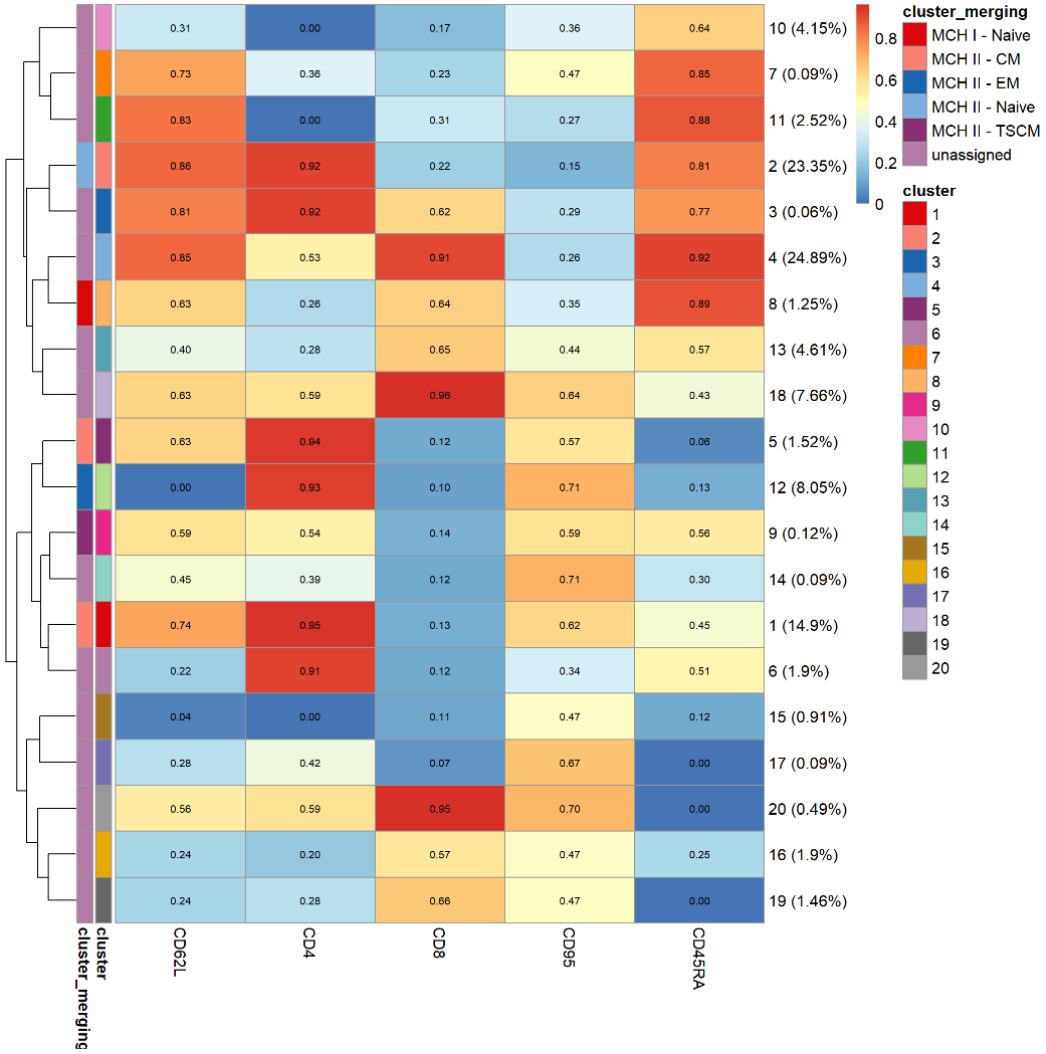


Figure 43 - Heatmap of the marker intensities

A simple histogram is even more straightforward to read when we focus just on relative quantities (Figure 37).

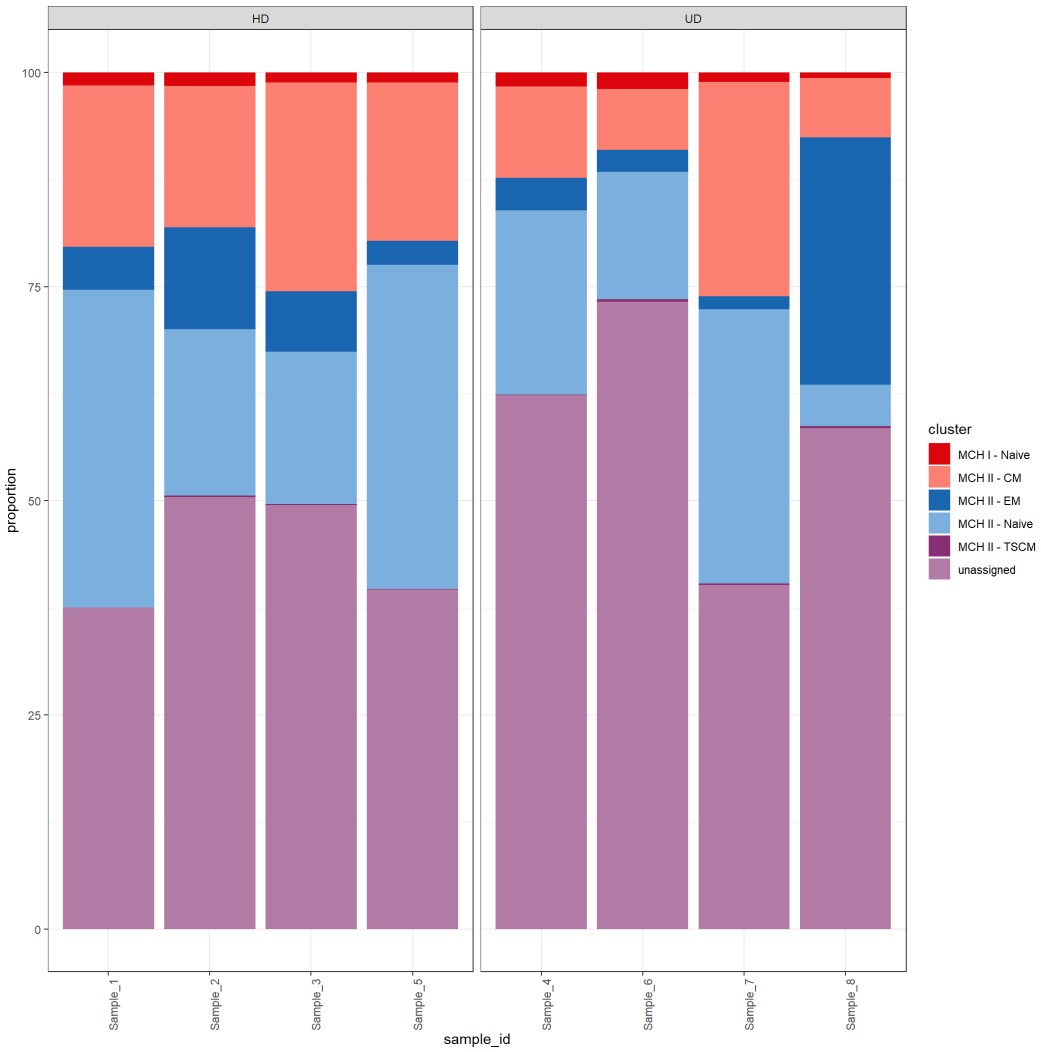


Figure 44 - histogram showing the various phenotype proportion for each condition (tag1)

Or again to splitting by each user defined phenotype (Figure 38):

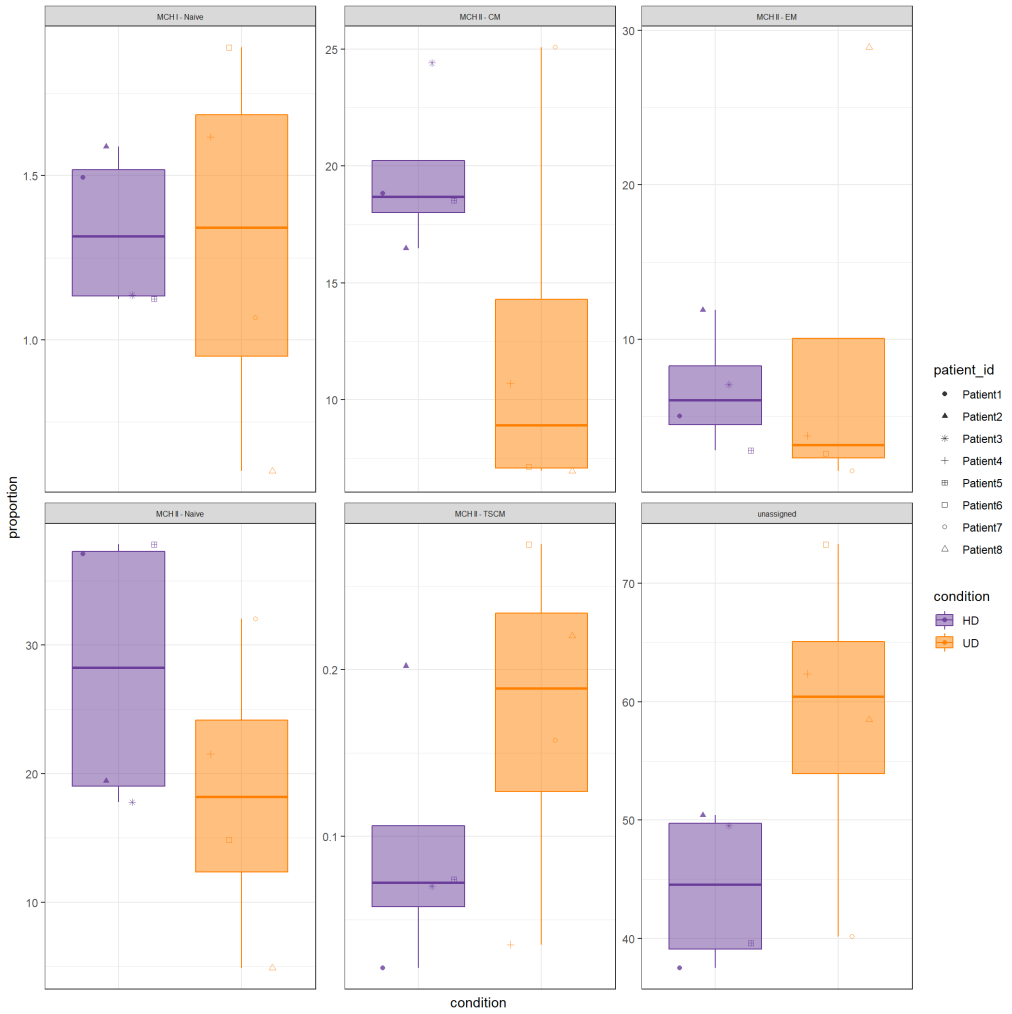


Figure 45 - Barplot showing the condition/phenotype

More type of figures could be implemented in order to provide always a more refined and efficient way to depict the various analysis elements (think about the streamgraphs to show the expression evolution through the time_steps).

7.3 The different strategies for the phenotype analysis

The clustering workflow provides a clustering operation (using one of the selected clustering algorithm) which may be followed by a meta-clustering (we use the *ConsensusClusterPlus* algorithm – see The High Dimensional Analysis Workflow). Reducing the amount of groups of similar events to a reasonable number of homogeneous sets [16] could help in simplifying the analysis the data results, the map plots and so on.

[16] E.g. typically the flowSOM algorithm is tuned by default with a number of 100 clusters: collapsing this great amount of events set to a lower amount of 10÷20 could leverage the analysis. It should be also possible to perform this type of phenotype analysis directly on the clustered samples (before the meta-

We named the operation of mapping a set of edited phenotypes (by the phenotype table entered in The Edit Metadata tab) to the various calculated clusters, the *merging* operation. It is possible to map the various entries expressed by the phenotype matrix to the clusters (within the *Merging I* tab) and to further map the phenotypes in the meta-clusters (within the *Merging II* tab). The procedure of assigning a certain cluster to a certain phenotype is the merging operation in which every single event belonging to a certain cluster is labelled and assigned to certain phenotype.

In any case the cluster/meta-cluster option is not the only way to perform a phenotype analysis. Let's introduce the three different analysis strategies:

1. Clustering + Meta-clustering (no Labelling)
2. Clustering + Labelling
3. Clustering + Labelling + Meta-clustering

The following paragraphs will introduce each single strategy.

7.4 Phenotype analysis – Type 1: Clustering + Meta-clustering

Let's suppose that you do not have any idea about the possible phenotypes which could be present inside your experiment. Of course you select a certain panel to get some results from your experiment but as often happens, apart from some set of lineage markers, it could be difficult to compose data frame like Table 4. Another possibility is that you just do not care about any specific phenotype and this is a common case especially in high dimensional flow cytometry analysis when you just set up a big panel with more than 20 markers without dealing with any specific indication about the outcome of your experiment.

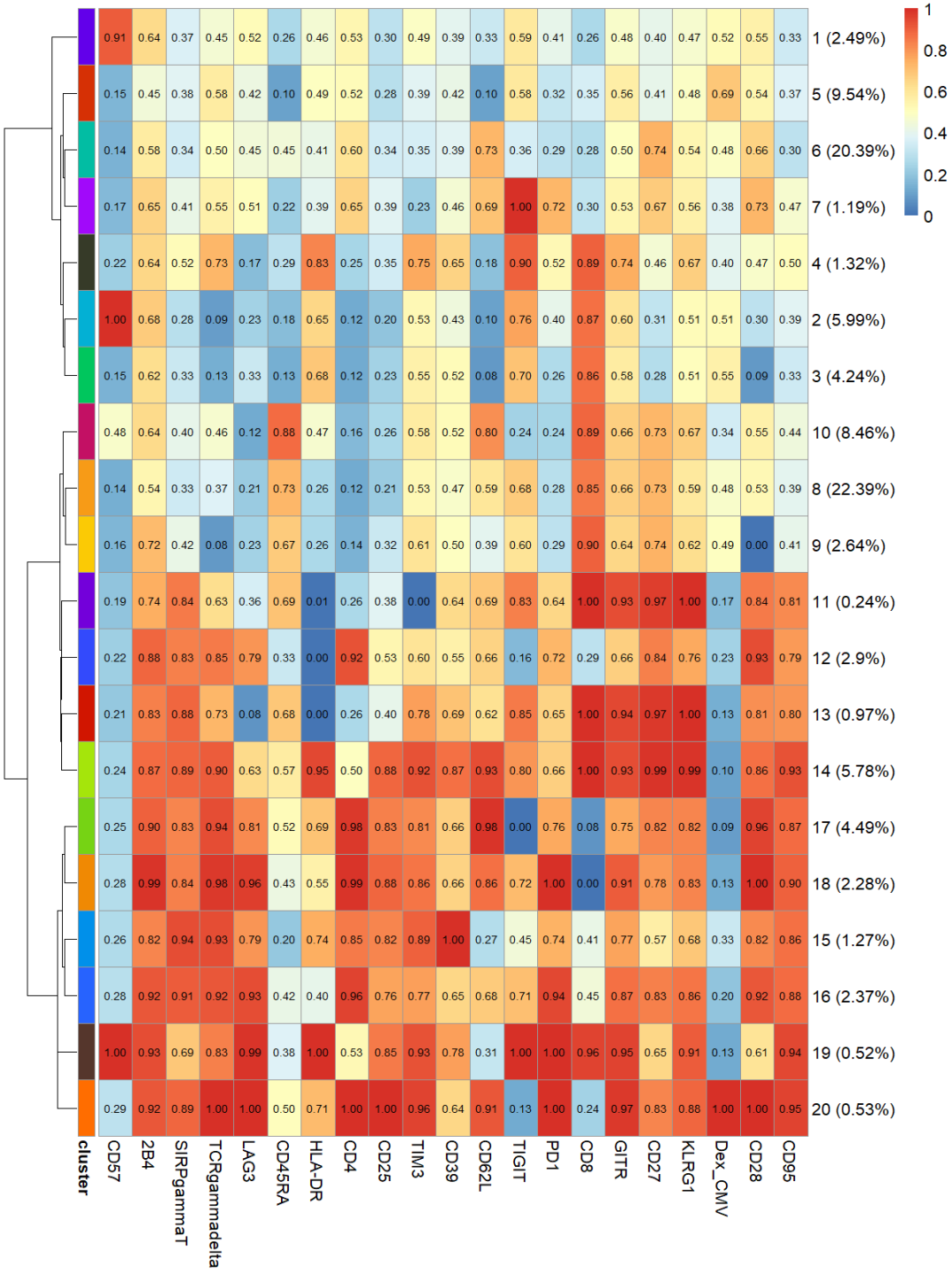
In this case the clustering algorithm will split up your total set of event (collected in a single concatenated flowFrame) assigning each of them to a specific cluster of a set of *Num_cluster*: so then, each event belonging to a cluster will be assigned an additional dimension (a *clusterId)*.

Since most of times, especially with very complex experiments, you will deal with a lot of clusters, it is always better to reduce this number of clusters with the meta-clustering operation. The meta-clustering operation will just re-group each single cluster within a meta-cluster, namely a "cluster of clusters. The meta-clustering algorithm currently implemented in cytoChain is the ConsensusClusterPlus which will collects in input a data *frame [nr.of clusters x marker expression]* with the median expression of each single marker within the single cluster. Other input are parameters like the specific algorithm to use, the adopted measurement distances between the events and of course the number of meta-clusters which will be smaller than the number of clusters in input.

With this you will get all the analysis results, the plots and the related quantity tables without any specific name assigned but just with a number which define a specific cluster first and another number related to the meta-cluster. One of the main outcome could be the table with all the possible phenotype in each calculated meta-cluster which is possible to obtain from the "Meta-clustering" tab in the workflow: a table

---

clustering operation). This could help in finding potentially more marker expression combination but a price of having a lot of events sets.



in *.csv* format which will collect the phenotype as depicted in 6.1.3 together with the correspondent heatmap table (see Figure 39). The translation in terms of phenotypes, it could be in turn the input in the "Metadata and Assays" workflow as a phenotype table to be further studied in a subsequent session.

Figure 46 - A meta-clustering heatmap

As you can see the various meta-clusters are just identified with a number and not by a consistent name.
It is always possibile to visualize the various in the maps related to each specific tag defined in the sample meta-table. If the tSNE dimensionality reduction map with all the events colored by clusters is the following:

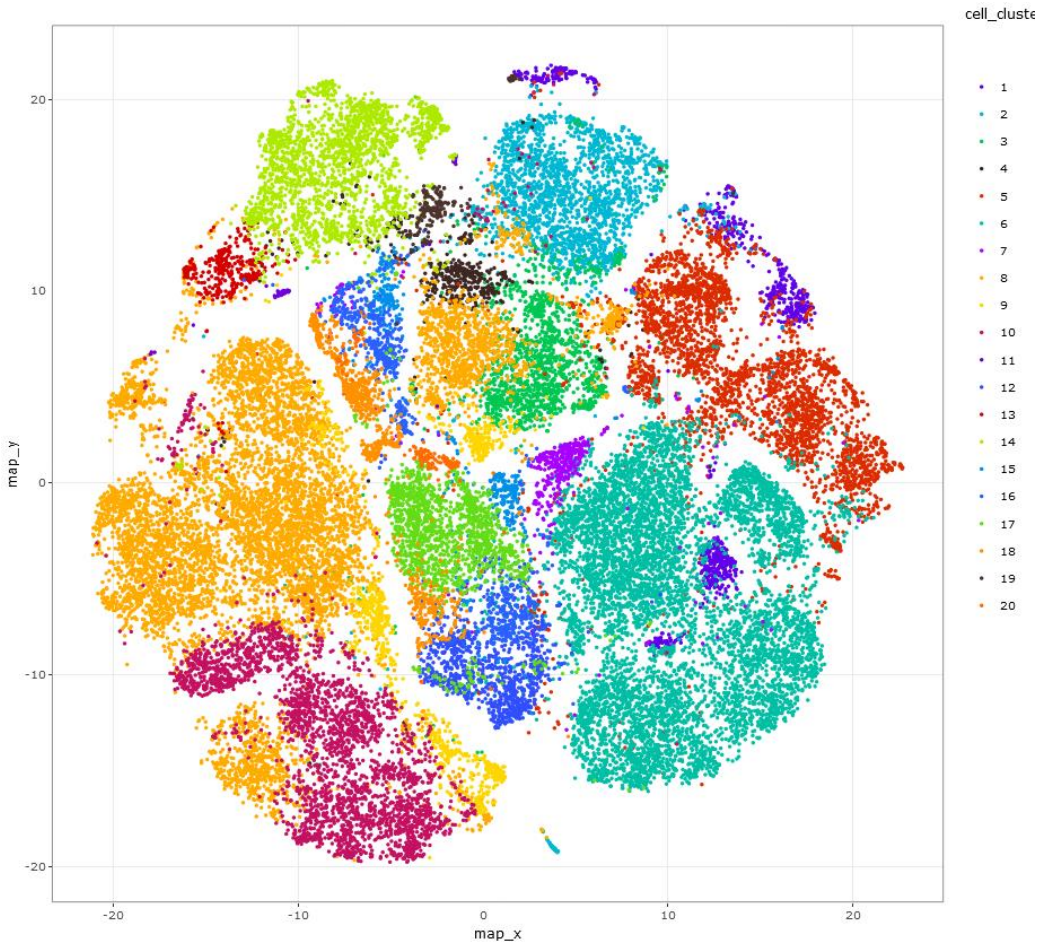


Figure 47 - tSNE map for 20 clusters

Then the related maps for a certain tag entered in the sample meta-table will be as follow:

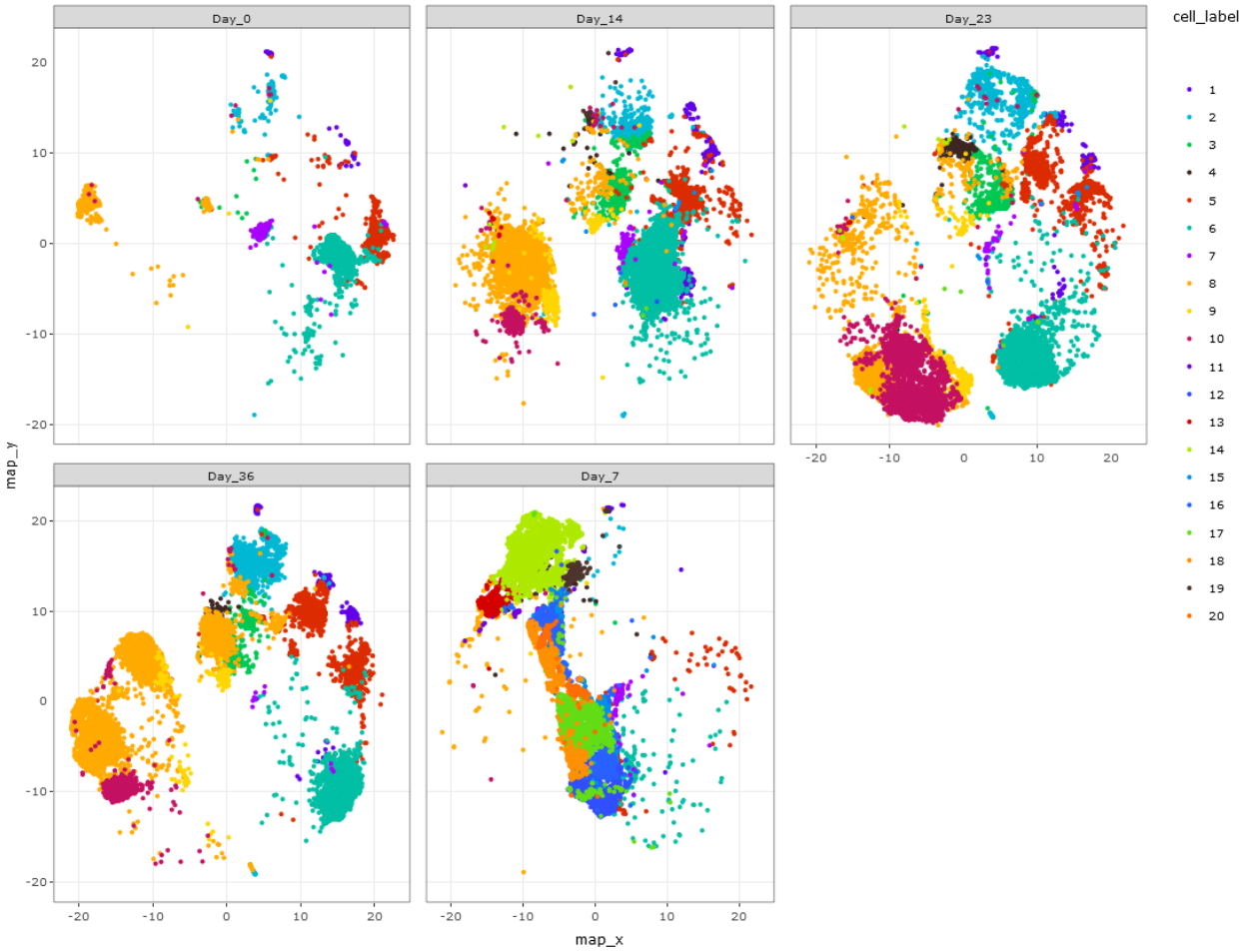


Figure 48 - The tSNE maps related to a single tag edited in the sample meta-table

While the quantities are better shown in the following barplot:

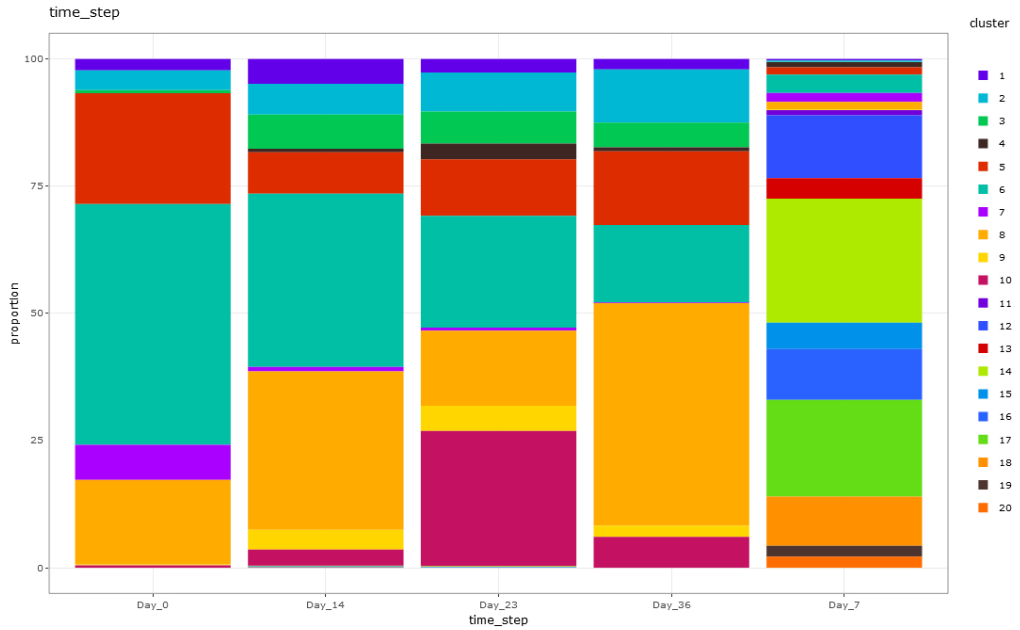


Figure 49 Quantity bar-plot for the cluster in selected tag

or, in case of time-step related tag, a stram-plot is also available:

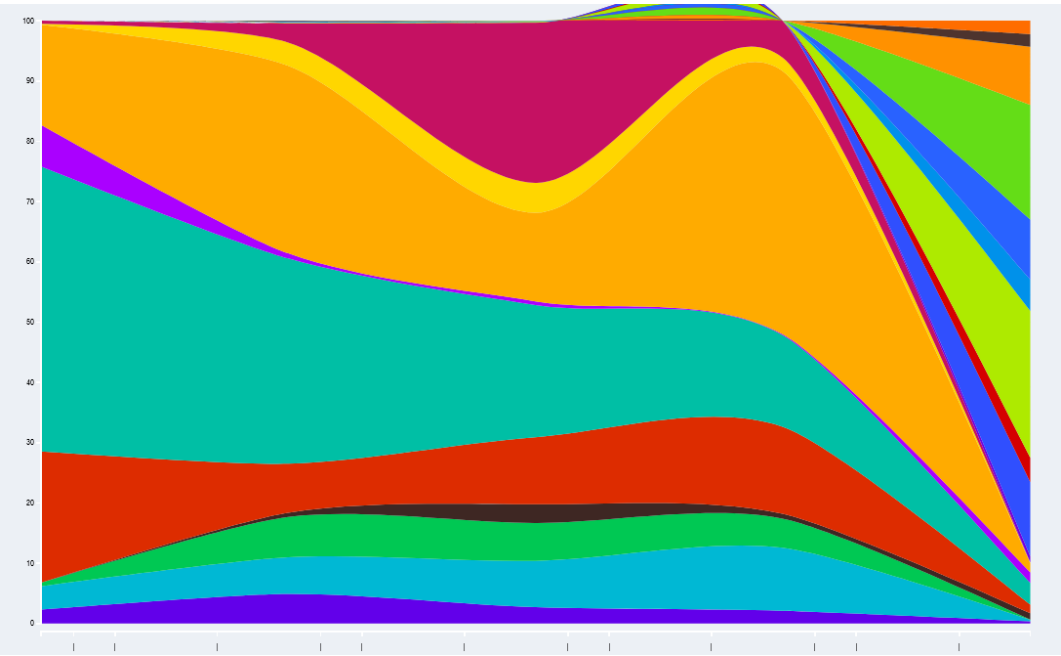


Figure 50 - A stream-plot for the quantities in a time step tag

7.5 Phenotype analysis – Type 2: Clustering + labelling (no meta-clustering)

The meta-clustering could reduce the number of entries and thus simplifying all the analysis but, as a drawback, it can reduce the detailed sub-setting introduced by the selected clustering algorithm. In fact, since a cluster could have a certain median expression for each single channel, a meta-cluster will have a median of a set of clusters collecting altogether more groups of events. Furthermore, while a panel of *N* channels could have 2^N combinations corresponding to the number of different phenotypes, a clustering algorithm calculating the expression of *K* clusters while discriminate *K* > *M* number of phenotypes (in *M* is the number of meta-clusters) . This strategy will be suitable in those cases in which it is important to find a very specific phenotype which can be formed with very few amount of events, namely a case of rare events.

In this case it can be useful to skip the meta-clustering and concentrate just to the clustering output with *K* (> *M)* different groups of events. On top of this advantage you can also put a label to your phenotype in the related table entry (through The Edit Metadata tab). Unlike the previous strategy, you have to enter the phenotype description table (like in Table 4) in order to perform the merging action.

Like for the type 1 phenotype analysis, the various plots are available but they reported the phenotypes defined in the entered in the related metadata table (see Table 4 - The phenotype table).

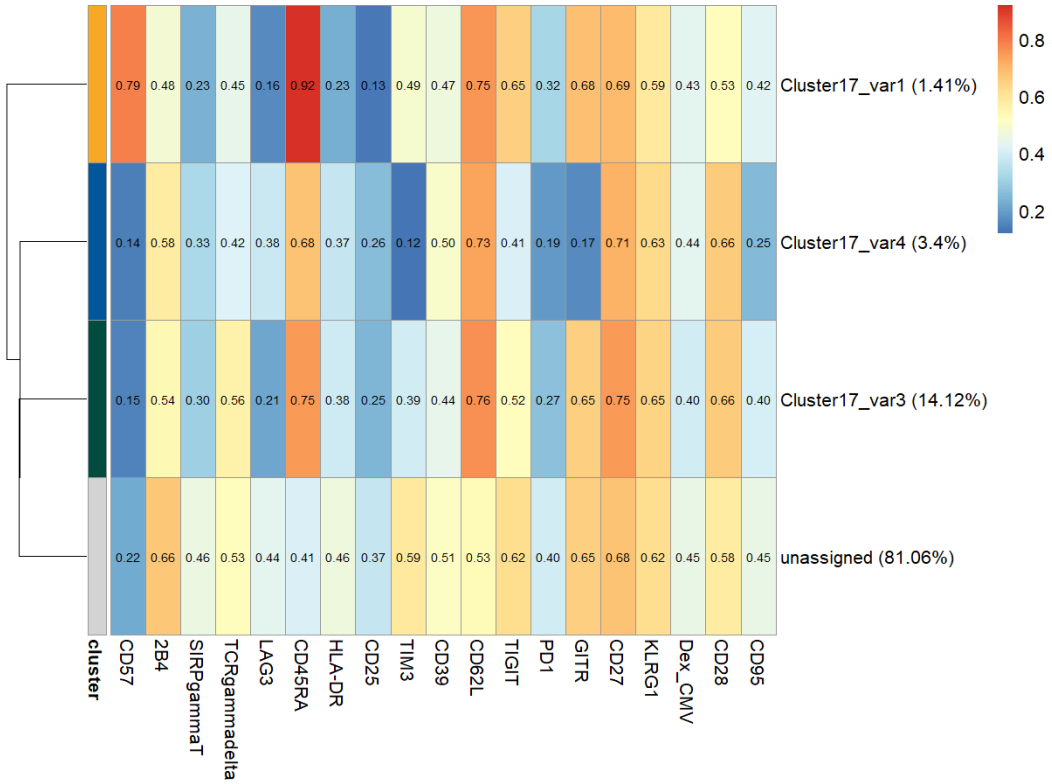


Figure 51 - The heatmap for the phenotype only

And the related tSNE map:

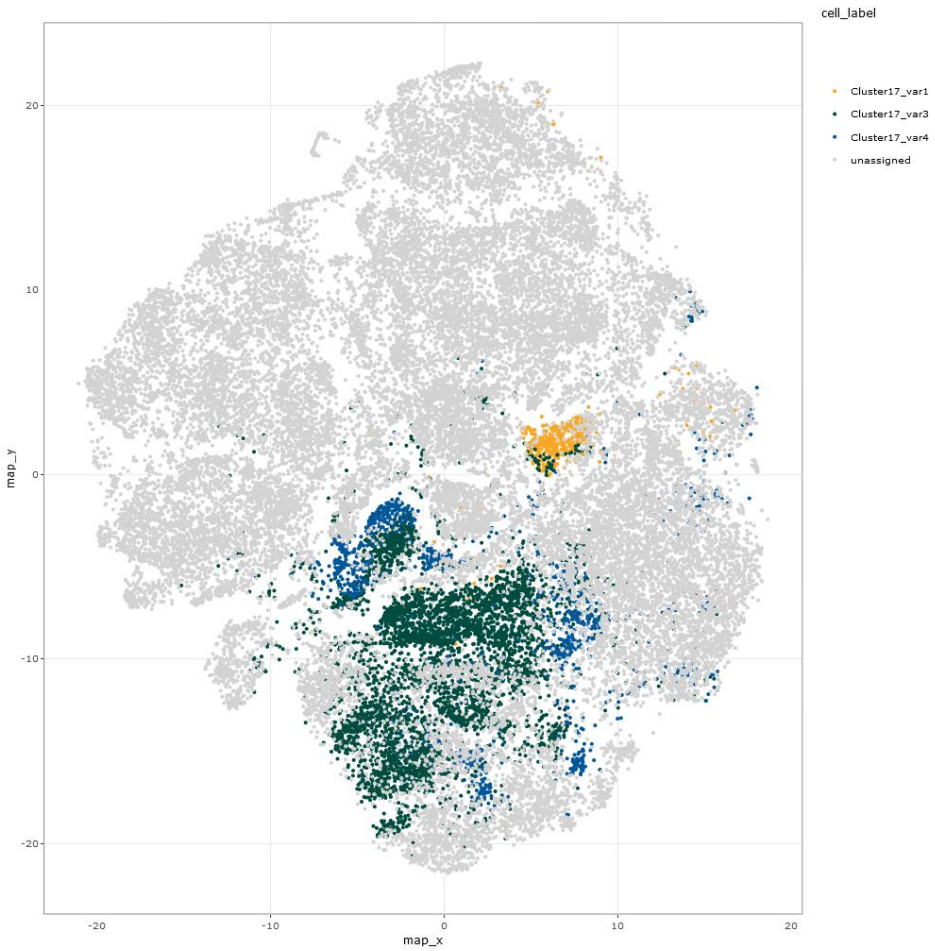


Figure 52 - The tSNE map with the colored phenotype events

Notice the green points which represent events which cannot be assigned to a defined phenotype.

7.6 Phenotype analysis – Type 3: Clustering + labelling + Meta-clustering

This is the most complete analysis in which it is possible to join both clustering and meta-clustering. In this case can be significant to analyze the following composite heat-map type like the one reported in Figure 53. Like in the previous analysis strategy you have to enter the phenotype table first. In this way both clusters and meta-clusters phenotype will be named as per your edited table.

This last and most complete strategy could be more robust in terms of results on the single meta-cluster. In any case bear in mind that the additional meta-clustering could reduce the number of recognized phenotypes. This is because the meta-clusters are less than the clusters and so it is less frequent the matching of the phenotype +/- mask (see Table 4 - The phenotype table) in the related meta-cluster entries:

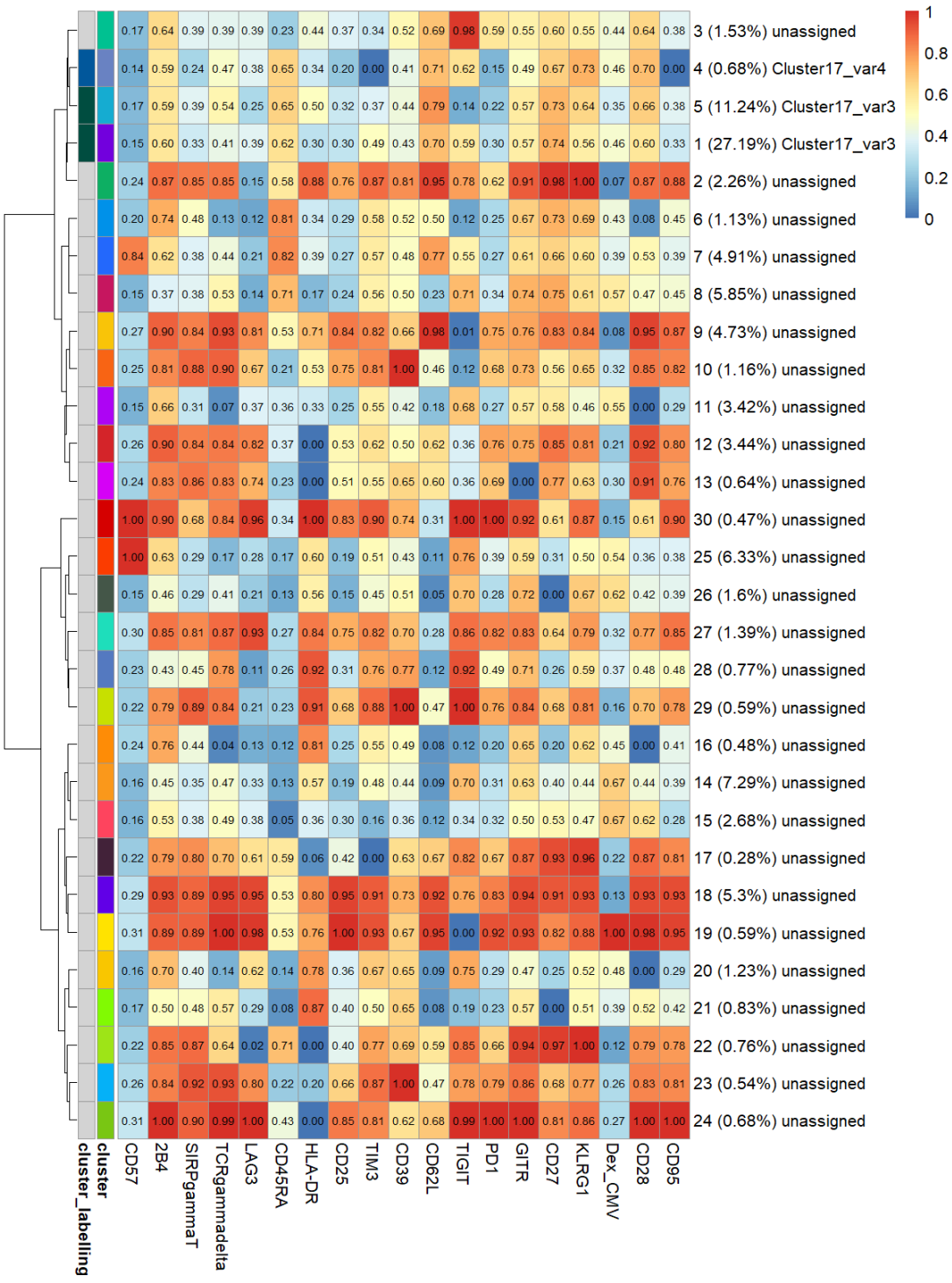


Figure 53 - The phenotype heat-map with the meta-cluster merging

	CD57	2B4	SIRPgam	TCRgamr	LAG3	CD45RA	HLA-DR	CD25	TIM3	CD39	CD62L	TIGIT	PD1	GITR	CD27	KLRG1	Dex_CM	CD28	CD95	cluster	cluster_labeld
1	0.15	0.60	0.33	0.41	0.39	0.62	0.30	0.30	0.49	0.43	0.70	0.59	0.30	0.57	0.74	0.56	0.46	0.60	0.33	1	Cluster17_var3
2	0.24	0.87	0.85	0.85	0.15	0.58	0.88	0.76	0.87	0.81	0.95	0.78	0.62	0.91	0.98	1.00	0.07	0.87	0.88	2	unassigned
3	0.17	0.64	0.39	0.39	0.39	0.23	0.44	0.37	0.34	0.52	0.69	0.98	0.59	0.55	0.60	0.55	0.44	0.64	0.38	3	unassigned
4	0.14	0.59	0.24	0.47	0.38	0.65	0.34	0.20	0.00	0.41	0.71	0.62	0.15	0.49	0.67	0.73	0.46	0.70	0.00	4	Cluster17_var4
5	0.17	0.59	0.39	0.54	0.25	0.65	0.50	0.32	0.37	0.44	0.79	0.14	0.22	0.57	0.73	0.64	0.35	0.66	0.38	5	Cluster17_var3
6	0.20	0.74	0.48	0.13	0.12	0.81	0.34	0.29	0.58	0.52	0.50	0.12	0.25	0.67	0.73	0.69	0.43	0.08	0.45	6	unassigned
7	0.84	0.62	0.38	0.44	0.21	0.82	0.39	0.27	0.57	0.48	0.77	0.55	0.27	0.61	0.66	0.60	0.39	0.53	0.39	7	unassigned
8	0.15	0.37	0.38	0.53	0.14	0.71	0.17	0.24	0.56	0.50	0.23	0.71	0.34	0.74	0.75	0.61	0.57	0.47	0.45	8	unassigned
9	0.27	0.90	0.84	0.93	0.81	0.53	0.71	0.84	0.82	0.66	0.98	0.01	0.75	0.76	0.83	0.84	0.08	0.95	0.87	9	unassigned
10	0.25	0.81	0.88	0.90	0.67	0.21	0.53	0.75	0.81	1.00	0.46	0.12	0.68	0.73	0.56	0.65	0.32	0.85	0.82	10	unassigned
11	0.15	0.66	0.31	0.07	0.37	0.36	0.33	0.25	0.55	0.42	0.18	0.68	0.27	0.57	0.58	0.46	0.55	0.00	0.29	11	unassigned
12	0.26	0.90	0.84	0.84	0.82	0.37	0.00	0.53	0.62	0.50	0.62	0.36	0.76	0.75	0.85	0.81	0.21	0.92	0.80	12	unassigned
13	0.24	0.83	0.86	0.83	0.74	0.23	0.00	0.51	0.55	0.65	0.60	0.36	0.69	0.00	0.77	0.63	0.30	0.91	0.76	13	unassigned
14	0.16	0.45	0.35	0.47	0.33	0.13	0.57	0.19	0.48	0.44	0.09	0.70	0.31	0.63	0.40	0.44	0.67	0.44	0.39	14	unassigned
15	0.16	0.53	0.38	0.49	0.38	0.05	0.36	0.30	0.16	0.36	0.12	0.34	0.32	0.50	0.53	0.47	0.67	0.62	0.28	15	unassigned
16	0.24	0.76	0.44	0.04	0.13	0.12	0.81	0.25	0.55	0.49	0.08	0.12	0.20	0.65	0.20	0.62	0.45	0.00	0.41	16	unassigned
17	0.22	0.79	0.80	0.70	0.61	0.59	0.06	0.42	0.00	0.63	0.67	0.82	0.67	0.87	0.93	0.96	0.22	0.87	0.81	17	unassigned
18	0.29	0.93	0.89	0.95	0.95	0.53	0.80	0.95	0.91	0.73	0.92	0.76	0.83	0.94	0.91	0.93	0.13	0.93	0.93	18	unassigned
19	0.31	0.89	0.89	1.00	0.98	0.53	0.76	1.00	0.93	0.67	0.95	0.00	0.92	0.93	0.82	0.88	1.00	0.98	0.95	19	unassigned
20	0.16	0.70	0.40	0.14	0.62	0.14	0.78	0.36	0.67	0.65	0.09	0.75	0.29	0.47	0.25	0.52	0.48	0.00	0.29	20	unassigned
21	0.17	0.50	0.48	0.57	0.29	0.08	0.87	0.40	0.50	0.65	0.08	0.19	0.23	0.57	0.00	0.51	0.39	0.52	0.42	21	unassigned
22	0.22	0.85	0.87	0.64	0.02	0.71	0.00	0.40	0.77	0.69	0.59	0.85	0.66	0.94	0.97	1.00	0.12	0.79	0.78	22	unassigned
23	0.26	0.84	0.92	0.93	0.80	0.22	0.20	0.66	0.87	1.00	0.47	0.78	0.79	0.86	0.68	0.77	0.26	0.83	0.81	23	unassigned
24	0.31	1.00	0.90	0.99	1.00	0.43	0.00	0.85	0.81	0.62	0.68	0.99	1.00	1.00	0.81	0.86	0.27	1.00	1.00	24	unassigned
25	1.00	0.63	0.29	0.17	0.28	0.17	0.60	0.19	0.51	0.43	0.11	0.76	0.39	0.59	0.31	0.50	0.54	0.36	0.38	25	unassigned
26	0.15	0.46	0.29	0.41	0.21	0.13	0.56	0.15	0.45	0.51	0.05	0.70	0.28	0.72	0.00	0.67	0.62	0.42	0.39	26	unassigned
27	0.30	0.85	0.81	0.87	0.93	0.27	0.84	0.75	0.82	0.70	0.28	0.86	0.82	0.83	0.64	0.79	0.32	0.77	0.85	27	unassigned
28	0.23	0.43	0.45	0.78	0.11	0.26	0.92	0.31	0.76	0.77	0.12	0.92	0.49	0.71	0.26	0.59	0.37	0.48	0.48	28	unassigned
29	0.22	0.79	0.89	0.84	0.21	0.23	0.91	0.68	0.88	1.00	0.47	1.00	0.76	0.84	0.68	0.81	0.16	0.70	0.78	29	unassigned
30	1.00	0.90	0.68	0.84	0.96	0.34	1.00	0.83	0.90	0.74	0.31	1.00	1.00	0.92	0.61	0.87	0.15	0.61	0.90	30	unassigned

Table 7 – The phenotype mapping table for each meta-cluster

The various phenotype are assigned to the meta-cluster following the meta-data phenotype table (see Table 4)

cluster_name	CD57	2B4	SIRPgamn	TCRgamm	LAG3	CD45RA	HLA-DR	CD25	TIM3	CD39	CD62L	TIGIT	PD1	GITR	CD27	KLRG1	Dex_CMV	CD28	CD95
cluster_14	-	-	-	-	-	-	+	-	-	-	-	+	-	+	-	-	+	-	-
cluster_26	-	-	-	-	-	-	+	-	-	+	-	+	-	+	-	+	+	-	-
cluster_21	-	-	-	+	-	-	+	-	-	+	-	-	-	+	-	+	-	+	-
cluster_28	-	-	-	+	-	-	+	-	+	+	-	+	-	+	-	+	-	-	-
cluster_08	-	-	-	+	-	+	-	-	+	-	-	+	-	+	+	+	+	-	-
cluster_15	-	+	-	-	-	-	-	-	-	-	-	-	-	-	+	-	+	+	-
cluster_03	-	+	-	-	-	-	-	-	-	+	+	+	+	+	+	+	-	+	-
cluster_11	-	+	-	-	-	-	-	-	+	-	-	+	-	+	+	-	+	-	-
cluster_16	-	+	-	-	-	-	+	-	+	-	-	-	-	+	-	+	-	-	-
cluster_04	-	+	-	-	-	+	-	-	-	-	+	+	-	-	+	+	-	+	-
cluster_01	-	+	-	-	-	+	-	-	-	-	+	+	-	+	+	+	-	+	-
cluster_06	-	+	-	-	-	+	-	-	+	+	+	-	-	+	+	+	-	-	-
cluster_20	-	+	-	-	+	-	+	-	+	+	-	+	-	-	-	+	-	-	-
cluster_05	-	+	-	+	-	+	-	-	-	-	+	-	-	+	+	+	-	+	-
cluster_29	-	+	+	+	-	-	+	+	+	+	-	+	+	+	+	+	-	+	+
cluster_22	-	+	+	+	-	+	-	-	+	+	+	+	+	+	+	+	-	+	+
cluster_02	-	+	+	+	-	+	+	+	+	+	+	+	+	+	+	+	-	+	+
cluster_23	-	+	+	+	+	-	-	+	+	+	-	+	+	+	+	+	-	+	+
cluster_13	-	+	+	+	+	-	-	+	+	+	+	-	+	-	+	+	-	+	+
cluster_12	-	+	+	+	+	-	-	+	+	+	+	-	+	+	+	+	-	+	+
cluster_24	-	+	+	+	+	-	-	+	+	+	+	+	+	+	+	+	-	+	+
cluster_10	-	+	+	+	+	-	+	+	+	+	+	-	-	+	+	+	-	+	+
cluster_27	-	+	+	+	+	-	+	+	+	+	-	+	+	+	+	+	-	+	+
cluster_17	-	+	+	+	+	+	-	-	-	+	+	+	+	+	+	+	-	+	+
cluster_09	-	+	+	+	+	+	+	+	+	+	+	-	+	+	+	+	-	+	+
cluster_19	-	+	+	+	+	+	+	+	+	+	+	-	+	+	+	+	+	+	+
cluster_18	-	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	-	+	+
cluster_25	+	+	-	-	-	-	+	-	+	-	-	+	-	+	-	+	+	-	-
cluster_07	+	+	-	-	-	+	-	-	+	-	+	+	-	+	+	+	-	+	-
cluster_30	+	+	+	+	+	-	+	+	+	+	-	+	+	+	+	+	-	+	+

Table 8 - The phenotype assignment as per the phenotype meta-table

The details of the phenotype assignment are explained in the following chapter.

7.7 The Phenotype analysis in details

This particular feature is a step beyond, comparing other available workflows in literature (that is especially true for the already mentioned [the CyTOF workflow: differential discovery in high-throughput

high-dimensional cytometry datasets - BioC 2017 workshop[17] which was the main inspiration for this work). Details about the phenotype setting to perform the analysis of the expression abundances of the various marker selected are provided in The Edit Metadata tab.

This new feature has also been introduced inside the tool, making possible to interrogate different datasets for a common signature, defined a priori. This advancement allows to speedily compare datasets, even if acquired with different fluoro-chromes or with dissimilar machine settings, automatically retrieving similarities.

The possibility to define all the possible phenotype in terms of marker expression could lead the researcher to better picture the marker's interactions and to frame possible new cell lineages and their relation with certain subpopulation (even the smallest one).

Having already introduced the importance of editing the phenodata table (see - The phenotype table) for the definition of interesting sub-population of cells having a selected phenotype, let's go under the hood of the algorithm in terms of formula and implementation details.

Let's call *EM*, the subset of the *flowSet* expression matrix. The entry dataset (see note at page 7) is a flowSet which is a list of *N* flowFrame$_i$ (with $1 \leq i \leq N$). In order to allow the comparison of the different samples in the various dimensionality reduction maps, they must be aggregated (in flow cytometry jargon the *concatenated* verbs is often used) in a single flowFrame having an *EM* with $n \times m$ elements, where n is the total number of events and m is the subset of selected markers of interest, part of the whole marker set available in the entry flowSet (which could be the result of the optimization workflow).

Since the expression values and the related ranges of the various marker intensities can spread through unpredictable values (even after the transformation/scaling process) we need another transformation in order to have a 0 to 1 normalized ranges. To produce this we scale the marker expression of the *EM* matrix using the percentiles as a boundaries values. Typically for values around the 0.1%÷5% for the lower bound, 95%÷99.9% for the upper one, are the ones used and they can be set via *minQuantile* and *maxQuantile* parameters used also in meta-clustering. So for each EM column (each marker expression) we calculate the lower and the upper percentile and we apply the following formula:

$$em = \mathrm{t}\left(\frac{\mathrm{t}(\mathrm{EM}) - \mathrm{low\_b}}{\mathrm{up\_b} - \mathrm{low\_b}}\right)$$

Equation 4 – The Zero function normalization

Where *up_b* and *low_b* are the quintile values calculated for the selected percentile. To put it clear, for the *low_b* the question is: "which is the channel expression value where the *minQuantile* percent of events fall below it". As a final transformation step we force to 1 the values greater than 1 and to 0 the negative values: *($em \rightarrow em_{01}$)*[18]. Since the selected percentage for the percentile values are parametrized

[17] At time of writing the most updated version is available in https://bioconductor.org/packages/release/workflows/vignettes/cytofWorkflow/inst/doc/cytofWorkflow.html

[18] The transformation on the Equation 4 is very similar to the one used for the simple normalization transformation in Equation 3. The main difference here is the fixed boundaries *min(x)* and *max(x)* which

in the program and can be edit as desired, this coercion can be always tuned (limiting its effects keeping as close to the 0% the lower and to 100% the upper limit. With this, we obtain em_{01}, a transformed *EM* matrix with values ranging from 0 to 1.

With the clustering algorithm, every single event (a single row) of the *em* matrix is bounded to a cluster (and/or to a meta-cluster) identifier (an integer that can spread from 1 to *k* where *k* is the total number of cluster or meta-cluster) in practice we work with a data frame composed by a $[em_{01}][mc]$ matrix where mc is a single array of integers with the same number of rows (events) of the *em*, collecting the clusters or meta-cluster identities belonging to a set of $\{1, \dots, N_{mc}\}$ values.

We could consider different type of grouping values to pass form the $[em_{01}][mc]$ matrix to a reduced em_m matrix with a number of N_{mc} rows (each row is related to a single cluster or meta-cluster). The most used method for such a kind of grouping operation is the median, so for each cluster or meta-cluster value we calculate for each N_{mc}:

$$em_{med\,j} = median(em_{01}) | \forall j \in \{1, \dots, N_{mc}\}$$

Equation 5 – A single row of the em_{01med} matrix

where the median is calculated for each em_{01} row assigned to a certain cluster for each clustering identity $\in \{1, \dots, N_{mc}\}$. With this we reduce the *em* matrix to an em_{med} matrix of dimension $N_{mc} \times m$ in which every row is the normalized median value of each m (marker expression) calculated for each set of events having a certain clustering identity.

With this, to map the each of the em_{med} rows to a proper phenotype, we have to take into account the expression rules defined in a phenodata table in which each cell's phenotype is defined in terms of:

- + : positively expressed
- - : not expressed (negatively expressed)

To pursue this goal we simply define two[19] thresholds for the level of expression: one for the upper limit (*pos_thr*), and one for the lower limit (*neg_thr*). These thresholds will be typically set around 0.4 ÷ 0.6 (and of course, having the possibility to be parametrized within the app with a dedicated widget in the related tabs) to get:

$$em_{pheno} = [em_{med}] | \forall i,j = \begin{cases} 1\ if\ value > pos\_th \\ 0\ if\ value < neg\_th \end{cases}$$

Equation 6 – The threshold criteria

The em_{pheno} (which is still a *k x m* matrix) has a logical content in the sense that every element could be 1 (positively expressed aka TRUE) or 0 (not expressed aka FALSE). This new matrix collects the signature of each of cluster. With this, cytoChain allows the searching of a certain phenotype that matches the

---

are here replaced by the *low_b* and *up_b* providing a parameterizable solution in which *low_b = min(x)* and *up_b = max(x)* when *minQuantile = 0.0* (that is 0%) and *minQuantile = 1.0* (100%)

[19] A more sophisticated approach would be to define more thresholds instead of only two. Something like "+ +", "+", "-"and "- -"could introduce other expression levels with "+ +" denoting "overexpressed" and "+" as "positively" express. This could be evaluated as in improvement in some future releases of the app.

combination of entries of the phenodata table (where every + should correspond to a "1", while every – should match a "0" in em_{pheno}).

The thresholds parameters (together with the two defined before for the em_{01} normalized matrix) defines the sensitivity on the phenotype assignments. In fact, the more the thresholds are far from the 0.5 midrange, the less is the number of phenotype that will be assigned to each meta-cluster (in practice a bigger *pos_th – neg_th* difference will limit the assignment because a wider *pos_th ÷ neg_th* range would limit the two upper and lower em_{med} thresholds[20]).

As you may notice this last assumption provide a rule of thumb to assign a certain group of events (either a cluster or a meta-cluster) to a certain phenotype. This assignment mechanism can be verified in details in the heat map example shown in Figure 46 and in the following Table 5 – The phenotype mapping table for each meta-cluster and Table 6 - The phenotype assignment as per the phenotype meta-table which in turns are related to the Table 4 - The phenotype table.

Actually each single marker expression could have a different distribution and the allocation of a certain phenotype could not depends on a common single threshold for each marker (even if customizable with the two - positive and negative - thresholds). A more precise and rigorous method could be to assign the proper threshold for each single marker and it will be implemented in next releases of cytoChain.

All in all a, cluster (or a meta-cluster) with a high/low average expression for a certain marker (especially if selected with a high threshold) has a high probability to be filled by events which are truly described by the phenotype table.

In any case, this criteria could also apply to build a phenotype table from the scratch: in other terms it is possible to use the same criteria depicted through the Equation 5 and the Equation 6 to determine whether a certain cluster most likely collects the kind of event expressing the same type of phenotype. With this, we can assume to assign a certain expression profile to define a generic phenotype for the homogenous group of events collected by the chosen clustering algorithm (see The different strategies for the phenotype analysis).

7.8 On the phenotype analysis, the labelling and the related parameters

The selected clustering algorithm group the various events in K_c (with K_c= number of clusters) elements. The meta-clustering, in turn, acts upon the clustering to further classify the clustered elements and to reduce them further in a K_M (with K_M= number of meta-clusters with $K_M < K_c$) elements.

In any case, the main results of this clustering and meta-clustering algorithms are the heat-maps (like to ones shown in Figure 36 and Figure 39). In each of the three type of analysis described in the previous paragraphs (7.2, 7.3 and 7.4) the heat-maps show the values obtained from the clustering (or meta-clustering) handled with the selected type of criteria (average, mcquitty, median, centroid) and the selected type of distance calculation method ("Euclidean", "maximum", "manhattan", "camberra", "binary" and "minkowski"). The heat-map values are the basis for the definition of the phenotype

[20] E.g. a *0.7 ÷ 0.3* range would limit the "+" assignment for those $em_{med} > 0.7$ and the "-" assignment for those $em_{med} < 0.3$. A cluster with $0.3 < em_{med} < 0.7$ will be "unknown" for the selected marker.

assignments what here is called “labelling”. The phenotype tables (like Table 4 and Table 8) are the results of this labelling operations. Along with the criteria and the distance calculation method, other two important pairs of parameters may affect this labelling operation and they can be used to tune the way the phenotype are eventually classified.

The first pair is composed by the *minQuantile/maxQuantile* that is the lower/upper limits for the zero transformation. These two thresholds (which we recommend to set them symmetrically like for the default setting: *minQuantile* = 0.05 and *maxQuantile* = 0.95) affect the “zero” transformation (see 5.2) and are explained in 7.5 together with the other pair of relevant parameters: the *threshold for the positive/negative MFI* values.

As the ranges of marker intensities can vary substantially, the *minQuantile/maxQuantile* pair fixes the low and the high percentiles as a boundary to force the marker MFI to 0 or to 1 (for the *maxQuantile*). This additional transformation of the *arcsinh*-transformed data can give better representation of relative differences in marker expression between annotated cell populations. This effect can be undone fixing the values to 0 (for the *minQuantile*) or to 1 (for the *maxQuantile*).

The other two parameters can be very sensitive in order to be better constrained the criteria in which the calculated normalized MFI can be greater/less than the fixed related parameters. By default a phenotype is considered “+” (aka positively expressed) for certain marker if the normalized MFI is greater than 0.5 and considered “-” (aka negatively expressed) if the normalized MFI is inferior than 0.5. With a more stringent threshold (e.g: 0.6 and 0.4) you will identify less cluster (or meta-cluster) as belonging to a certain phenotype: you will have more “undetermined” type of cells, but the ones which can be assigned are more likely to be truly positive (or truly negative)

7.9 Tip and tricks of the multidimensional analysis

Most of the good practice can only be the consequence of several analysis but some recommendations can be listed here. Clustering is the essential type of unsupervised learning analysis in every intelligent machine process. To achieve a good performance in the clustering algorithms it is important to recall some basic rules or betters some DDI (D*on’t Do It!*) suggestions.

Sometimes a sample is used just for control or verification purpose inside the panel. Check the various marker expressions (the best snapshot of the whole marker distribution is inside the **transformation** tab of the **optimization workflow**). Try to verify the experiment with your expectation and to understand the reason of such behavior.

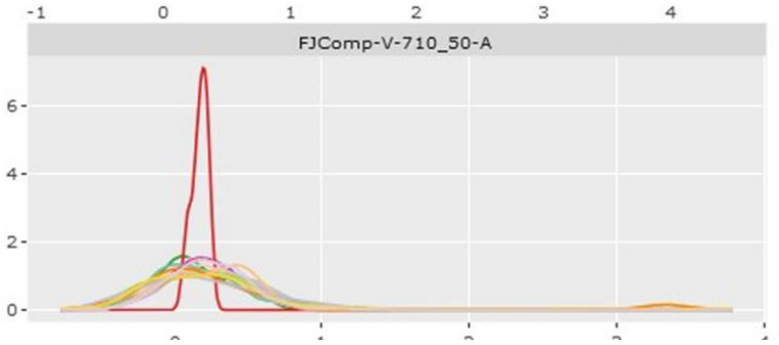


Figure 54 - The MFI expression of one channel – A spike is present for one sample only

Another important check is on the complete set of the various maps (this can be found in the **Mapping** tab of the **High dimensional analysis workflow**). With this maps you can check which part of the map is more or less in expressed for each single channels as shown in the following Figure 48.

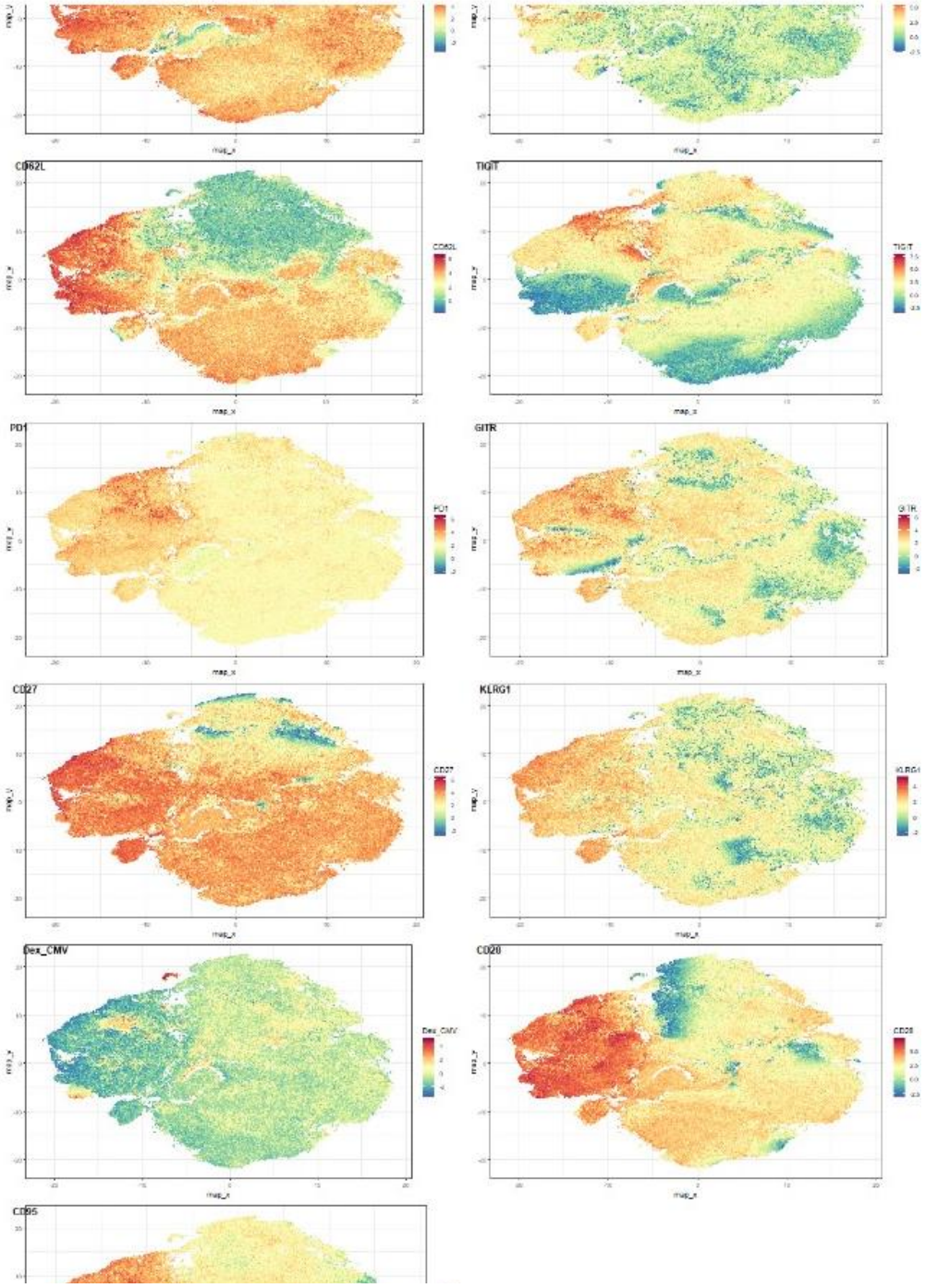


Figure 55 – An extraction of a complete set of tSNE map for each single marker expression

As you can see in this example reported in Figure 48 every map is well different from the other in terms of marker expression and most of all, the red (high expression) regions and the blue regions (low expression) are well isolated in big portions of the maps.

The situation of the following marker is not that clear instead:

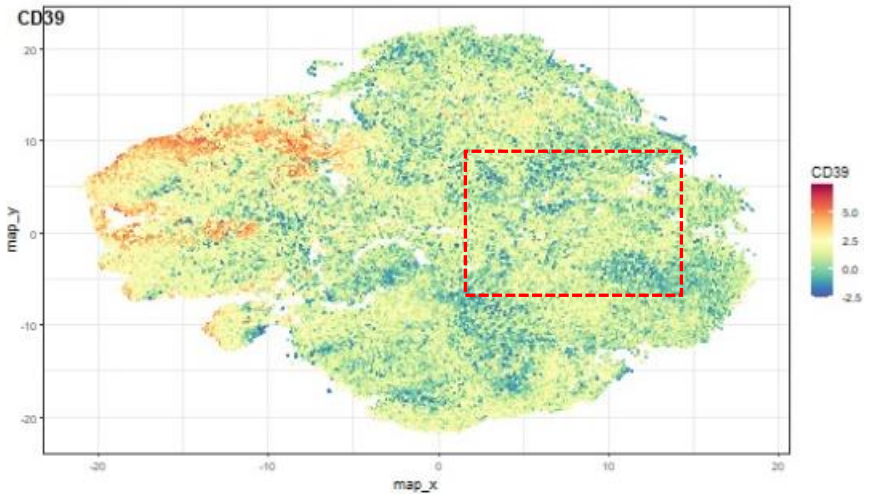


Figure 56 - The expression map for one channel - the pointillism effect is quite evident

As you can see the "pointillisme" which is typical of a smeared channel when every single event seems to have a density of expression different from the one immediately nearby. It is a typical case in which the expression of the marker is quite stable across all events around the 50% of the median expression. This phenomenon brings the heat-map to show equally + and – clusters with a phenotype doubling effect. For this reason try to not select the marker which shows such behavior for your analysis.

An even clearer situation is when the expression is stable like in the maps below:

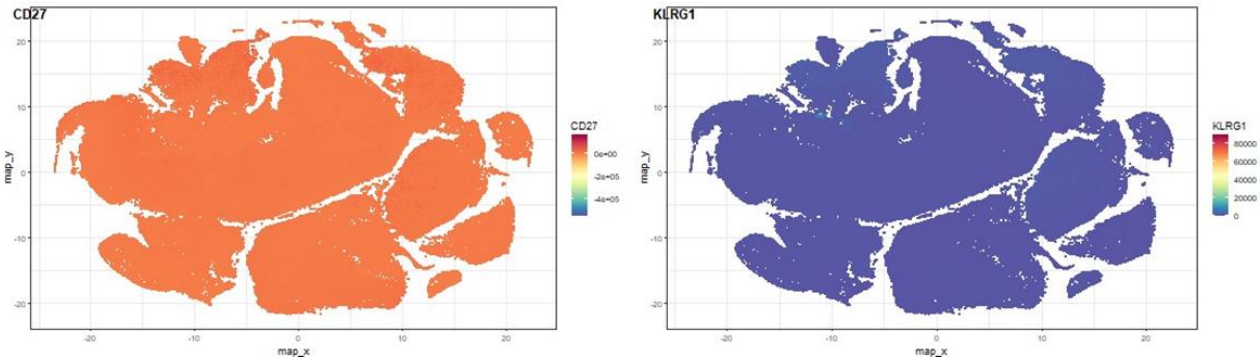


Figure 57 - Flat expression maps

Also in this case try to avoid to consider markers with this type of behavior. After all, also if a marker is always positive or always negative expressed, what is used for? (0 information and it can be dangerous).

7.10 On the auto-analysis button

The “auto-analysis” button is introduced for convenience, to automatically produce all the relevant data results and plots in a single zipped file to be downloaded.

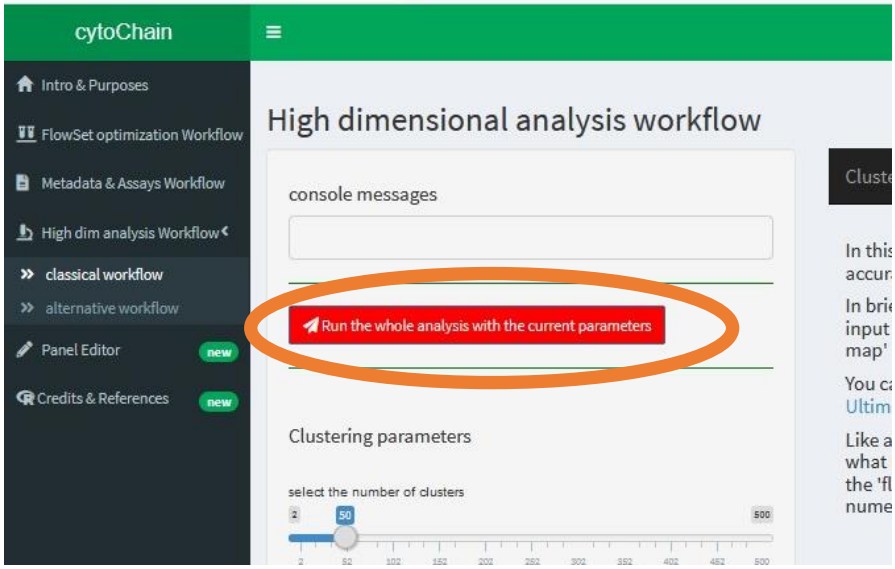


Figure 58 - The auto-analysis button

PROS: it avoids the downloading of every single table and plot from the various “high dimensional analysis workflow” tabs (at the end of the elaboration you can download all at once)

CONS: no interactivity (also the plots are not interactives) - all analysis are conducted using the current settings. The plots are not interactive as they are produced in the “manual” analysis investigation through the various page tabs (phases) and not every report available in the on-line version (in particular the ones generated in the **Metadata & Assays workflow)** are produced inside the zipped downloadable file.

The following is the table of the naming conventions of the produced results as they are listed within the zipped file to be downloaded.

file name	**Content**	**workflow**
*_MDS.png	multi-dimensional scaling plots (one for the samples and one for each single tag)	common
PCA.png	PCA plot	common
PCA.csv	data frame of the values related to the first two principal components	common
Plot*.png	flowSOM plots (summary and details: one for each selected marker)	common
HM_full_label.*	heatmap plot and related table of the clustering	wf2&wf3
pheno_full_label.csv	table which collects all the different phenotype found by the clustering algorithm	wf2&wf3
HM_meta.*	heatmap plot and related table of the meta-clustering	wf1&wf3
pheno_meta.csv	table which collects all the different phenotype found after the meta-clustering	wf1&wf3

map_metacluster.png	meta-cluster related map (with meta-cluster color)	wf1&wf3f3
marker_map	a collection of density map (in a two column grid: one plot for each selected marker)	common
map_cluster.fcs	a concatenated dataframe with the additional entry: sample_id, cluster (and meta-cluster)	common
HM_label_wf3.*	heatmap plot and related table of the meta-clustering and found phenotype	wf3
HM_pheno_wf3.*	heatmap plot and related table details of found phenotype	wf3
map_*_wf*.png	map plot and found phenotype	common
df_*_wf*.csv	quantity table for each tag	wf2&wf3
bar_*_wf*.png	bar plot for each tag	wf2&wf3
median_*_wf*.png	median plot for each tag	wf2&wf3

Table 9 - The file naming convention of the data analysis

Where:

wf#: stands for workflow nr.#

common: stands for wf1&wf2&wf3

and the **wf2** and **wf3** related files are produced only if the phenotype meta-table is entered

Please notice that the wf2 and wf3 files which are related to the worflow2 and worflow3 analysis are produced only if the phenotype table is correctly loaded for the analysis.

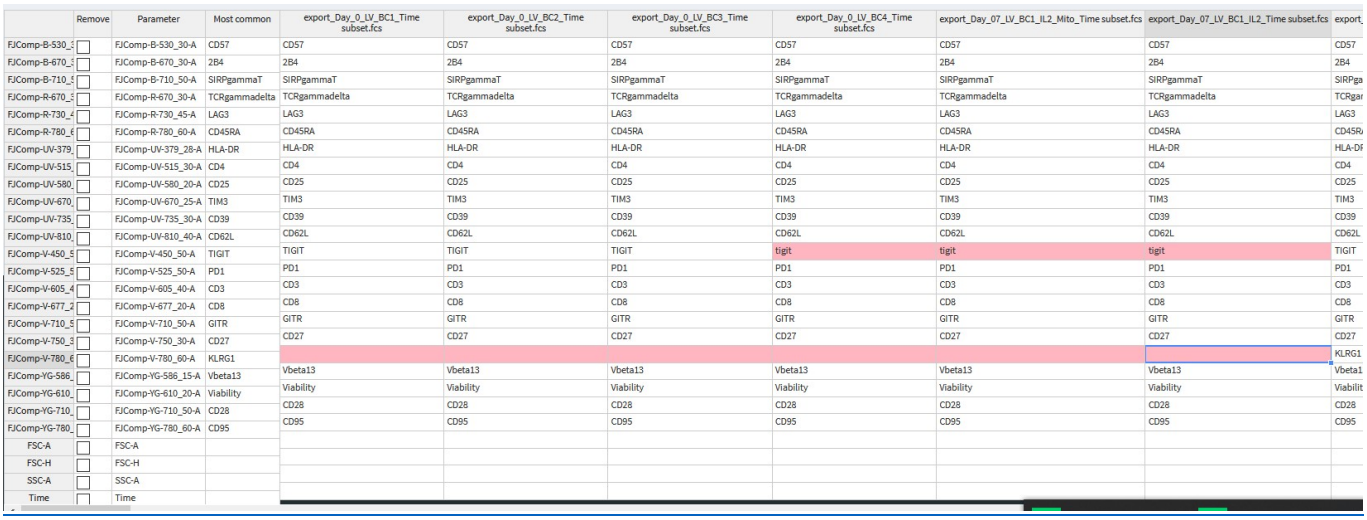
7.11 The Panel Editor

Discrepancies within the samples could happen very often. Think all the cases in which the facility where you access the machinery is open to different groups which can manipulate the flow cytometer configuration and most of all the panel setting, or when you collect the samples in different sessions.

The samples collected then could have different marker names and/or marker description even if the actual panel has not been touched at all and this could represent a real problem for cytoChain (or for any other kind of dedicated software) which can only handle “pure” flowSet, namely a set of flowFrames (aka sample, aka tube, aka fcs file…) in which every single element has the same number of marker (aka stain, aka dimension, aka channel) otherwise it belongs to another flowSet.

For this reason cytoChain implements the *Panel Editor,* a simple way to handle the marker’s names/description and to fix such issue. You can also remove the markers, in case you need to.

Thanks to Pier Federico Gherardini for this idea from *premessa* package, see: SCAFFoLD



	Remove	Parameter	Most common	export_Day_0_LV_BC1_Time subset.fcs	export_Day_0_LV_BC2_Time subset.fcs	export_Day_0_LV_BC3_Time subset.fcs	export_Day_0_LV_BC4_Time subset.fcs	export_Day_07_LV_BC1_IL2_Mito_Time subset.fcs	export_Day_07_LV_BC1_IL2_Time subset.fcs	export_
FJComp-B-530_3	☐	FJComp-B-530_30-A	CD57	CD57	CD57	CD57	CD57	CD57	CD57	CD57
FJComp-B-670_3	☐	FJComp-B-670_30-A	2B4	2B4	2B4	2B4	2B4	2B4	2B4	2B4
FJComp-B-710_5	☐	FJComp-B-710_50-A	SIRPgammaT	SIRPgammaT	SIRPgammaT	SIRPgammaT	SIRPgammaT	SIRPgammaT	SIRPgammaT	SIRPga
FJComp-R-670_3	☐	FJComp-R-670_30-A	TCRgammadelta	TCRgammadelta	TCRgammadelta	TCRgammadelta	TCRgammadelta	TCRgammadelta	TCRgammadelta	TCRgar
FJComp-R-730_4	☐	FJComp-R-730_45-A	LAG3	LAG3	LAG3	LAG3	LAG3	LAG3	LAG3	LAG3
FJComp-R-780_6	☐	FJComp-R-780_60-A	CD45RA	CD45RA	CD45RA	CD45RA	CD45RA	CD45RA	CD45RA	CD45R
FJComp-UV-379_	☐	FJComp-UV-379_28-A	HLA-DR	HLA-DR	HLA-DR	HLA-DR	HLA-DR	HLA-DR	HLA-DR	HLA-DI
FJComp-UV-515_	☐	FJComp-UV-515_30-A	CD4	CD4	CD4	CD4	CD4	CD4	CD4	CD4
FJComp-UV-580_	☐	FJComp-UV-580_20-A	CD25	CD25	CD25	CD25	CD25	CD25	CD25	CD25
FJComp-UV-670_	☐	FJComp-UV-670_25-A	TIM3	TIM3	TIM3	TIM3	TIM3	TIM3	TIM3	TIM3
FJComp-UV-735_	☐	FJComp-UV-735_30-A	CD39	CD39	CD39	CD39	CD39	CD39	CD39	CD39
FJComp-UV-810_	☐	FJComp-UV-810_40-A	CD62L	CD62L	CD62L	CD62L	CD62L	CD62L	CD62L	CD62L
FJComp-V-450_5	☐	FJComp-V-450_50-A	TIGIT	TIGIT	TIGIT	tigit	tigit	tigit	TIGIT	
FJComp-V-525_5	☐	FJComp-V-525_50-A	PD1	PD1	PD1	PD1	PD1	PD1	PD1	PD1
FJComp-V-605_4	☐	FJComp-V-605_40-A	CD3	CD3	CD3	CD3	CD3	CD3	CD3	CD3
FJComp-V-677_2	☐	FJComp-V-677_20-A	CD8	CD8	CD8	CD8	CD8	CD8	CD8	CD8
FJComp-V-710_5	☐	FJComp-V-710_50-A	GITR	GITR	GITR	GITR	GITR	GITR	GITR	GITR
FJComp-V-750_3	☐	FJComp-V-750_30-A	CD27	CD27	CD27	CD27	CD27	CD27	CD27	CD27
FJComp-V-780_6	☐	FJComp-V-780_60-A	KLRG1							KLRG1
FJComp-YG-586_	☐	FJComp-YG-586_15-A	Vbeta13	Vbeta13	Vbeta13	Vbeta13	Vbeta13	Vbeta13	Vbeta13	Vbeta1
FJComp-YG-610_	☐	FJComp-YG-610_20-A	Viability	Viability	Viability	Viability	Viability	Viability	Viability	Viabilit
FJComp-YG-710_	☐	FJComp-YG-710_50-A	CD28	CD28	CD28	CD28	CD28	CD28	CD28	CD28
FJComp-YG-780_	☐	FJComp-YG-780_60-A	CD95	CD95	CD95	CD95	CD95	CD95	CD95	CD95
FSC-A	☐	FSC-A								
FSC-H	☐	FSC-H								
SSC-A	☐	SSC-A								
Time	☐	Time								

Figure 59 - The flowSet report as it is shown by the Panel Editor

8 A complete clustering analysis – an example

t.b.d

9 Conclusion

These are the main benefits of the cytoChain tool and what this web-app could offer:

- **Ease of use**: Platform- and technology-independency through SaaS (Software as a Service web-app). No problems due to any software installation, operating systems… It needs just a web-browser and the input samples to analyze.
- **Enhance productivity**: All major analysis blocks available in a unique integrated environment (optimization flowSet – data fitness – clustering – meta-clustering)
- **Completeness**: Especially suitable for iterating your analysis, tuning the different processes through all the parameters, controlling every aspect of the whole process
- **Up to date & openness**: Everlasting improvement with the all the best in breed algorithms, graphics and features setting (also borrowed from genetics and other application fields). All suggestions are welcome in order to improve the results.
- **Modularity**: Pursue your analysis steps by step, passing from one workflow to another but always getting your output with no hassle
- **Multi-level analysis**: Integrated experimental subgroups through metadata, condensing useful info in a simple output
- **Discovery-oriented**: Conceived to go beyond clustering, discovering new features and key points of your datasets
- **Reproducibility** – by entering a common seed setting in each used algorithm

10 Bibliography

Leland McInnes, John Healy, James Melville. 2018. "UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction." arXiv.org.

Lukas M. Weber, Mark D. Robinson. 2016. "Comparison of Clustering Methods for High-Dimensional Single-Cell Flow and Mass Cytometry Data." *Cytometry.*

Maaten, Laurens van der. 2008. "Visualizing Data using t-SNE." *Journal of Machine Learning Research.*

Sean C. Bendall, 1,* Erin F. Simonds,1,* Peng Qiu,2 El-ad D. Amir,3 Peter O. Krutzik,1 Rachel Finck,1 Robert V. Bruggner,1,7 Rachel Melamed,3 Angelica Trejo,1 Olga I. Ornatsky,4,5 Robert S. Balderas,6 Sylvia K. Plevritis,2 Karen Sachs,1 Dana Pe'er,3 Scott. 2011. "Single-Cell Mass Cytometry of Differential Immune and Drug Responses Across a Human Hematopoietic Continuum." *Science.*

11 APPENDIX

Here you can find the list of packages which are used in the various parts of the citoChain code

Some of them constitute the sources of inspiration for this work. I should like to offer my heartfelt thanks to all of the entire community of reasearcher and in particular to Malgorzata Nowicka, Carsten Krieg, Lukas M. Weber, Felix J. Hartmann, Silvia Guglietta, Burkhard Becher, Mitchell P. Levesque, Mark D. Robinson for their workflow (see "CyTOF workflow: differential discovery in high-throughput high-dimensional cytometry datasets. BioC 2017 workshop" http://bioconductor.org/help/course-materials/2017/BioC2017/Day2/Workshops/CyTOF/doc/cytofWorkflow_BioC2017workshop.html#data-import) and all the developers and bioinformatics who makes available some of their precious software contribution.

Here the list of all the packages focusing on the different areas of the application. In particular for flow cytometry data manipulation, clustering, pre-clustering and samples assays:

for **flowCore**: B Ellis, Perry Haaland, Florian Hahne, Nathan Le Meur, Nishant Gopalakrishnan, Josef Spidlen, Mike Jiang and Greg Finak (2019). flowCore: flowCore: Basic structures for flow cytometry data. R package version 1.50.0

for **ncdfFlow**: Mike Jiang,Greg Finak,N. Gopalakrishnan (2019). ncdfFlow: ncdfFlow: A package that provides HDF5 based storage for flow cytometry data. R package version 2.30.1.

for **flowAI**: Monaco,G. et al. (2016) flowAI: automatic and interactive anomaly discerning tools for flow cytometry data. Bioinformatics. 2016 Aug 15;32(16):2473-80

for **flowStats**: Florian Hahne, Nishant Gopalakrishnan, Alireza Hadj Khodabakhshi, Chao-Jen Wong and Kyongryun Lee (2019). flowStats: Statistical methods for the analysis of flow cytometry data. R package version 3.40.1. http://www.github.com/RGLab/flowStats

for **spade**: M. Linderman, P. Qiu, E. Simonds and Z. Bjornson (2018). spade: SPADE -- An analysis and visualization tool for Flow Cytometry. R package version 1.10.4. http://cytospade.org

for **limma**: Ritchie, M.E., Phipson, B., Wu, D., Hu, Y., Law, C.W., Shi, W., and Smyth, G.K. (2015). limma powers differential expression analyses for RNA-sequencing and microarray studies. Nucleic Acids Research 43(7), e47.

for **matrixStats**: Henrik Bengtsson (2018). matrixStats: Functions that Apply to Rows and Columns of Matrices (and to Vectors). R package version 0.54.0. https://CRAN.R-project.org/package=matrixStats

for **factoextra**: Alboukadel Kassambara and Fabian Mundt (2017). factoextra: Extract and Visualize the Results of Multivariate Data Analyses. R package version 1.0.5. https://CRAN.R-project.org/package=factoextra

for **DDoutlier**: Jacob H. Madsen (2018). DDoutlier: Distance & Density-Based Outlier Detection. R package version 0.1.0. https://CRAN.R-project.org/package=DDoutlier

for **mclust**: Scrucca L., Fop M., Murphy T. B. and Raftery A. E. (2016) mclust 5: clustering, classification and density estimation using Gaussian finite mixture models The R Journal 8, pp. 205-233

for **FlowSOM**: Sofie Van Gassen, Britt Callebaut and Yvan Saeys (2019). FlowSOM: Using self-organizing maps for visualization and interpretation of cytometry data. http://www.r-project.org, http://dambi.ugent.be

for **kohonen**: Wehrens R, Kruisselbrink J (2018). "Flexible Self-Organizing Maps in kohonen 3.0. "Journal of Statistical Software", *87*(7),1-18. doi:10.18637/jss.v087.i07 - https://doi.org/10.18637/jss.v087.i07

for **ConsensusClusterPlus**: Wilkerson, M.D., Hayes, D.N. (2010). ConsensusClusterPlus: a class discovery tool with confidence assessments and item tracking. Bioinformatics, 2010 Jun 15;26(12):1572-3.

for **Rphenograph**: Hao Chen (2015). Rphenograph: R implementation of the phenograph algorithm. R package version 0.99.1.

for **flowVS**: Ariful Azad (2019). flowVS: Variance stabilization in flow cytometry (and microarrays).

for **Rtsne**: L.J.P. van der Maaten and G.E. Hinton. Visualizing High-Dimensional Data Using t-SNE. Journal of Machine Learning Research 9(Nov):2579-2605, 2008. L.J.P. van der Maaten. Accelerating t-SNE using Tree-Based Algorithms. Journal of Machine Learning Research 15(Oct):3221-3245, 2014. Jesse H. Krijthe (2015). Rtsne: T-Distributed Stochastic Neighbor Embedding using a Barnes-Hut Implementation, URL: https://github.com/jkrijthe/Rtsne

for **umap**: Tomasz Konopka (2019). umap: Uniform Manifold Approximation and Projection. R package version 0.2.2.0. https://CRAN.R-project.org/package=umap

---

We cannot forget to mention all the effort made by members of the R (and Python) developers' community for all the features of such an indispensable set of accessory libraries:

for **DT**: Yihui Xie, Joe Cheng and Xianying Tan (2019). DT: A Wrapper of the JavaScript Library 'DataTables'. R package version 0.7. https://CRAN.R-project.org/package=DT

for **tidyverse**: Hadley Wickham (2017). tidyverse: Easily Install and Load the 'Tidyverse'. R package version 1.2.1. https://CRAN.R-project.org/package=tidyverse

for **lubridate**: Garrett Grolemund, Hadley Wickham (2011). Dates and Times Made Easy with lubridate. Journal of Statistical Software, 40(3), 1-25. URL http://www.jstatsoft.org/v40/i03/

for **plotly**: Carson Sievert (2018) plotly for R. https://plotly-r.com

for **RColorBrewer**: Erich Neuwirth (2014). RColorBrewer: ColorBrewer Palettes. R package version 1.1-2. https://CRAN.R-project.org/package=RColorBrewer

for **reshape2**: Hadley Wickham (2007). Reshaping Data with the reshape Package. Journal of Statistical Software, 21(12), 1-20. URL http://www.jstatsoft.org/v21/i12/

for **rhandsontable**: Jonathan Owen (2019). rhandsontable: Interface to the 'Handsontable.js' Library. R package version 0.3.7. http://jrowen.github.io/rhandsontable/

for **pheatmap**: Raivo Kolde (2018). pheatmap: Pretty Heatmaps. R package version 1.0.12

for **png**: Simon Urbanek (2013). png: Read and write PNG images. R package - https://CRAN.R-project.org/package=png

for **gridExtra**: Baptiste Auguie (2017). gridExtra: Miscellaneous Functions for 'Grid' Graphics. https://CRAN.R-project.org/package=gridExtra

for **streamgraph**: Bob Rudis (2015). streamgraph: streamgraph is an htmlwidget for building streamgraph visualizations. R package version 0.8.1. http://github.com/hrbrmstr/streamgraph

for **fS**: Jim Hester and Hadley Wickham (2020). fs: Cross-Platform File System Operations Based on 'libuv'. R package version 1.4.1. https://CRAN.R-project.org/package=fs

for **ggpubr**: Alboukadel Kassambara (2020). ggpubr: 'ggplot2' Based Publication Ready Plots. R package version 0.3.0. https://CRAN.R-project.org/package=ggpubr

---

and finally the shiny packages:

for **shiny**: Winston Chang, Joe Cheng, JJ Allaire, Yihui Xie and Jonathan McPherson (2019). shiny: Web Application Framework for R. R package version 1.3.2. https://CRAN.R-project.org/package=shiny

for **shinythemes**: Winston Chang (2018). shinythemes: Themes for Shiny. R package version 1.1.2. https://CRAN.R-project.org/package=shinythemes

for **shinydashboard**: Winston Chang and Barbara Borges Ribeiro (2018). shinydashboard: Create Dashboards with 'Shiny'. R package version 0.7.1. https://CRAN.R-project.org/package=shinydashboard

for **shinycssloaders**: Andras Sali (2017). shinycssloaders: Add CSS Loading Animations to 'shiny' Outputs. R package version 0.2.0. https://CRAN.R-project.org/package=shinycssloaders

for **shinyWidgets**: Victor Perrier, Fanny Meyer and David Granjon (2019). shinyWidgets: Custom Inputs Widgets for Shiny. R package version 0.4.8. https://CRAN.R-project.org/package=shinyWidgets

for **shinyBS**: Eric Bailey (2015). shinyBS: Twitter Bootstrap Components for Shiny. R package version 0.61. https://CRAN.R-project.org/package=shinyBS