Charles Meijer 5938244 Major research project final report 15-01-2022

Automated flow cytometry analysis using machine learning algorithm FlowSOM

Abstract

Flow cytometry has been instrumental in cell phenotyping for the past decades, but with more and more data being generated, it becomes increasingly difficult and time-consuming to keep track of all the proteins that are detected by the flow cytometer when clustering the cells through manual gating. To solve this issue, this study aims to create a pipeline for fully automated analysis of flow cytometry data, from reading the output of the flow cytometer to clustering the cells together by cell type. Various machine learning algorithms have been developed to perform the clustering of these cells, which will be the most important step in the pipeline. One promising algorithm with this purpose is FlowSOM. This "R" package aims to apply the Self-Organizing Map (SOM) clustering method to flow cytometry analysis, which allows for clustering, as well as easy visualisation of the data. It has also shown to be fast, as well as accurate, compared to other available methods. However, the output of the algorithm is sensitive to change in parameter values. For this study, FlowSOM was tested with various combinations of parameter settings, including varying random states, durations of training, and learning rates. However, analysis of the clusters created by FlowSOM with these different settings did not lead to a combination of parameter values which resulted in a consistent output. The consistency of the SOM quickly improves and then stagnates with minimal training, but further analysis must be performed to find how to set up the algorithm so that small parameter adjustments have minimal effect on its clustering output.

Layman's Summary

Flow cytometry is a technique used to identify the cell types of the many different cells present in a mixed sample, like our blood, by features like cell size, complexity, and the presence of specific proteins. It has formed the basis of many biomedical studies over the last decades and still does to this day. However, with increasingly large and complicated datasets becoming available, it becomes more difficult to keep an overview of all the information when deciding on the cell type of each individual cell. Therefore, the idea of using computer algorithms aiming to automate this process is gaining popularity in modern research and this study aims to use one of these algorithms to create a pipeline for fully automated analysis of flow cytometry data. FlowSOM is a clustering algorithm based on the Self-Organizing Map method, capable of quickly and accurately grouping cells of the same type together. However, the output of FlowSOM can be sensitive to change in the parameter setup. It is difficult to show improvement of the clustering after mere minutes of the SOM adapting to the dataset, and further analysis of the algorithm is required to find how it can be made more consistent and robust, so that it may be the basis of a fully automated flow cytometry analysis pipeline in the future.

Introduction

Flow cytometry, the technology to identify the individual cells in a mixed sample, like the blood or a colon biopsy, for example, has been a critical tool in cell research of the last decades in a wide variety of disciplines, from agriculture and plant breeding to virology and systems vaccinology (Ochatt,

2008)(Grifoni, *et al.*, 2020). The technique consists of having the cells, stained with various fluorescent compounds for specific proteins of interest, flow past one or more lasers in a fluidics system. There, multiple lasers and detectors are being used to gain information on the physical and chemical properties of the individual cells that flow through (Shapiro, 2005). The researcher uses these properties to group the cells into biologically distinct groups. This technique is called manual gating (McKinnon, 2018). The datasets created with cytometry techniques have drastically increased in size in the last few years. More and more cell markers are being analysed at the same time, which makes it increasingly difficult and time-consuming to keep track of them all during the gating process (Van Gassen, *et al.,* 2015). To solve this issue, this study aims to create a fully automated pipeline for flow cytometry analysis, including automation of pre-processing steps taken before clustering the data, and the clustering itself. A variety of flow cytometry clustering algorithms have become available, the most promising of which may be FlowSOM (Weber & Robinson, 2016). This R package is designed to apply the Self-Organizing Map unsupervised machine learning algorithm to flow cytometry data (Van Gassen, *et al.*, 2015). The general idea of the SOM algorithm is to create a grid consisting of a few clusters, or 'nodes', that acts as a representation of the millions of cells in the dataset. During the training process, this map of nodes will constantly be updated to better fit the dataset by having cells pull one or more nodes in its direction. After training, each cell will be grouped into the node that best matches the cell. The nodes then undergo metaclustering, during which different clustering methods, including SOM itself, may be used to cluster the nodes, containing several cells, together into a smaller number of groups, or 'metaclusters', that may represent the cell types that made up the mixed sample (Van Gassen, *et al.*, 2015). FlowSOM has shown to be fast and accurate compared to other available algorithms and it allows for easy visualisation of the clustered data, as the nodes can be plotted in a 2D grid. FlowSOM also has the advantage of using the entirety of the dataset, and the option to either manually or automatically decide on the number of metaclusters (Weber & Robinson, 2016). However, for the pipeline to be based around FlowSOM, it should be consistent, and resistant to small parameter changes, as it is not desirable for the output to change drastically if the training period is extended or shortened. For this reason, the consistency of FlowSOM with several combinations of parameter settings was investigated to find how to set up the algorithm to create consistent, reproducible results.

Methods

Data from two different datasets were clustered using FlowSOM with different combinations of parameter values (Mateus, *et al.,* 2020)(Grifoni, *et al.*, 2020). Both sets contained cells from human blood samples, taken from control groups, stained to detect various proteins commonly associated with the immune system. All the data has been processed before being clustered with the FlowSOM algorithm. The data has been compensated for the spectral overlap between different channels, cleared of debris and doublets, transformed, and scaled. Transformation of the data has been performed using the arcsinh transformation function available in the FlowVS R package (Azad, 2021). The option to scale the data is included in the FlowSOM package itself.

The similarity between SOM clusters was measured by calculating the average Jaccard index (AJI). For each node in one SOM, the most similar node in another SOM was found by calculating the Jaccard index, the number of cells shared between the nodes divided by the total amount of different cells in the two nodes combined, for each pair of nodes between two SOMs (Leicht, *et al.,* 2006). The mean of the highest Jaccard indices for each node was then calculated to represent the similarity between two full SOMs after training.

Results

The number of metaclusters varies with increasing rlen

The first FlowSOM parameter analysed for its effect on the algorithm's output was the number of iterations over the dataset performed by the SOM during its training process (rlen). As mentioned before, the SOM picks random cells to modify its grid nodes. For example, if a dataset contains 10 cells, and the rlen value is set to FlowSOM's default value of 10 as well, the SOM would update 10*10=100 times. The expectation was that the SOM grid would adapt gradually to the dataset. It should move rapidly at the start, as the grid nodes are moving toward the data. Then, as the SOM better represents the dataset, further training should change the grid less drastically. The SOM algorithm is designed to behave this way. The number of nodes updated by each data point and the SOM algorithm's learning rate decrease during training, which means that the influence of each extra datapoint used to adapt the map will diminish over time. However, analysing the total number of metaclusters automatically generated by running FlowSOM with different rlen values, showed a surprising pattern (Fig. 1). The number of metaclusters is quite unpredictable, with rlen values up to around 1000. When the rlen is increased further, the result becomes consistent, until an rlen value of circa 1750 is reached, from which point the number of metaclusters gets unpredictable again. This same pattern arose with each combination of initialization functions, metaclustering methods, and distance functions available within the FlowSOM R package.

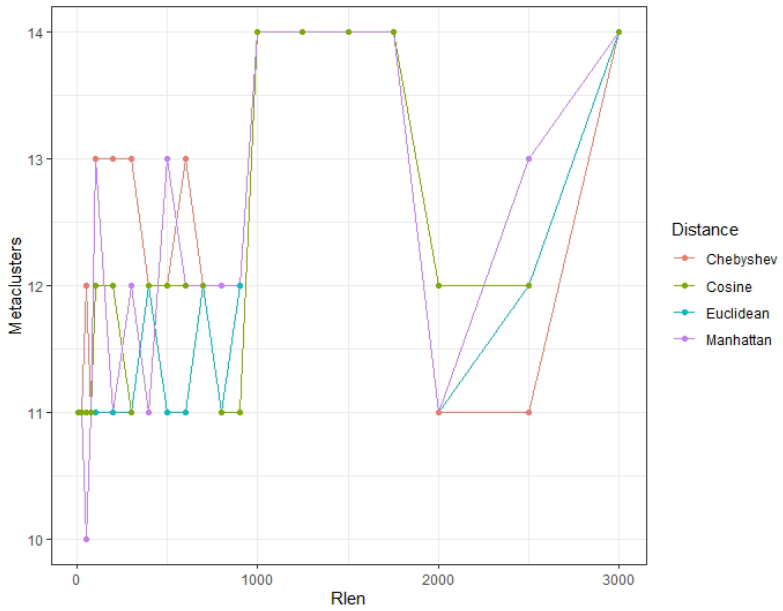

Figure 1. Amount of automatically generated metaclusters for different rlen values and distance metrics. Initialization = Random, Seed = 42, Metaclustering function = Consensus. (Data from Mateus, *et al.*, 2020)

Analysis of SOM output shows a surprising pattern containing perfect consistency

To try and find how this pattern may arise, a closer look was taken at the output of the SOM algorithm itself, so which cells end up in which nodes. AJI calculations were performed to determine the similarity between the finalised SOMs (Fig. 2a). These were generated with increasing rlen values, but the same seed value of 42, meaning that for each map, the random starting values of the nodes, as well as the order in which the cells are picked to update the SOM will be the same. The results indicate that the SOMs generated with rlen values ranging from 0 to 1000 show some similarity (AJI ≈0.75). Then, the SOMs that were generated with rlen set between 1000 and 1900 were all identical (AJI≈1), but showed little similarity to the ones generated with lower rlen values. Increasing the rlen beyond 1900 resulted in maps that were somewhat similar to each other and the maps generated with lower rlen values, but not to the SOMs in that consistent region between 1000 and 1900 rlen.

Also, when changing the seed variable from 42 to 28, the same pattern arises, but the SOM created consistently in that same region of rlen values between 1000 and 1900 would show little similarity to the one generated with the seed set to 42 earlier (Fig 2b). These observations hinted toward the underlying mechanism causing this outcome. In the C code called by the FlowSOM package to run the SOM algorithm, the variable holding the number of cycles of picking a datapoint and updating the grid specified by the user is limited by it being defined as a 32-bit integer. When the package tries to assign a value to this variable that exceeds the 32-bit integer limit, which occurs with an rlen value close to 1000 for this dataset, the variable will get a negative value. The variable being negative then causes the SOMs nodes not to be moved after their initialization. So the seemingly consistent outputs of FlowSOM were caused by the algorithm stopping directly after random initialization, which always gives the same results when using the same seed.

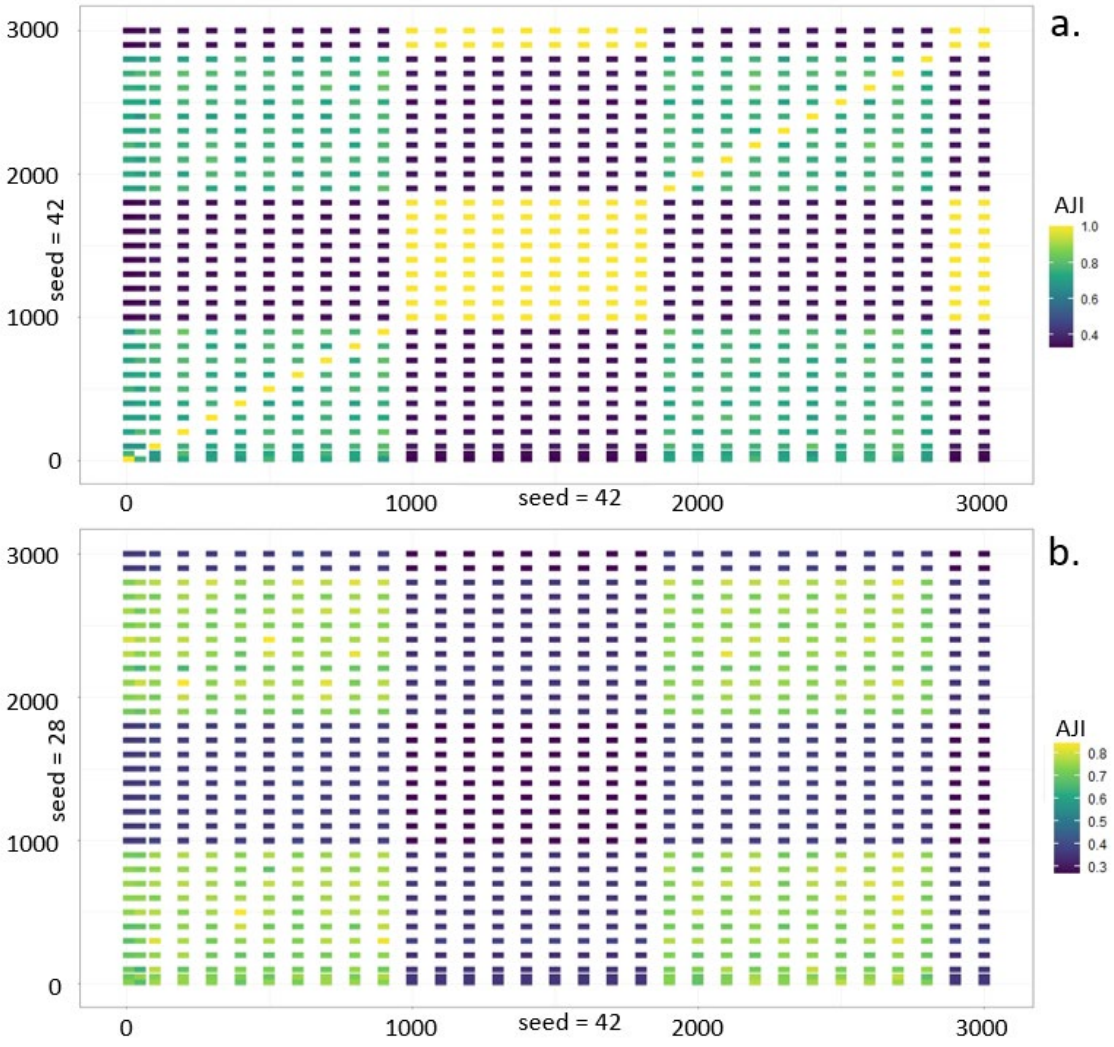

Figure 2. Heatmaps of AJIs of SOMs generated with different rlen values. Initialization = Random, Distance = Manhattan **a.** All parameters apart from rlen equal between SOMs. Seed = 42 **b.** Comparison of SOMs generated with seed values 28 and 42. (Data from Mateus, *et al.,* 2020)

This issue caused by the 32-bit integer limit was solved by simply changing the data format of the variable to one supporting 64-bit values. This allowed the algorithm to train for longer, but this did not result in a clear gradient of the SOMs becoming increasingly similar with higher rlen values. Instead, no clear difference in similarity was observed for SOMs made with very high rlen values

compared to SOMs created with an rlen around the default of 10 (Fig. 3). Therefore, to find another metric by which an rlen value from where the SOM becomes more stable and consistent can be extracted, the distance of the data to the nodes was analysed during the training process, as well as the amount of change made to the SOM nodes during each iteration over the dataset.

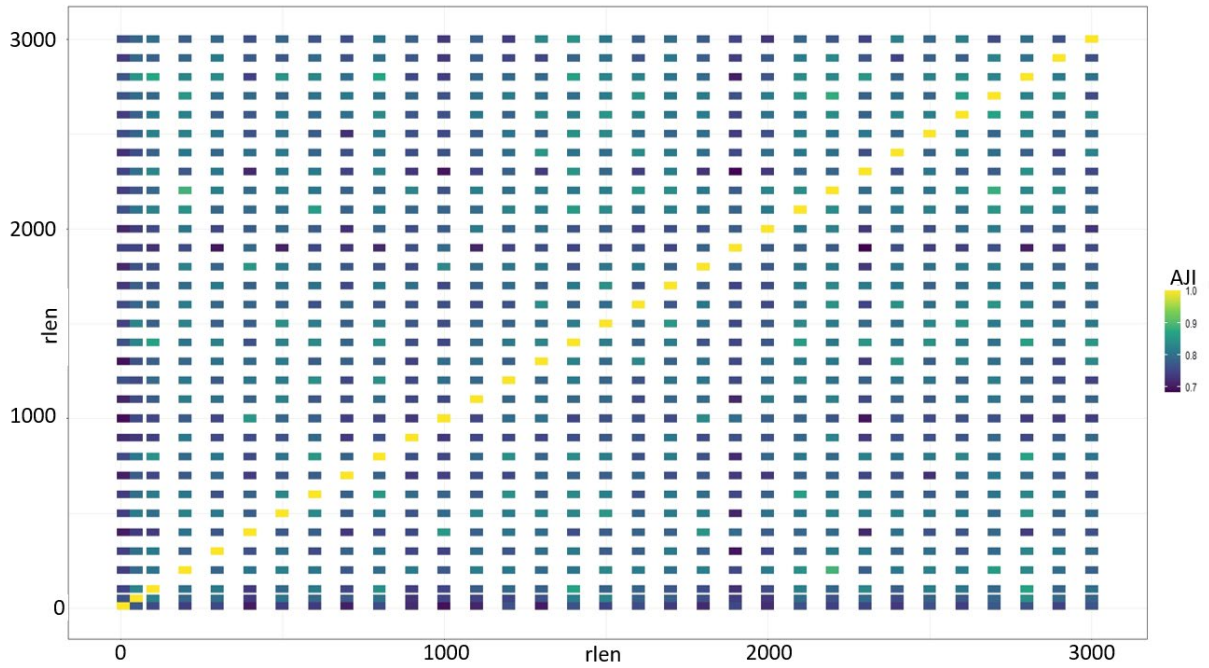

Figure 3. Heatmap of AJIs of SOMs generated with different rlen values after fixing integer limit issue. Initialization = Random, Seed = 42, Distance = Manhattan. (Data from Mateus, *et al.,* 2020)

The average distance and average change do not indicate SOM consistency

Since no increase in SOM consistency was found by analysing the rlen, other SOM features were investigated. First, the average distance of the cells to their nearest node was calculated after each rlen that passed during training. As mentioned before, the expectation is that the SOM grid will better represent the dataset when the rlen is increased, as the SOM will be modified by more and more data points. Therefore, as the SOM adapts to the data, the distance from each data point to the node closest to it is expected to gradually decrease and stabilize. However, instead of a gradual decline, a staircase pattern is observed when analysing the distance from the cells to their nearest nodes. These distances would fluctuate during the learning process, but then drop drastically when crossing certain rlen values (Fig. 4a). The same staircase pattern was observed when creating SOMs with increased rlen values (Fig. 4b). The drops in the average distances would occur after set increments of passed rlen values. For example, setting the total rlen to 3000 causes the drops to occur every 500 times the SOM was modified by the whole dataset. The staircase pattern seemed to be caused by the behaviour of the radius of the neighbourhood of the nearest node stored in the threshold variable. For each cell that is picked randomly during training, its nearest node gets defined, and only the nodes that are within the neighbourhood of that nearest node will be modified. The radius of the neighbourhood decreases over time, starting at 6 and going to 1, after which it drops to 0.5 with the default setting (Fig. 5). As the radius gets smaller, fewer nodes will be in the neighbourhood of each nearest node, and therefore a smaller number of nodes will be modified by each cell. Because cells are less likely to modify nodes other than their nearest nodes when the neighbourhood becomes smaller, the nodes will stay closer to the cells of which they are the nearest node. This is why the average distance from each cell to its nearest node decreases with the decreasing threshold value. The course of the neighbourhood radius suggests a gradual decline.

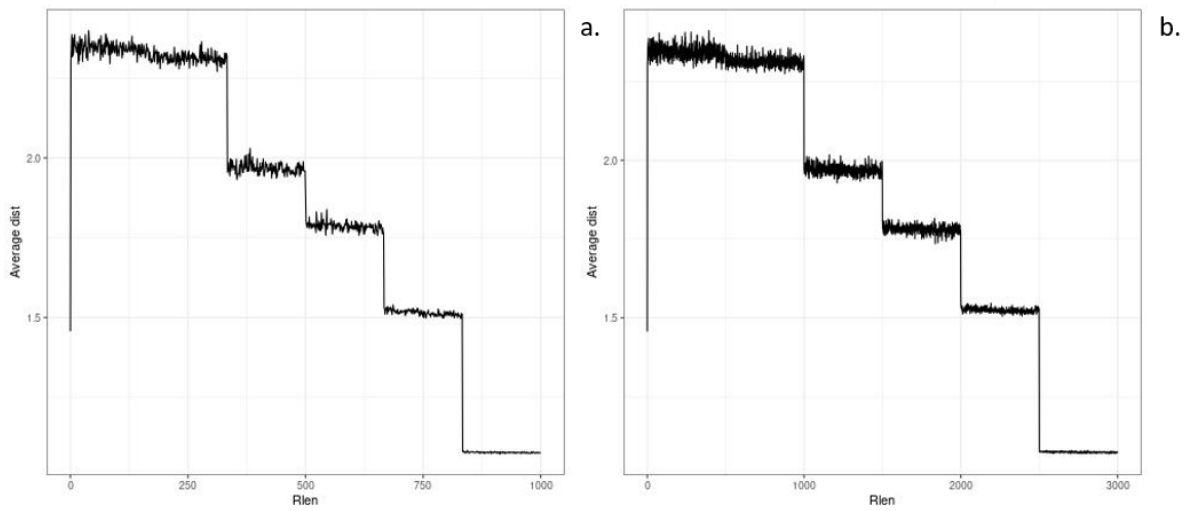

Figure 4. The average distance from cell to the nearest node during training after passed rlen values. Initialization = Random, Seed = 42, Distance = Euclidean **a.** rlen = 1000 **b.** rlen = 3000. (Data from Grifoni, *et al.,* 2020)

However, the distance between nodes is calculated as an integer value, which causes the decline of the radius to only take effect when it crosses an integer. The sudden drops in the average distance therefore occur at the same time as when the threshold variable crosses an integer value.

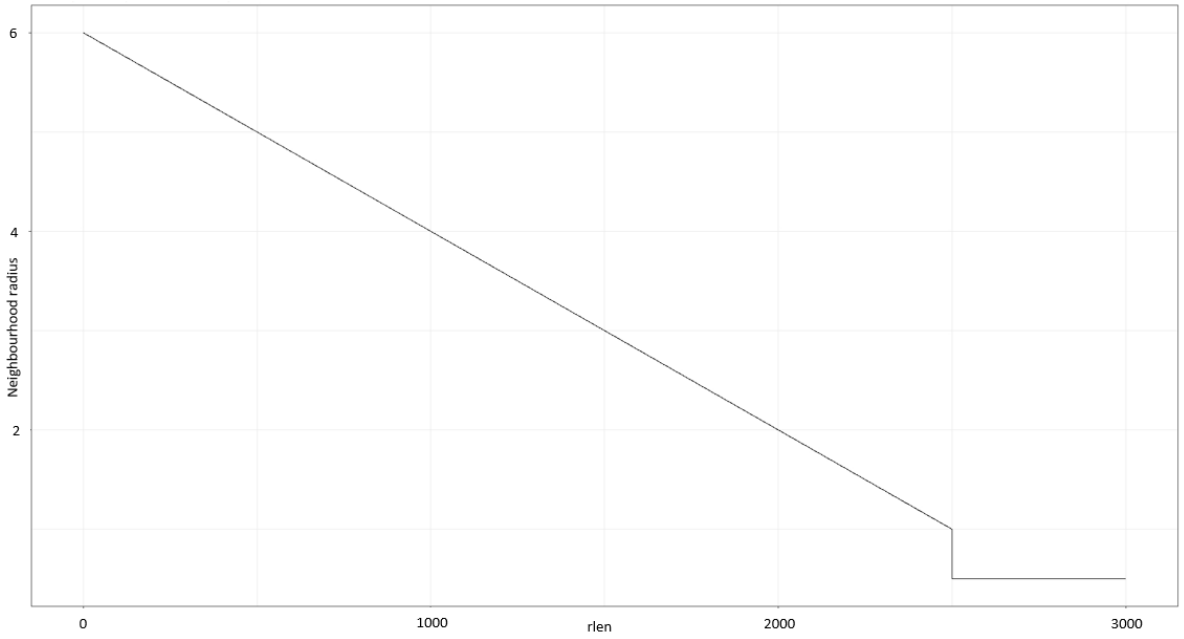

Figure 5. The radius of the neighbourhood of the nearest node versus passed rlen. Rlen = 3000

Unsurprisingly, as the decrease of the radius also determines it, the same staircase pattern is observed for the average change made to each node during training (Fig 6). The average change is calculated by adding up all the modifications caused by a cell to each node and dividing it by the number of nodes adapted by the cell, for each rlen during training. Similarly to how the threshold affects the distance to the nearest node, it also affects the average change. As the threshold decreases, the cells will start only modifying their nearest nodes, which will therefore stay closer to the cells. Because the cells get closer to their nearest node, the nearest node will not be modified as when it would be further away, which causes the average change to decline during training. This effect also seems to overshadow the influence of the learning rate, which directly influences how much a node will be modified by a cell. While the higher learning rates result in a higher average

change of each node, it is the threshold that causes the clear staircase pattern. This means that these variables are better suited as a representation of the SOMs learning process, than tools to measure when the SOM has become consistent.

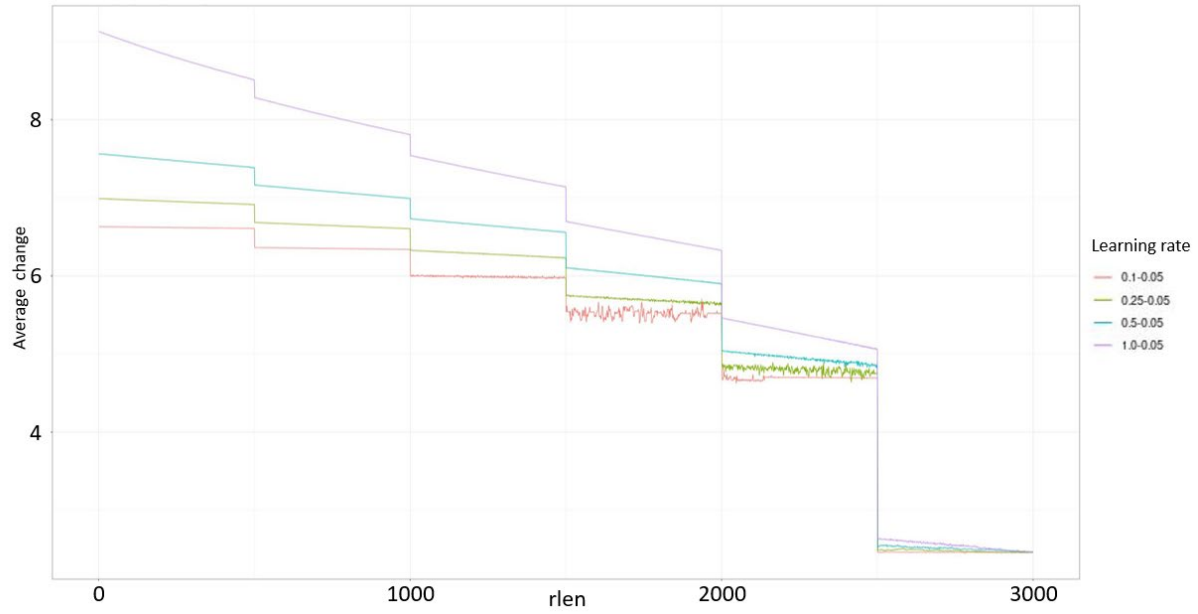

Figure 6. Average change per node versus passed rlen with different learning rates. Learning rate decreases during training, i.e. the highest learning rate included started at 0.1 when the grid had undergone no modification by the data, and ended at 0.05 after going over the dataset 3000 times. (Data from Grifoni, *et al.,* 2020)

SOM consistency rises and stagnates with low rlen values

As no clear increase in consistency of the SOM was observed by analysing the AJI of SOMs generated with rlen values of the default 10 and upwards, one possibility is that SOMs made with an rlen value of 10 are already as consistent as they are going to be. To test this idea, the algorithm was run with rlen values of 3000, 10, 1 and 0. Rlen equals 0 means that the algorithm would immediately cluster the cells after random initialization. The seed did also vary between 5 different values: 7, 28, 33, the default 42, and 98. Calculating the AJI for the SOMs made with different seeds and rlen values showed that, first of all, the similarity among SOMs stopped directly after random initialization was relatively low (AJI ≈ 0.265). This makes sense, as the output is essentially random. However, after 1 iteration of the dataset, the similarity among SOMs would rise substantially (AJI ≈ 0.663). With more training, an rlen value of 10, the SOMs would become a bit more similar to each other (AJI ≈ 0.771), but no clear improvement was observed when increasing the rlen up to 3000 (AJI ≈ 0.762). This finding suggests that the SOM could reach its maximum stability after a minimal amount of training.

The same analysis was performed while only comparing the cells that are the closest to their nearest nodes. The hypothesis was that these cells would be clustered into the same node more consistently, which would result in a higher similarity when comparing two SOMs. However, the results do not support this hypothesis. The AJI values of comparisons among maps generated with different seed values show that excluding cells relatively far from their nearest node did not improve the similarity between SOMs (Fig. 8). Instead, including these cells has a positive effect on the AJI values. This effect was observed for other rlen values of 0, 1, and 3000 as well and picking the cells that are closest per node instead of overall did not change the outcome either. The reason may be that cells close to the nodes may be affected by subtle grid changes more than cells that may be considered outliers. If we imagine a grid of nodes, one cell in the middle of the grid, and one very far away, it

seems intuitive that if the grid moves, the cell in the middle will switch its nearest node, while the one that lies far away will not. This would explain why only selecting the cells closest to the grid seems to negatively affect the similarity between FlowSOM outputs.

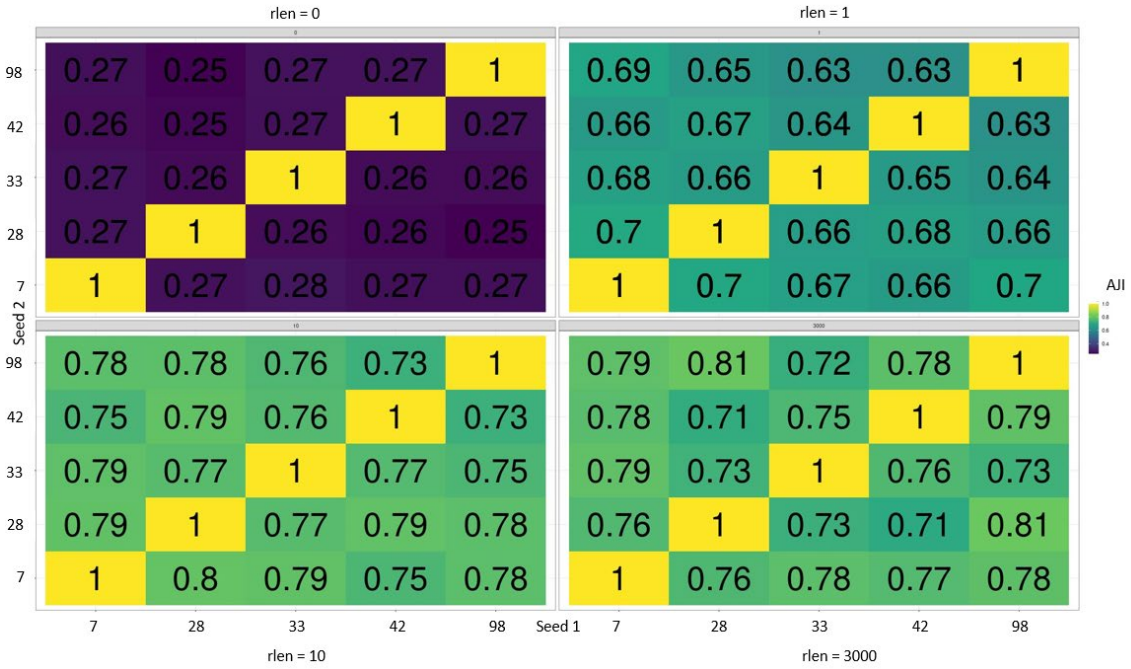

Figure 7. AJIs for SOMs created with different seeds and rlen values. (Data from Grifoni*, et al.,* 2020)

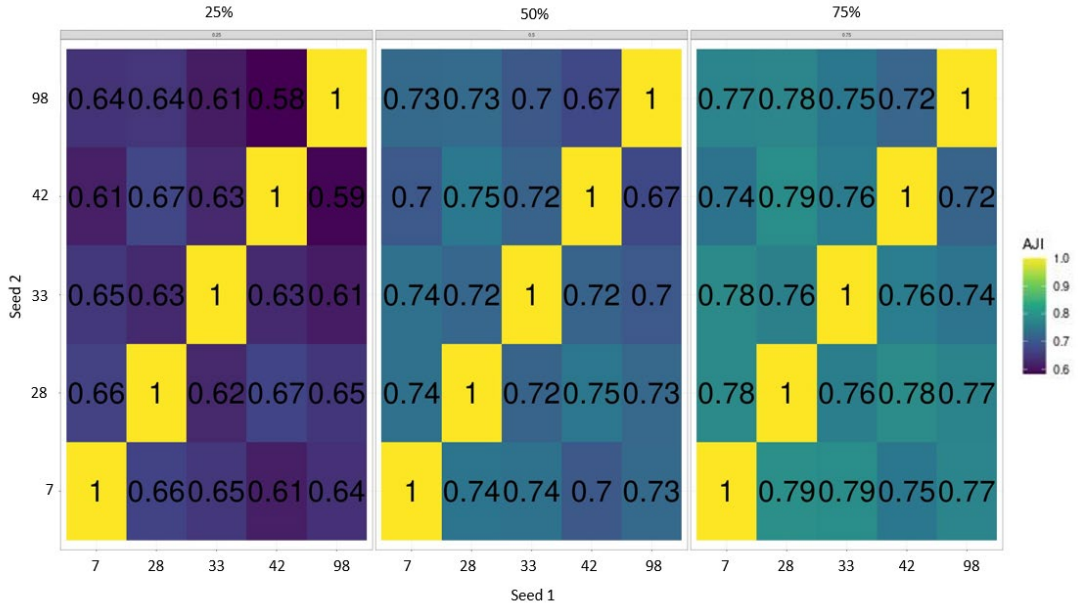

Figure 8. AJIs for SOMs created with different seeds. AJI calculated between SOMs containing only the 25, 50 and 75% of cells that are closest to their nearest nodes. Rlen = 10 (Data from Grifoni*, et al.,* 2020)

Conclusion and discussion

FlowSOM is a promising tool for building an automated flow cytometry pipeline due to its speed and accuracy compared to other available clustering methods, and its ability to generate easy visualisations of the clustered data in a 2D grid. However, it is difficult to decide on a standard combination of parameter settings for use with this package. While the SOM does seem to become more stable during training, it is hard to tell when it has become the best representation of the

dataset as it is going to be, as each extra datapoint will always change the grid by a certain amount. Moreover, parameters representing the state of the SOM, like the learning rate, distance from the cells to the nodes, and the change done to the nodes by the cells during training are not fit to indicate the consistency of the SOM. Instead, these indicators show how the SOM is being modified during training. It therefore makes sense to look at the finalised product as a whole, which shows that running the FlowSOM algorithm with the default rlen of 10 could give a SOM that could be considered stable, as increasing the length of the training process does not improve the similarity between maps that are made with various seed values, for example. This may be due to the dataset containing enough cells for the algorithm to be able to adapt to it within a few iterations, as was also observed by the creators of FlowSOM (Van Gassen, *et al.,* 2015). On the positive side, this would mean that a map representing around 4 million cells, each of which containing information about the presence of around 10 specific marker proteins, can be generated in mere minutes. However, more analysis and improvement of FlowSOM will need to be done before it can be used as the basis for an autonomous, automatic, flow cytometry analysis tool. This would require finding new ways to analyse and improve the quality and consistency of the SOM, which could include further analysis of the neighbourhood radius for example, as it has been kept at its default value for this study, and parameters that have not yet been explored during this study, like the total number of nodes. Furthermore, ideas from other SOM algorithms may be included. The Parameter-Less Self-Organizing Map algorithm, for example, was built to not be as reliant on learning rate and neighbourhood radius values, as these are calculated based on the map's current quality as a representation of the dataset (Berglund & Sitte, 2006). Apart from the SOM, other aspects of the FlowSOM package should also be explored. The package offers a variety of clustering algorithms to generate metaclusters of nodes and increasing the maximum number the algorithm is allowed to create from the default 30 to the maximum value of 90 greatly affected the outcome for the datasets used in this study. Also, for the analysis to be fully automated, the pre-processing of the data, including getting rid of debris and doublets before using FlowSOM should be performed within the pipeline. So although more research will be needed before the computer can perform the cytometry analysis on its own, advancements in this topic could prove to be extremely useful for an instrumental tool in a wide variety of biological and biomedical research.

Acknowledgements

Special thanks to Rianne Rijken, Bram Gerritsen and Aridaman for the supervision and support they provided during this project.

References

- Azad A (2021). *flowVS: Variance stabilization in flow cytometry (and microarrays)*. R package version 1.26.0
- Berglund, E., & Sitte, J. (2006). The Parameterless Self-Organizing Map Algorithm. IEEE Transactions on Neural Networks, 17(2), 305–316. https://doi.org/10.1109/tnn.2006.871720
- Van Gassen, S., Callebaut, B., Van Helden, M. J., Lambrecht, B. N., Demeester, P., Dhaene, T., & Saeys, Y. (2015). FlowSOM: Using self-organizing maps for visualization and interpretation of cytometry data. Cytometry Part A, 87(7), 636–645. https://doi.org/10.1002/cyto.a.22625
- Grifoni, A., Weiskopf, D., Ramirez, S. I., Mateus, J., Dan, J. M., Moderbacher, C. R., Rawlings, S. A., Sutherland, A., Premkumar, L., Jadi, R. S., Marrama, D., De Silva, A. M., Frazier, A., Carlin, A. F., Greenbaum, J. A., Peters, B., Krammer, F., Smith, D. M., Crotty, S., & Sette, A.

(2020). Targets of T Cell Responses to SARS-CoV-2 Coronavirus in Humans with COVID-19 Disease and Unexposed Individuals. Cell, 181(7), 1489–1501.e15. https://doi.org/10.1016/j.cell.2020.05.015

- Leicht, E. A., Holme, P., & Newman, M. E. (2006). Vertex similarity in networks. Physical Review E, 73(2), 026120.
- McKinnon, K. M. (2018). Flow Cytometry: An Overview. Current Protocols in Immunology, 120(1). https://doi.org/10.1002/cpim.40
- Mateus, J., Grifoni, A., Tarke, A., Sidney, J., Ramirez, S. I., Dan, J. M., Burger, Z. C., Rawlings, S. A., Smith, D. M., Phillips, E., Mallal, S., Lammers, M., Rubiro, P., Quiambao, L., Sutherland, A., Yu, E. D., Da Silva Antunes, R., Greenbaum, J., Frazier, A., . . . Weiskopf, D. (2020). Selective and cross-reactive SARS-CoV-2 T cell epitopes in unexposed humans. Science, 370(6512), 89–94. https://doi.org/10.1126/science.abd3871
- Ochatt, S. J. (2008). Flow cytometry in plant breeding. Cytometry Part A: the journal of the International Society for Analytical Cytology, 73(7), 581-598.
- Shapiro, H. M. (2005). Practical flow cytometry. John Wiley & Sons.
- Weber, L. M., & Robinson, M. D. (2016). Comparison of clustering methods for high-dimensional single-cell flow and mass cytometry data. Cytometry Part A, 89(12), 1084–1096. https://doi.org/10.1002/cyto.a.23030