Introduction to high-parameter analysis

For mass cytometry

Hefin Rhys
Francis Crick Institute
2020-01-05

Let's get started

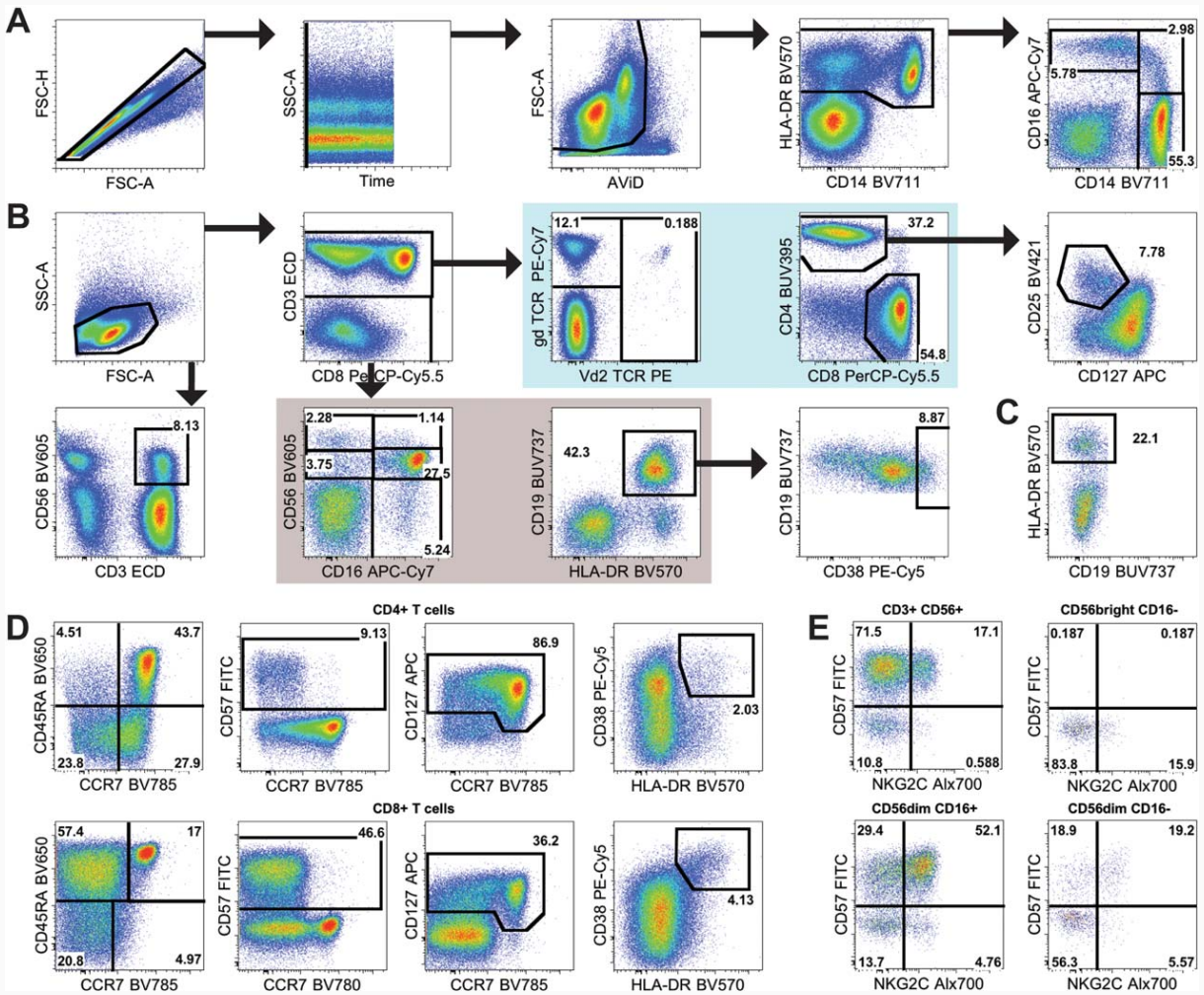
What are the goals of our analysis?

- Identify all cell types present in the sample
- Compare population frequencies between samples
- Compare antigen expression between samples
- Track cell development pathways
- Characterise functional state of populations

What challenges are presented by mass cytometry data?

- Larger number of parameters than conventional cytometry
- Visualizing the patterns in the data becomes difficult
- Manual gating is difficult to impossible
  - p(p-1)/2 number of bivariate combinations (435 combinations for 30 paramaters)
  - subjective
  - very likely to miss populations

What challenges are presented by mass cytometry data?



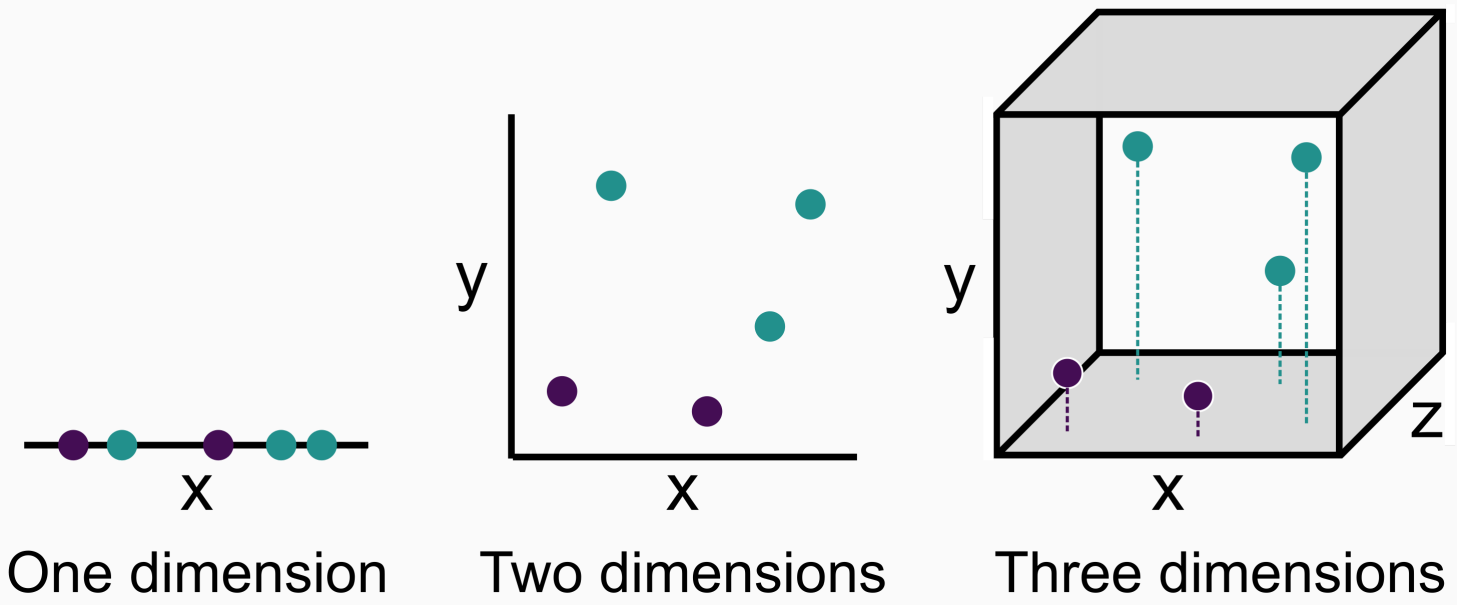
Moncunill et al., 2014

How do we solve these challenges?

We turn to computational, **machine learning** methods to gate populations for us.

But these techniques can perform poorly when the number of parameters is high. This is due to a phenomenon known as **the curse of dimensionality**.

The curse of dimensionality

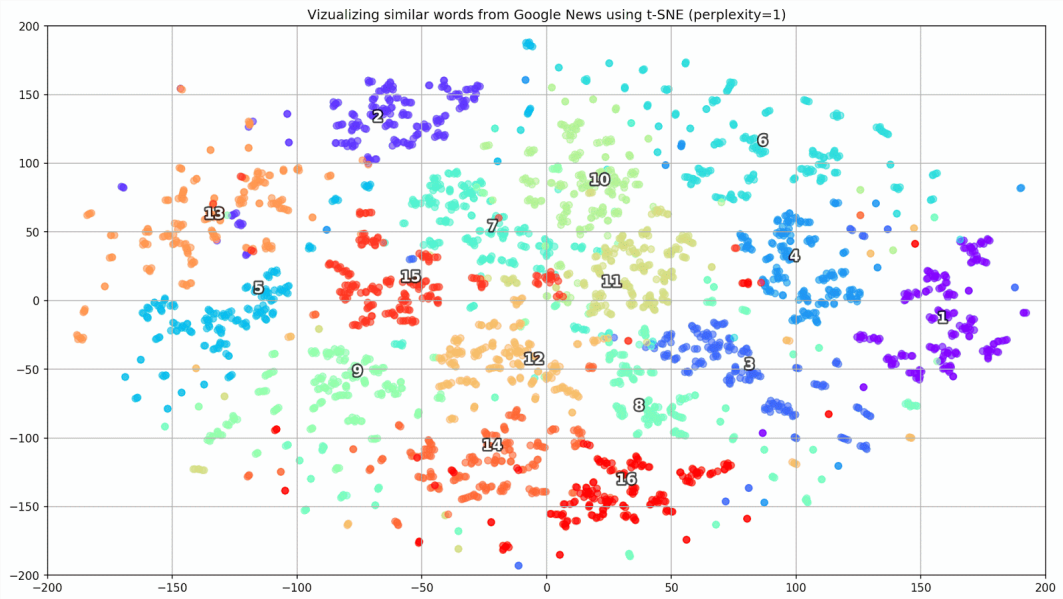


- Algorithms that rely on calculating the **distance** between points, suffer in high-dimensional space
- In high dimensions most of the "volume" of the **feature space** is empty, or **sparse**
- Algorithms are more likely to start learning from noise in the data, rather than signal
- Distance also starts to lose its meaning

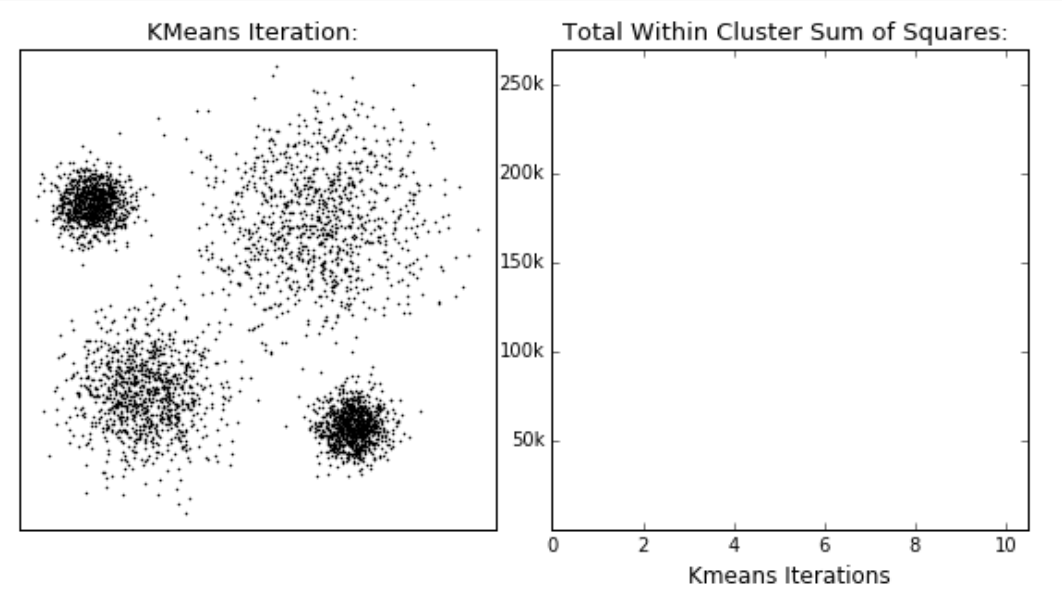
How do we solve these challenges?

Unsupervised machine learning!

Dimension reduction



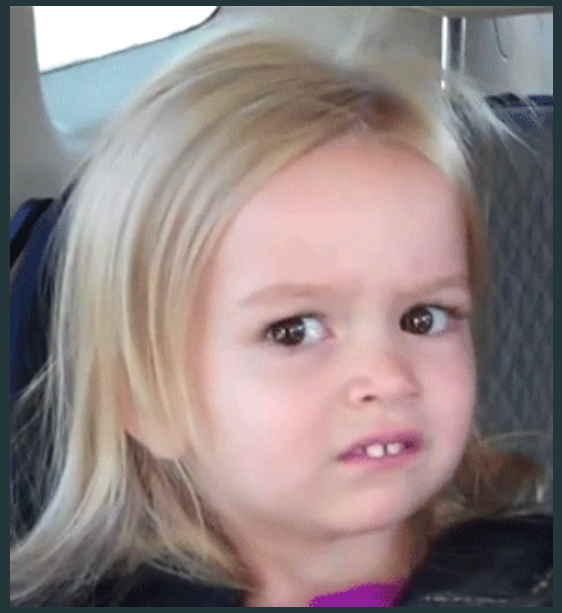
Clustering



What is dimension reduction?

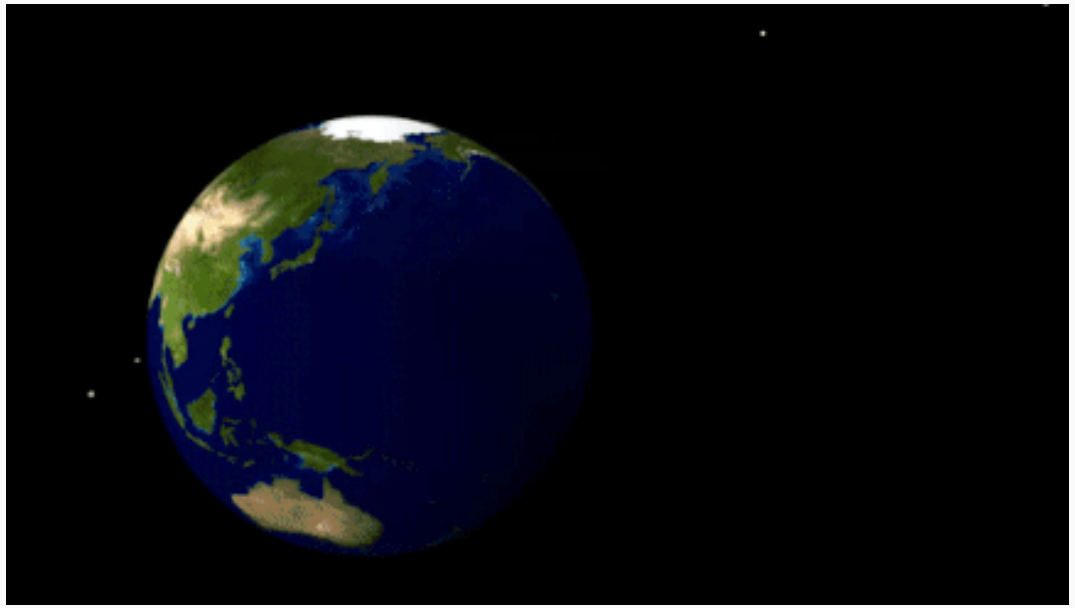
Dimension reduction algorithms aim to convert a large number of dimensions into a smaller number of dimensions, while preserving as much of the original, high-dimensional information as possible.

What is dimension reduction?

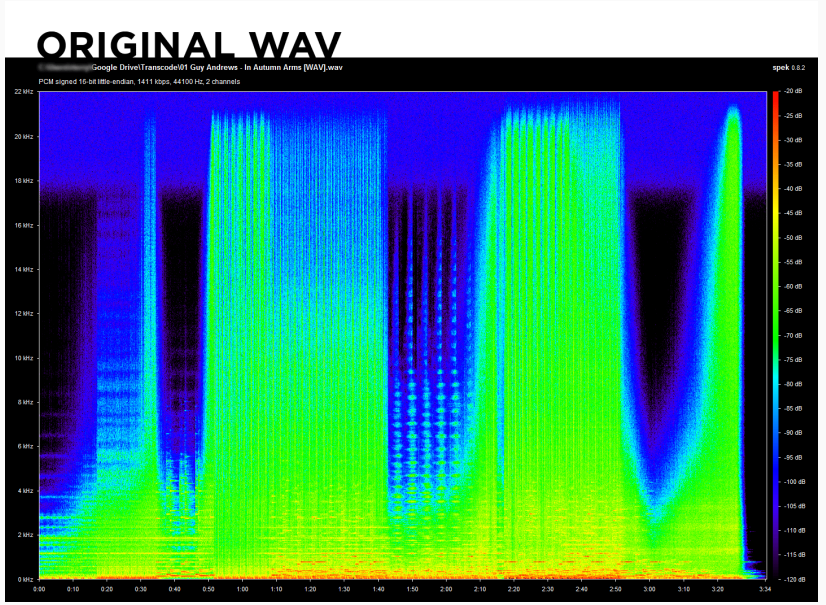


What is dimension reduction?

Mercator projection



Audio compression



Common dimension reduction algorithms

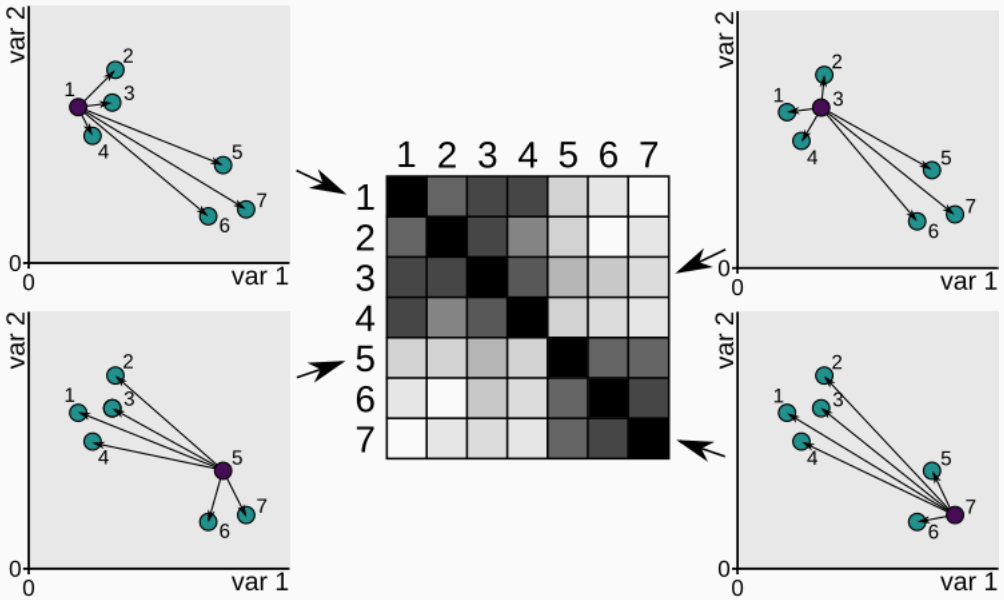
- **Principal components analysis**
  - Linear dimension reduction
  - The first few PCs explain most of the variation in the data, the rest can be discarded
  - Usually performs poorly for flow cytometry data
- **t-distributed stochastic neighbor embedding (t-SNE)**
  - Non-linear dimension reduction
  - Computationally expensive
  - Works well for flow cytometry data
- **Uniform manifold approximation and projection (UMAP)**
  - Non-linear dimension reduction
  - Not as computationally expensive
  - Works well for flow cytometry data

A brand new algorithm called PHATE was released in December: (https://www.biorxiv.org/content/10.1101/120378v1)

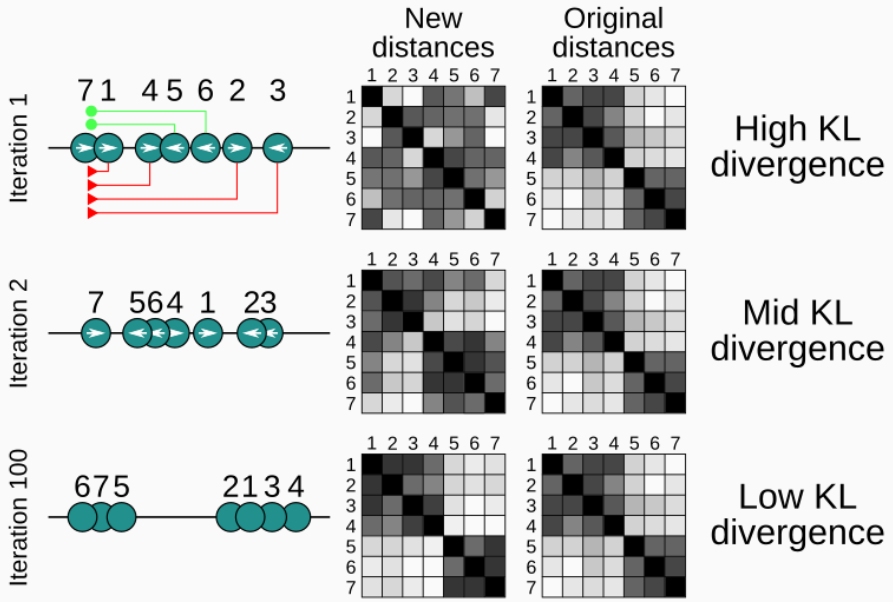
t-SNE

How does t-SNE work?

Calculate distances between each event, and every other event



Randomly distribute the events in a new, two-dimensional space. Iteratively move events closer together that were close by originally

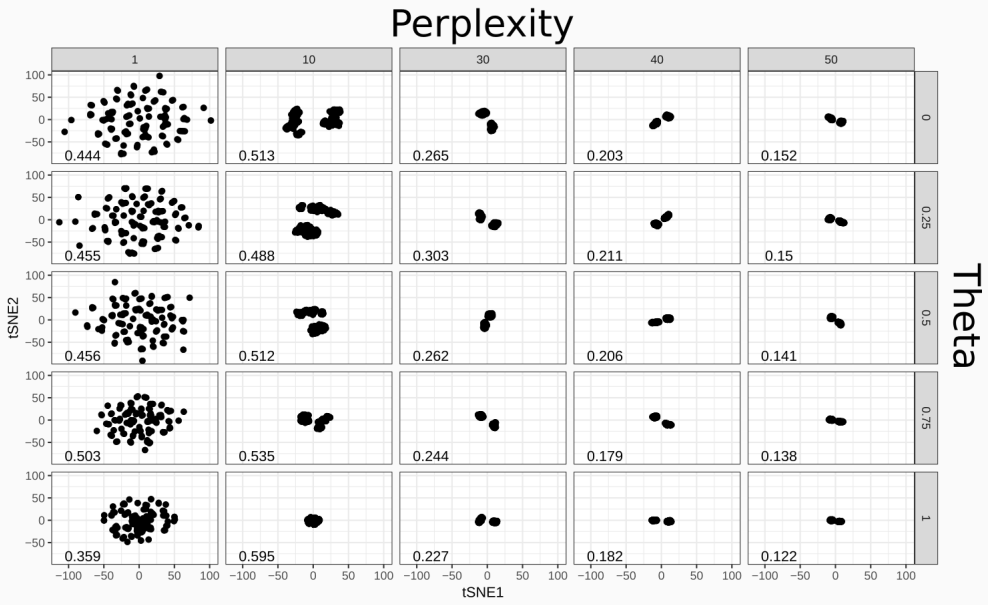


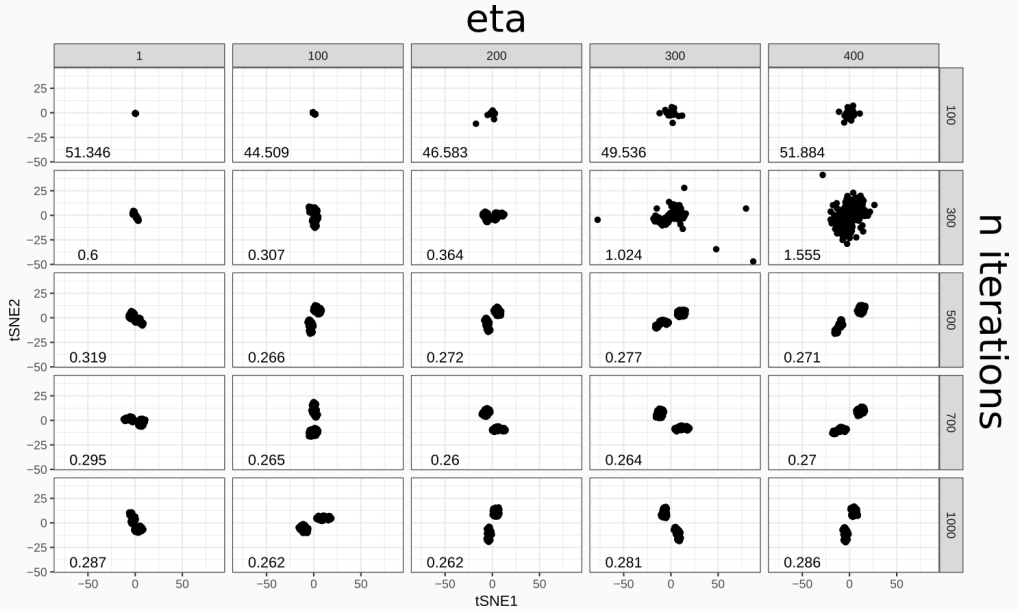
What does a t-SNE look like for mass cytometry data?

Hyperparameters of t-SNE

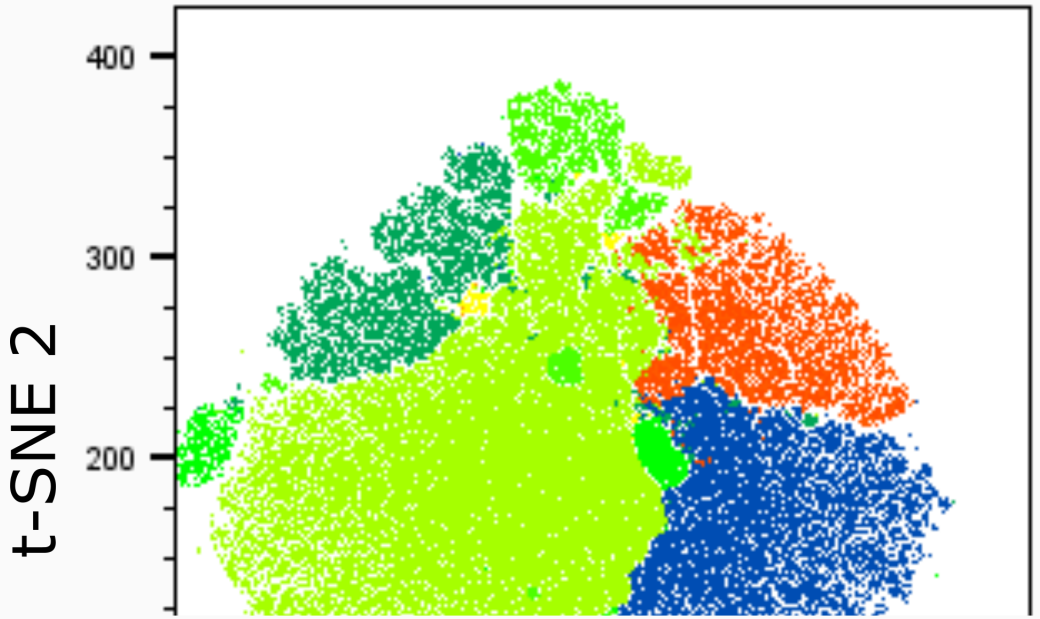
The t-SNE algorithm has **hyperparameters** that control what the final embedding looks like:

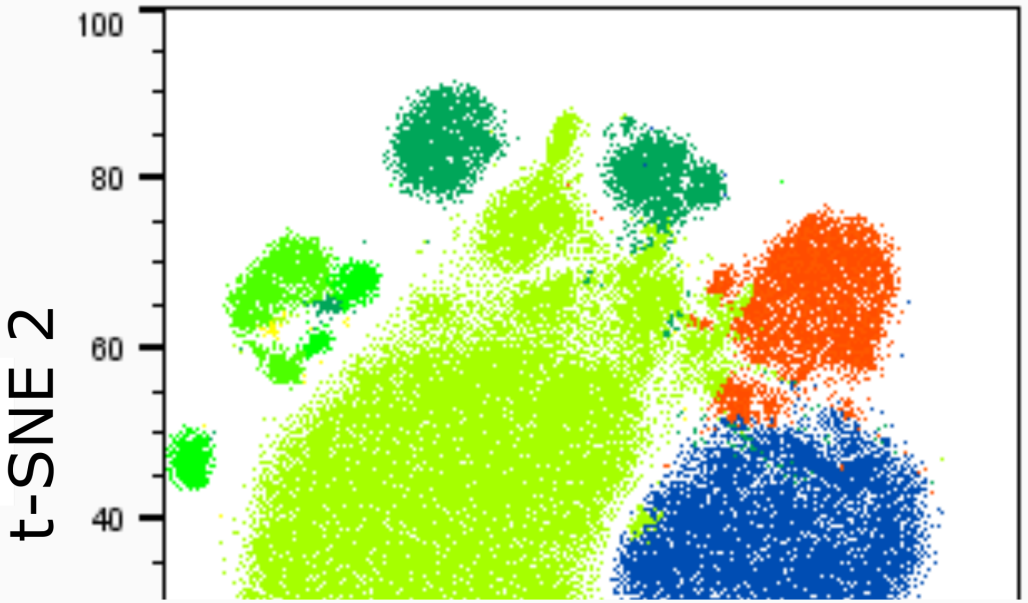
- perplexity (emphasis on global vs local structure)
- theta (0 -> 1 increases speed but decreases accuracy)
- eta (learning rate, default value usually fine)
- n iterations / epochs (must be high enough to converge)





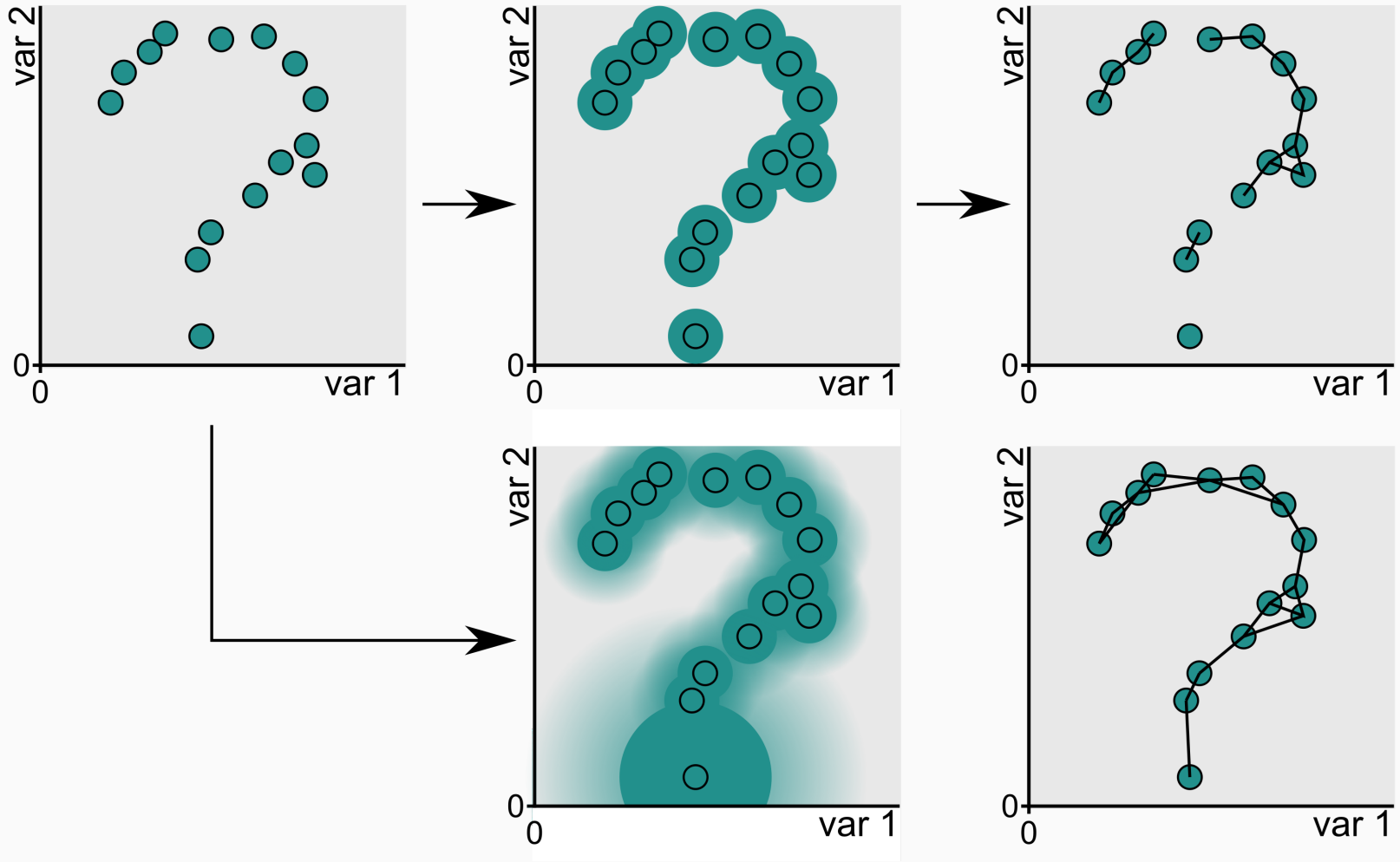
Hyperparameters of t-SNE



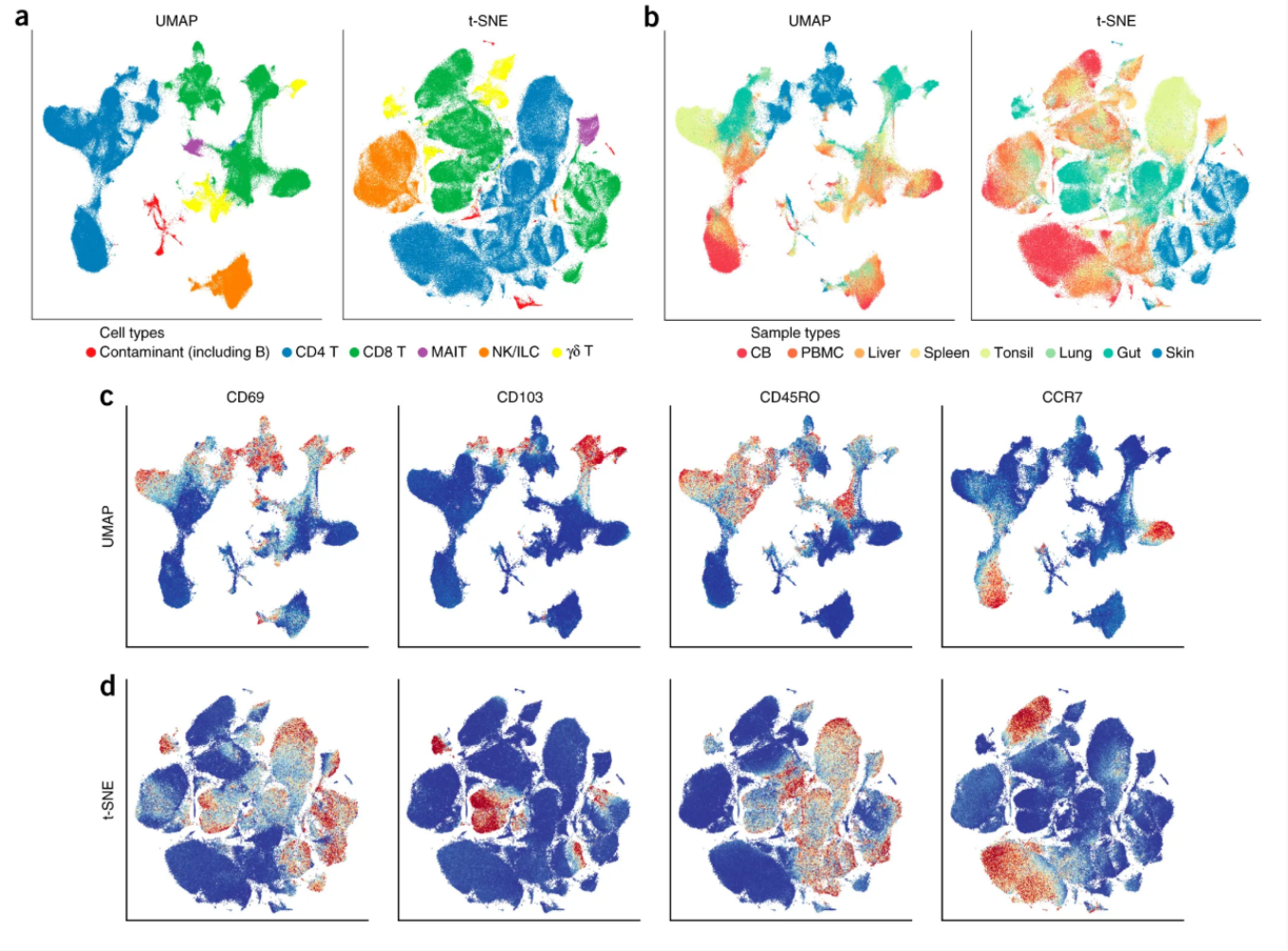


UMAP

How does UMAP work?



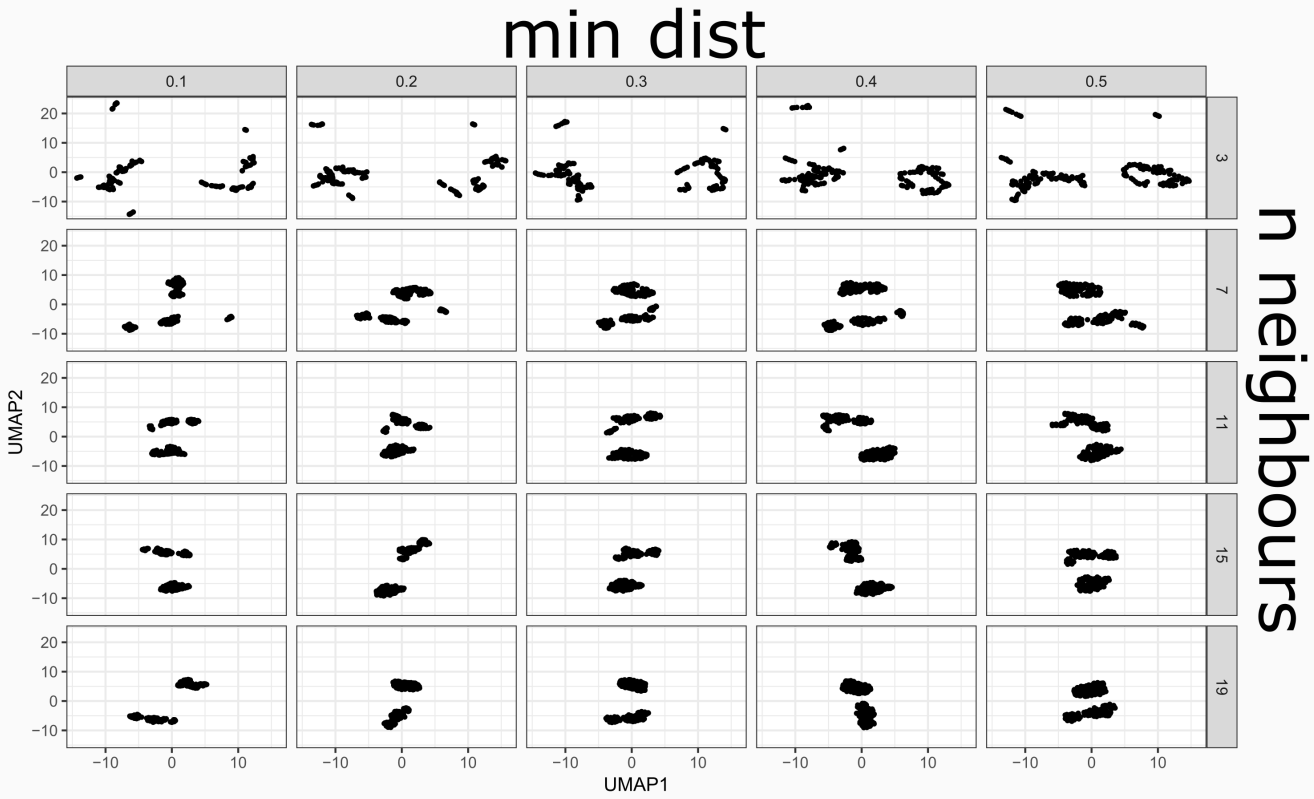
What does a UMAP look like for mass cytometry data?



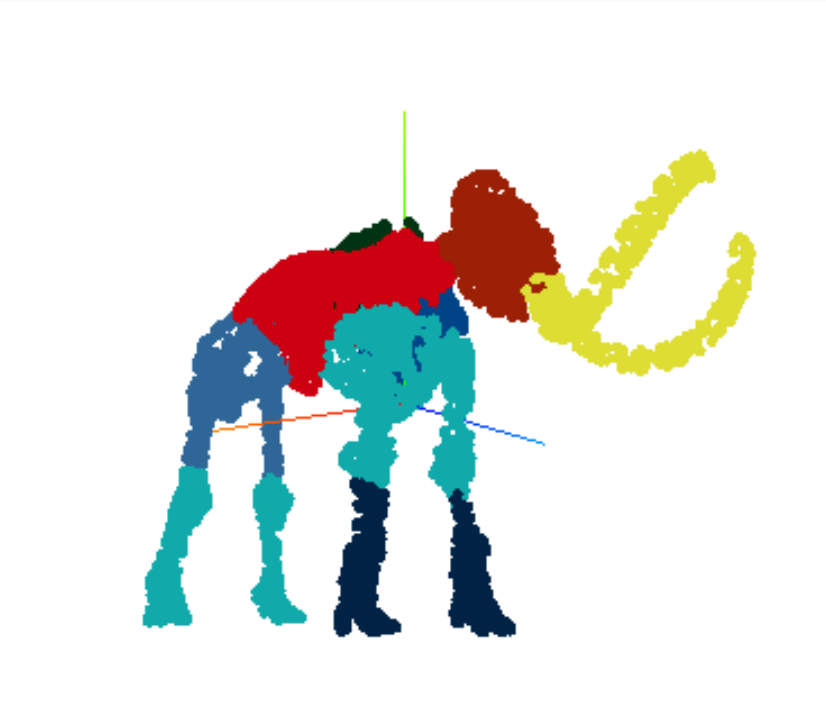
Hyperparameters of UMAP

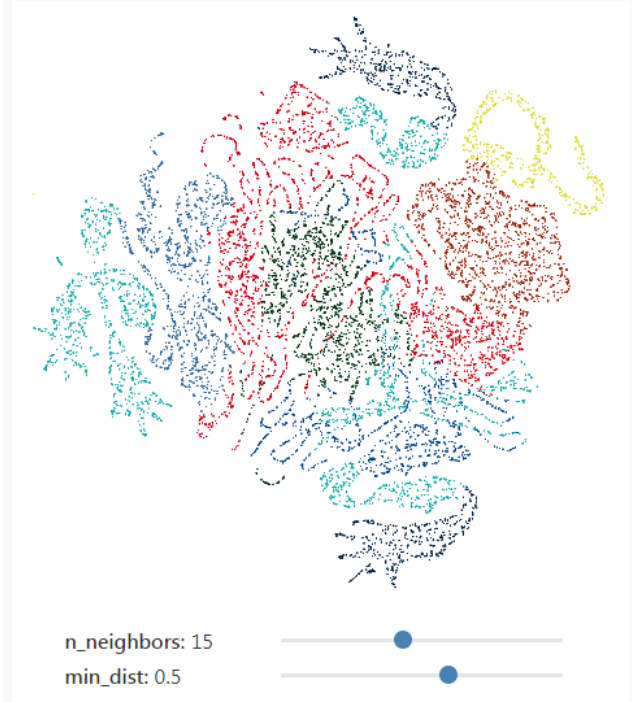
The UMAP algorithm also has hyperparameters that control what the final embedding looks like:

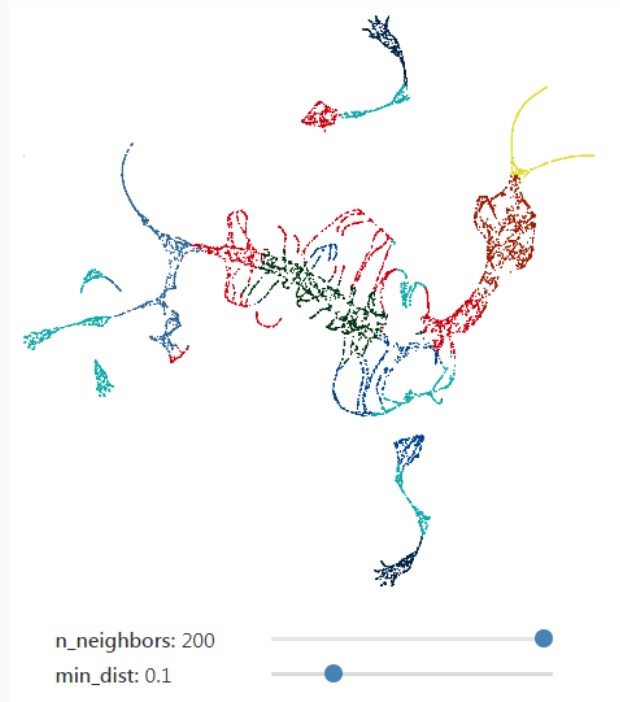
- n neighbours (emphasis on global vs local structure)
- min distance (clumpy vs spread out points)
- n iterations / epochs (must be high enough to converge)



Hyperparameters of t-SNE

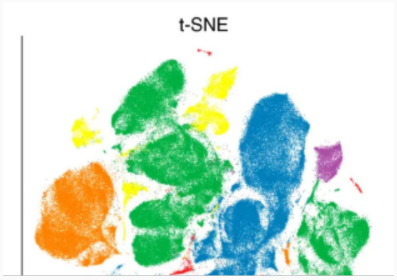






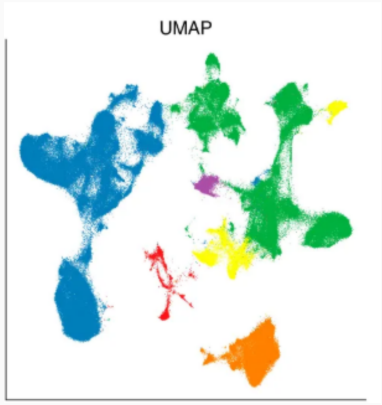
t-SNE vs. UMAP

t-SNE



- Preserves local structure but not global structure
- Computationally expensive
- Cannot map new data onto embedding
- Gives a different embedding each run
- Islands of points tend to be more globular

UMAP



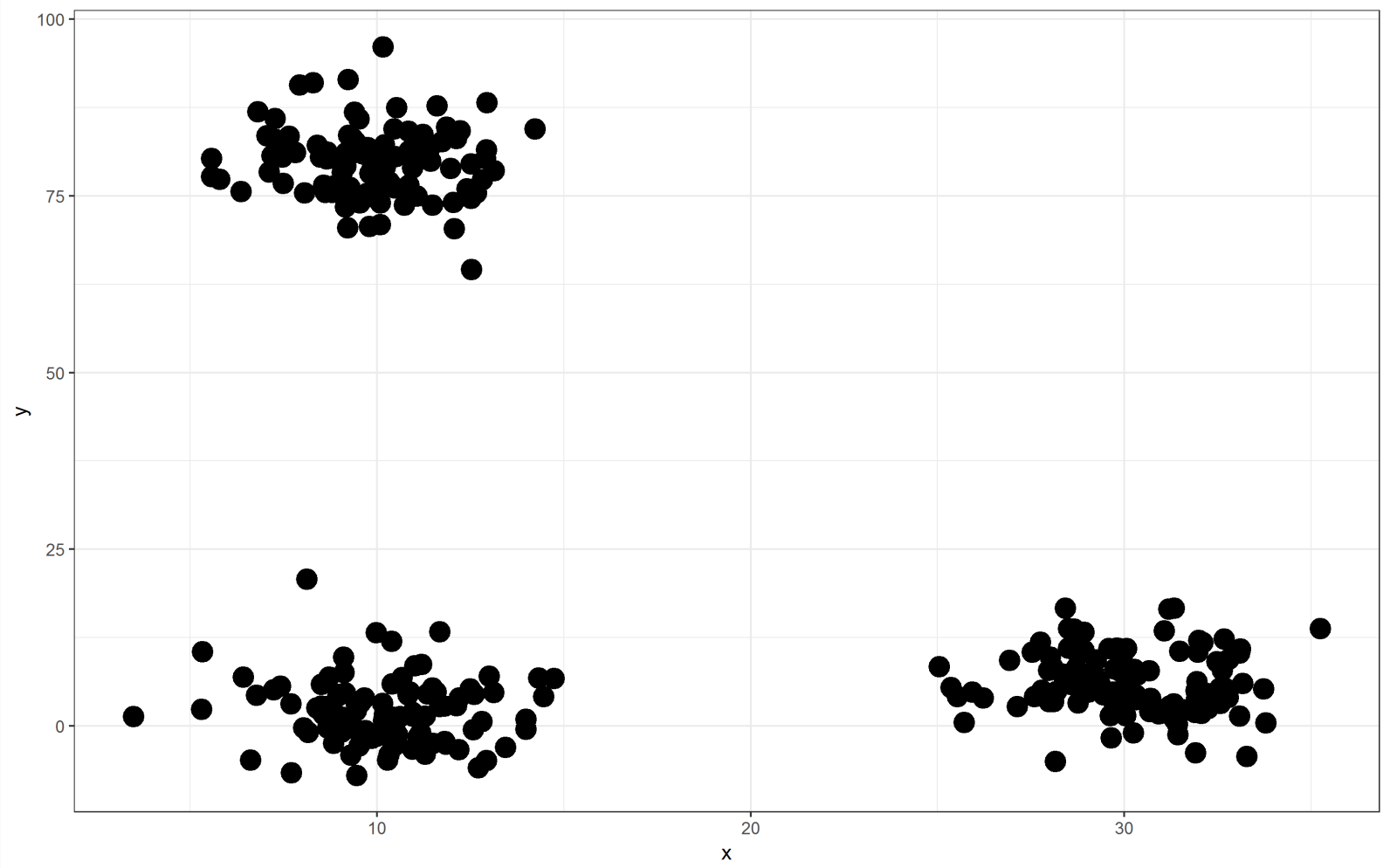
- Preserves local and *some* global structure
- Computationally less expensive
- New data can be mapped onto existing embedding
- Gives the same embedding each run
- Islands of points tend to follow a continuum

What is clustering?

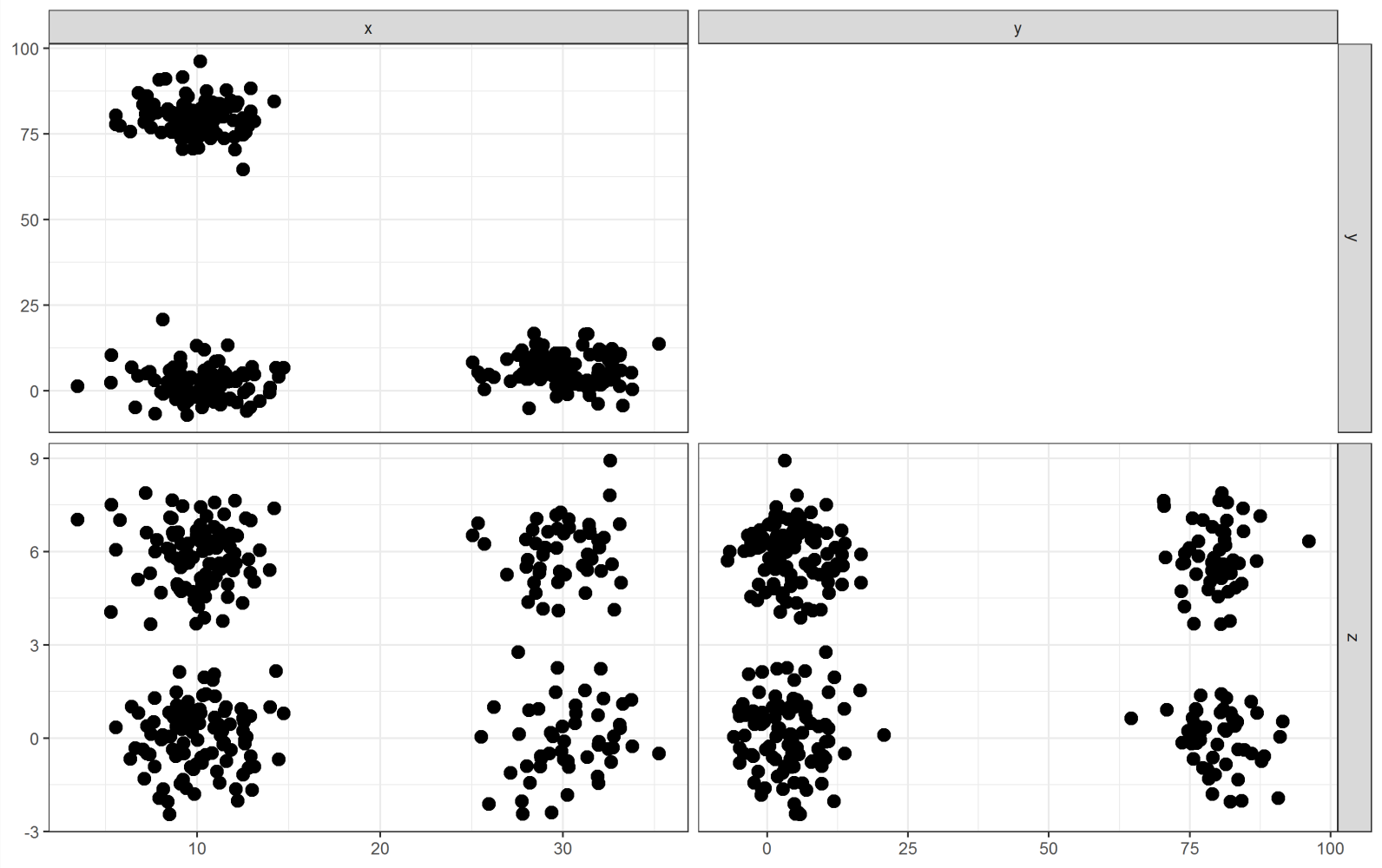
Clustering algorithms aim to partition the dataset into discrete **clusters** (populations). A cluster is a set of data points that are more similar to each other, than data points in other clusters.

What is clustering?

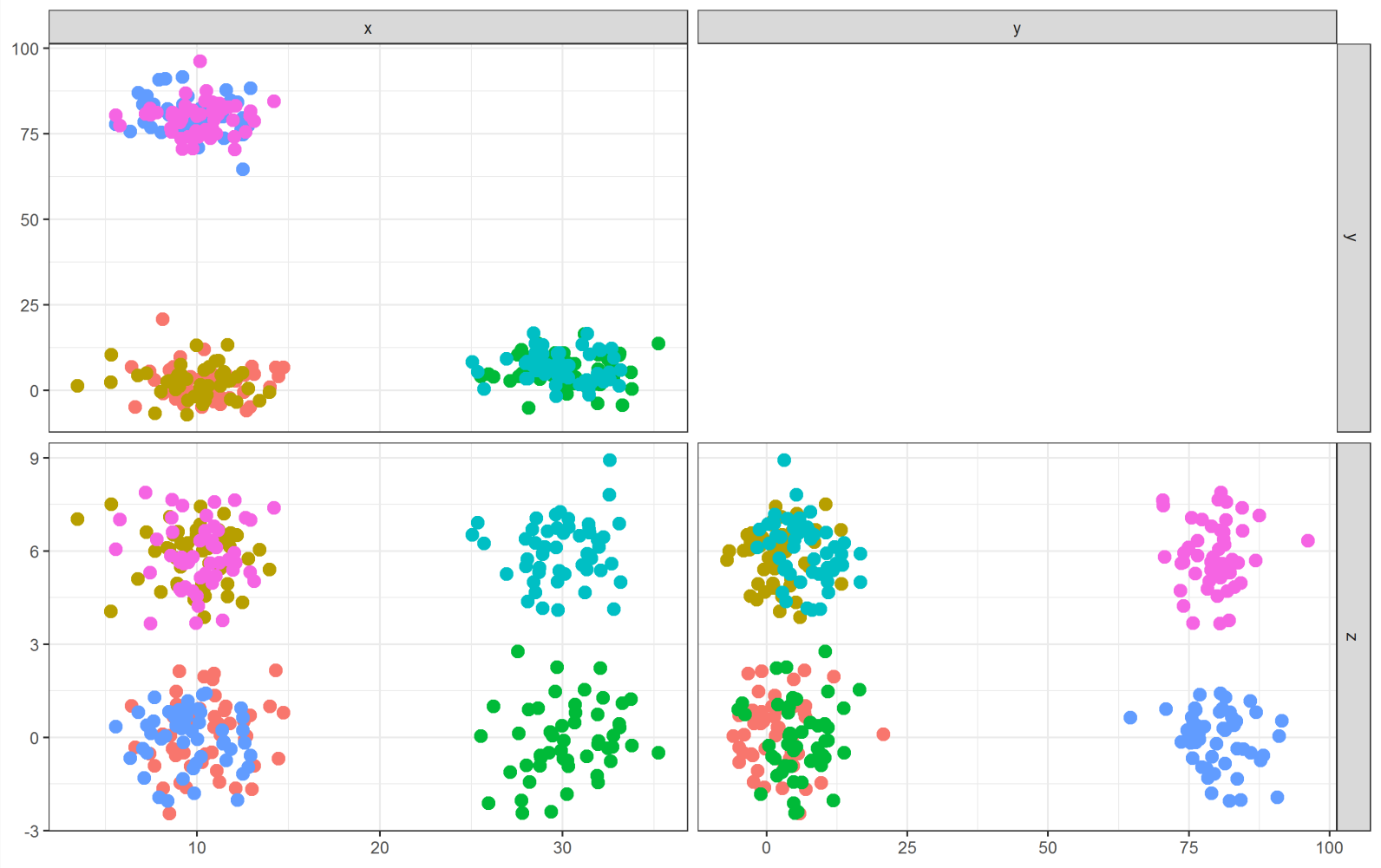
What is clustering?



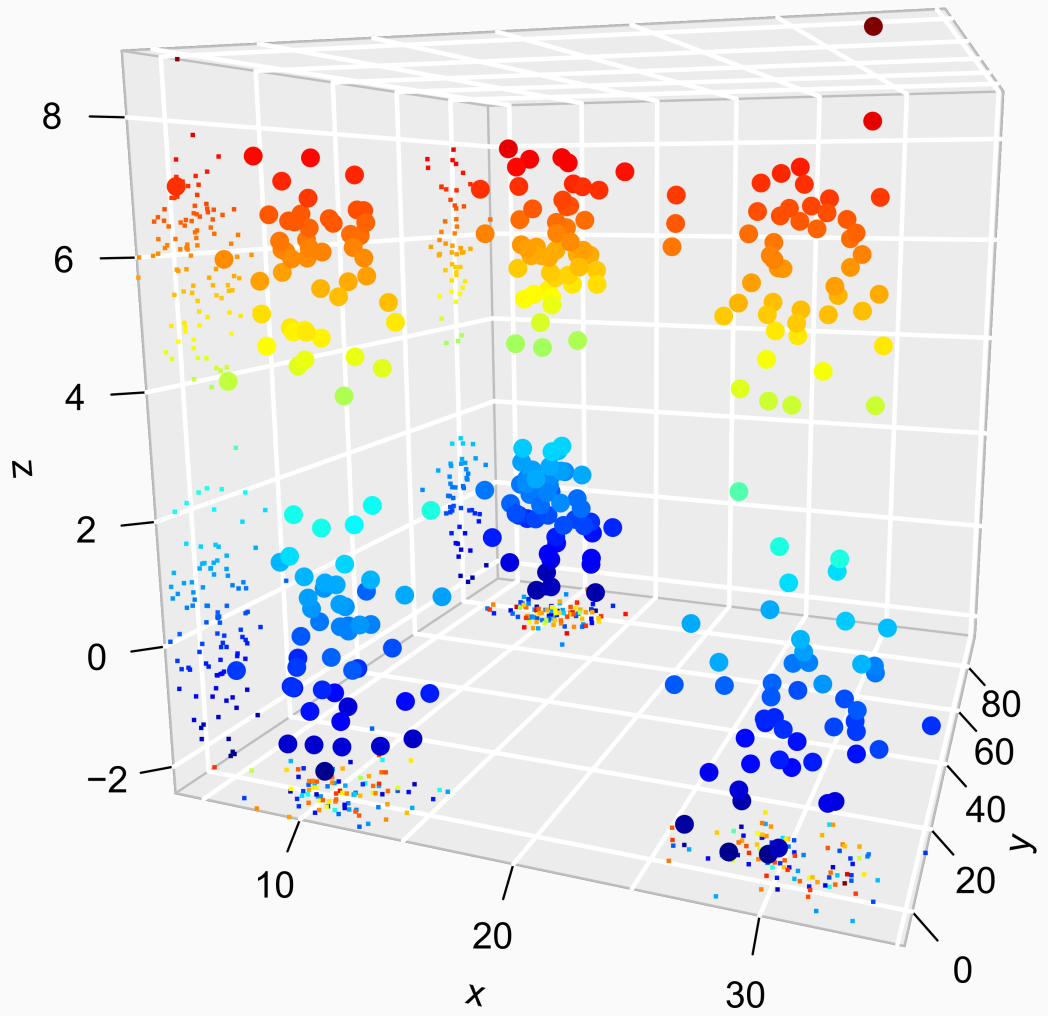
What is clustering?



What is clustering?



What is clustering?



Common clustering algorithms

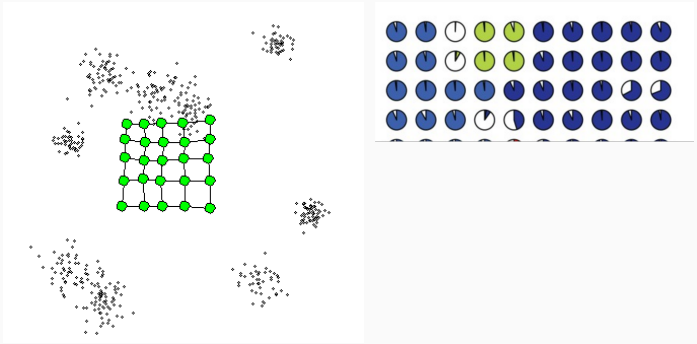
- **k-means/medians** (not great for cytometry)
- **Hierarchical clustering** (not great for cytometry)
- **flowMeans and flowPeaks** (outdated)
- **SPADE** (probably superceded by newer algorithms)
- **flowSOM** (state of the art)
- **Phenograph** (state of the art)
- **flowGrid** (very new, state of the art)

flowGrid was released in April 2019: (https://bmcsystbiol.biomedcentral.com/articles/10.1186/s12918-019-0690-2#article-info)

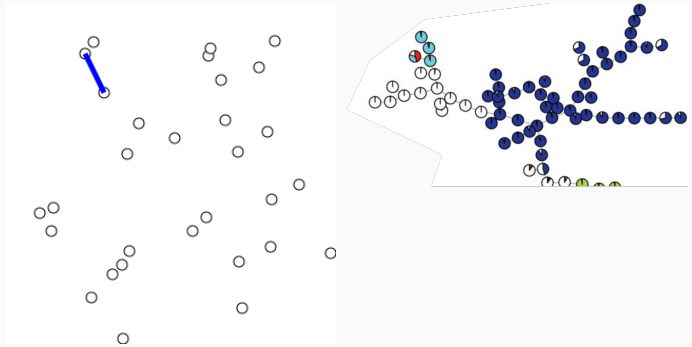
flowSOM

How does flowSOM work?

1 - Reduce the number of dimensions using a **self-organizing map (SOM)**

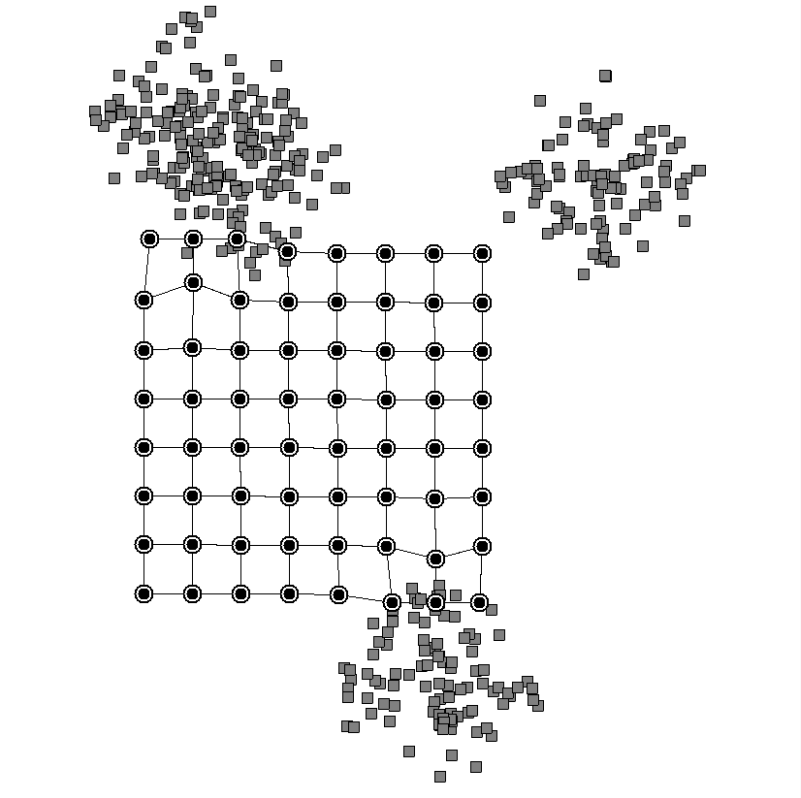


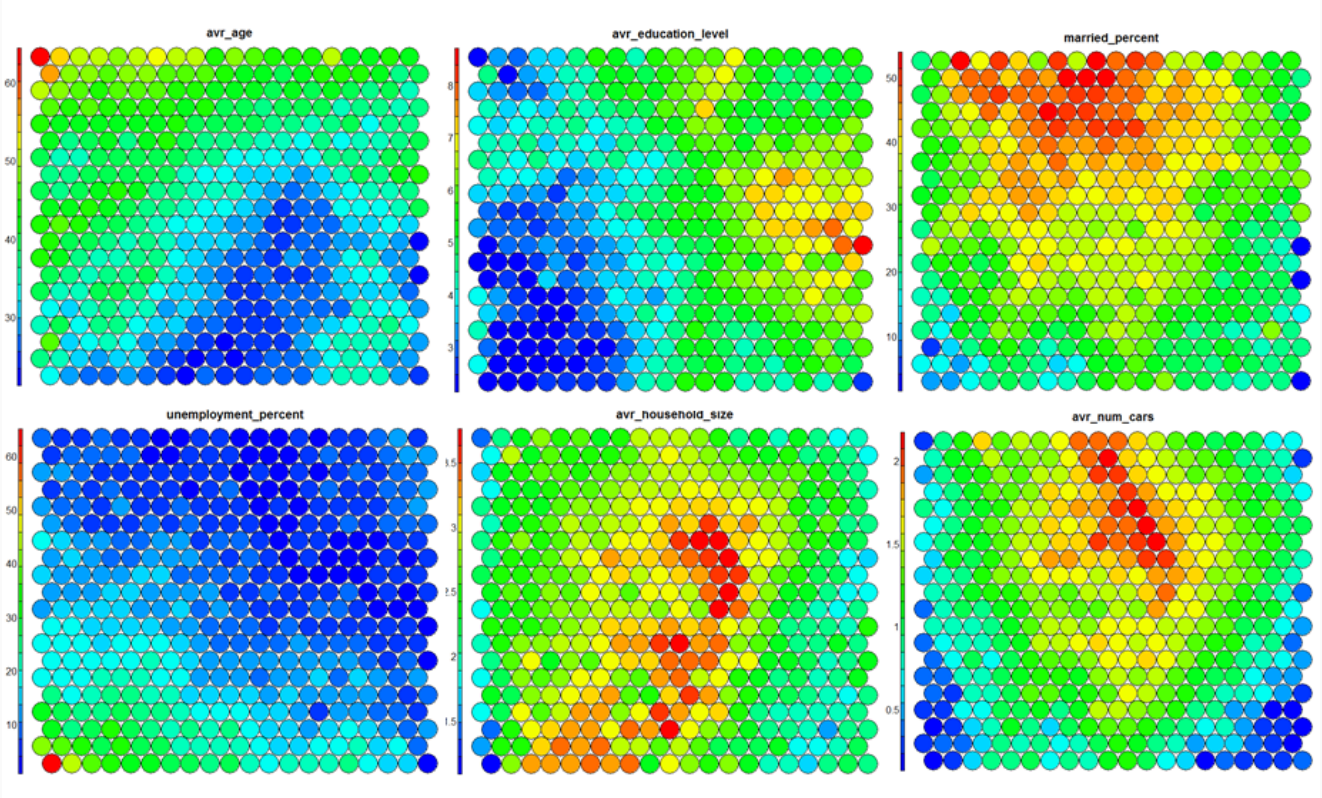
2 - Arrange the SOM into a **minimal spanning tree** for visualisation



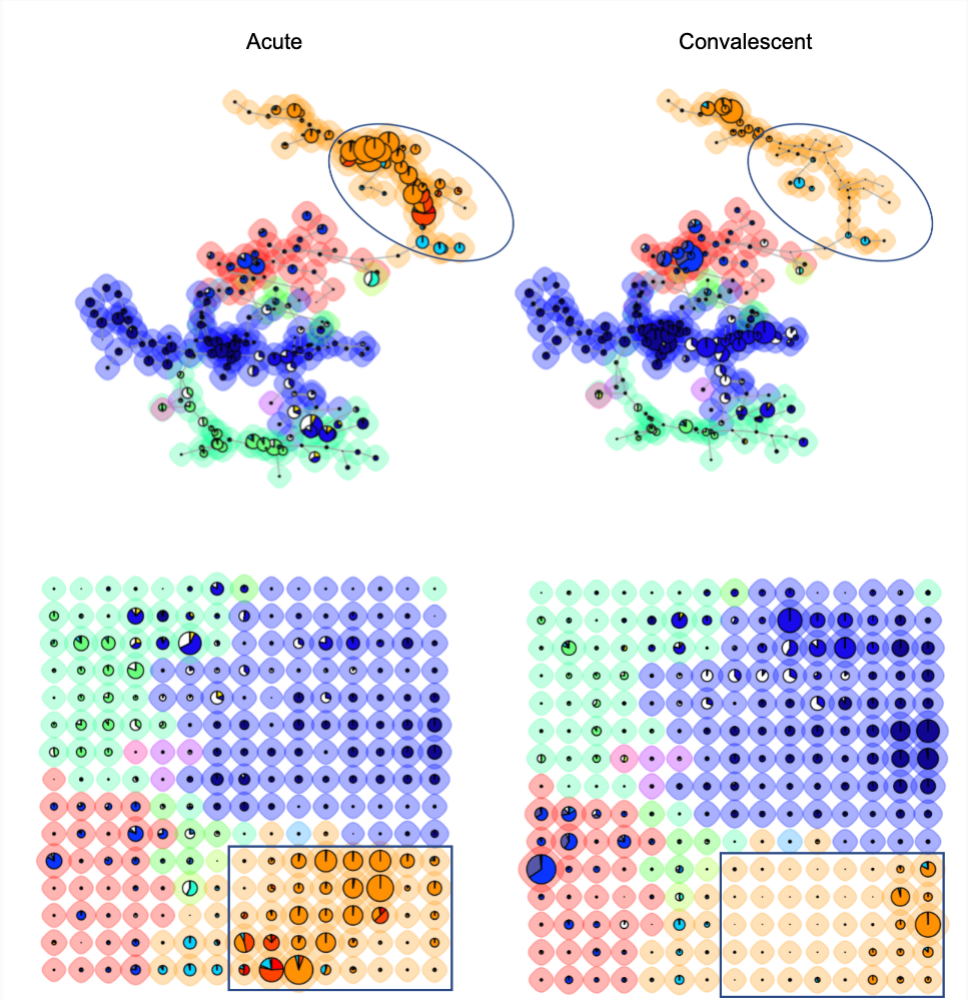
3 - Find clusters of nodes in the SOM, and place the events in the cluster of their node

Creating the self-organizing map (SOM)



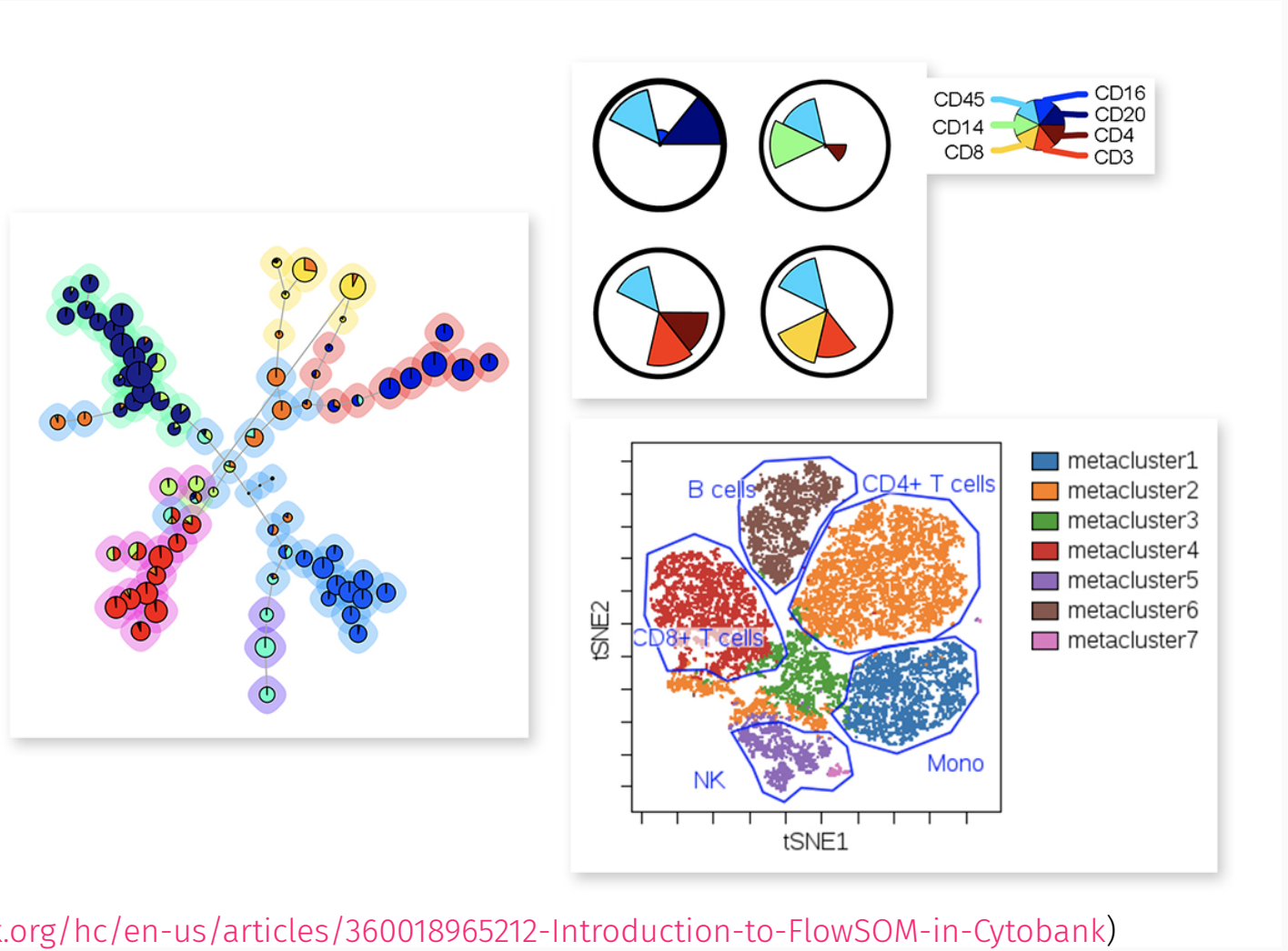


Arrange the SOM into a minimal spanning tree



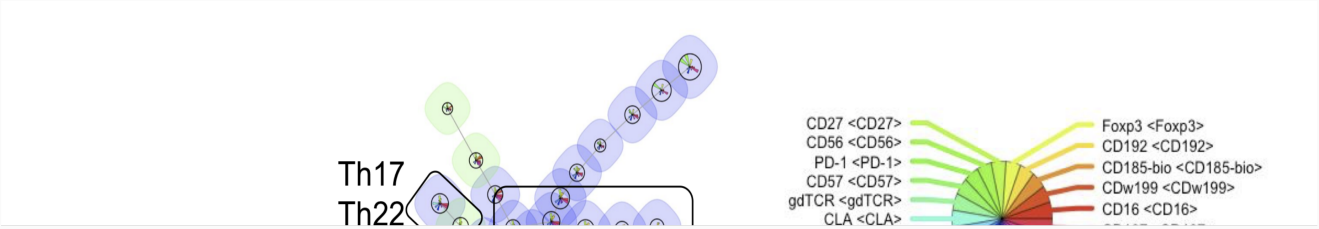
(https://blog.cytobank.org/2019/03/15/beginners-guide-to-flowsom-profiling-the-innate-immune-response-to-viral-infection/)

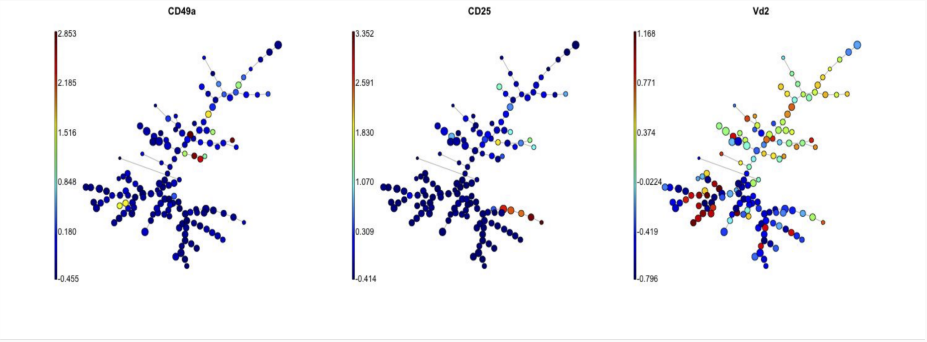
Cluster the nodes



(https://support.cytobank.org/hc/en-us/articles/360018965212-Introduction-to-FlowSOM-in-Cytobank)

What does flowSOM look like for mass cytometry data?

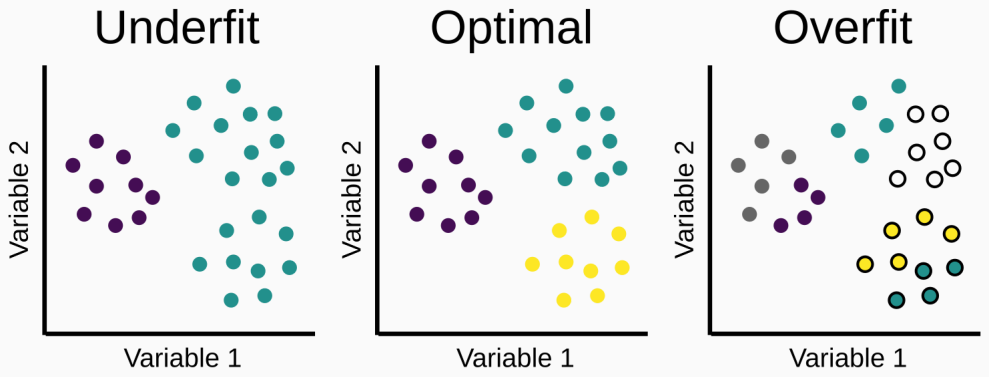




(https://github.com/hammerlab/t-cell-data/issues/29)

Choosing the number of clusters

- Identifying the number of clusters in a dataset is an **ill-posed problem**
- Some clustering algorithms need us to state how many clusters to find. Some identify clusters automatically, but may disagree with each other
- Deciding on the the number of clusters in a dataset can be hard and there may not even be one correct answer
- While clustering is a form of **unsupervised** machine learning, ALWAYS validate your clusters manually



Choosing the number of clusters with flowSOM

1 - Use *a priori* knowledge about the number of clusters

2 - Manually try a range of "sensible" numbers of clusters

3 - Allow flowSOM (in R) to choose for you

Whichever method you choose, you MUST evaluate your clustering model

Validating a clustering model

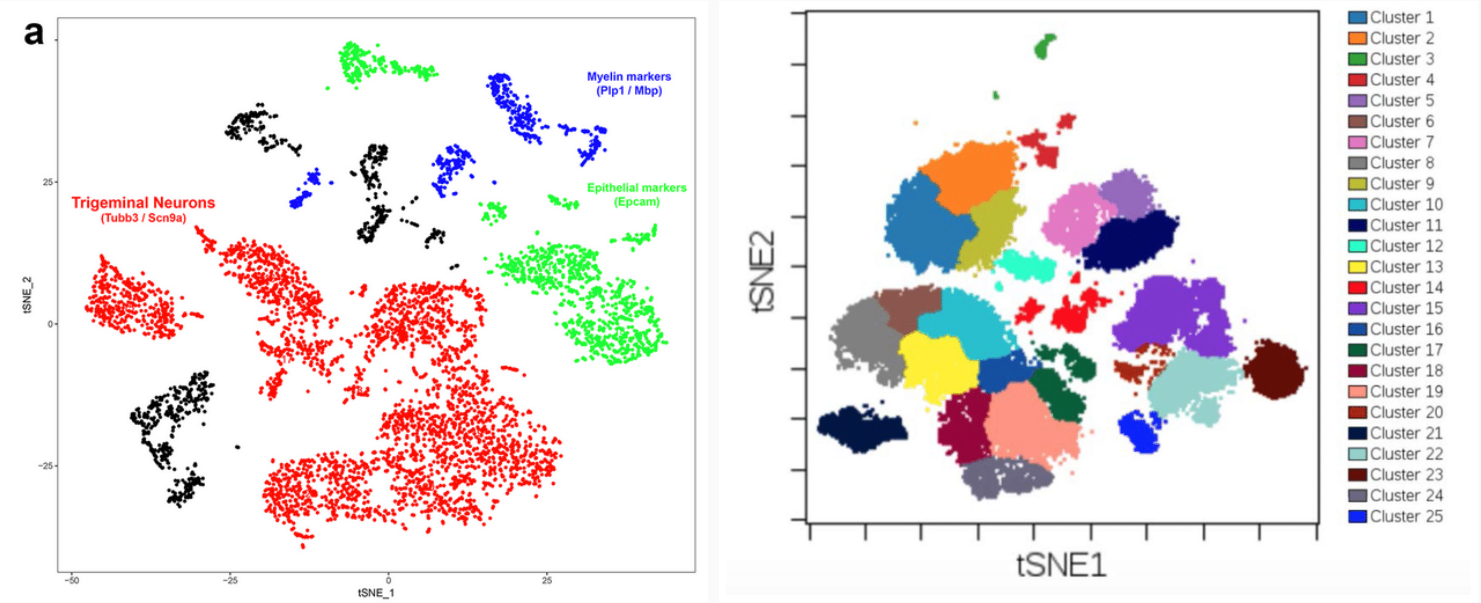
1 - Map cluster labels onto a lower-dimensional representation

2 - Map cluster labels onto bivariate plots

3 - Plot expression data for each cluster

Validating a cluster model

1 - Map cluster labels onto a lower-dimensional representation

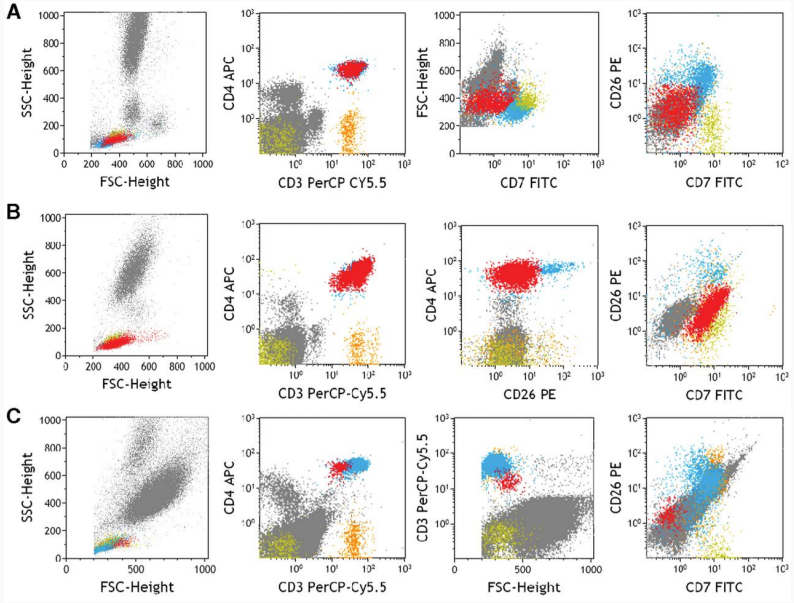


Pro: simple method to evaluate clusters in a single bivariate plot

Con: relies on the lower-dimensional embedding being a faithful representation of the data

Validating a cluster model

2 - Map cluster labels onto bivariate plots

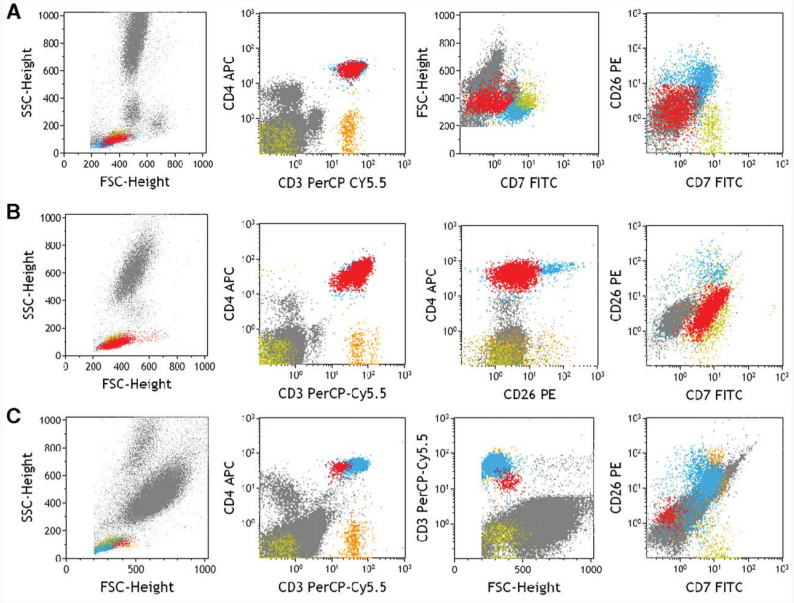


Pro: Allows us to incorporate our expert knowlege of biology

Con: Time consuming, impossible to interrogate all combinations

Validating a cluster model

3 - Plot expression data for each cluster



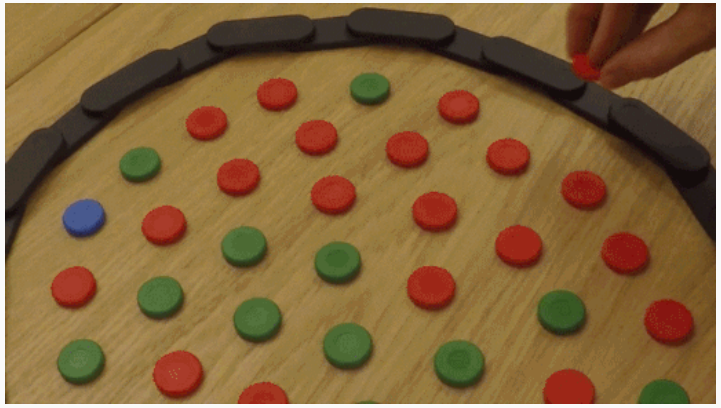
NEED TO REPLACE THIS IMAGE WITH ONE FROM PHIL **Pro:** Allows us to incorporate our expert knowlege of biology

Con: Time consuming, impossible to interrogate all combinations

What if the clustering model doesn't fit well?

If the model **under-clusters**, increase the number of clusters, duh.

If the model **over-clusters**, decrease the number of clusters, OR manually merge clusters you believe represent the same cell type.



Practical tips for dimension reduction and clustering

Practical tips

1 - Clean and transform[1] your data first!

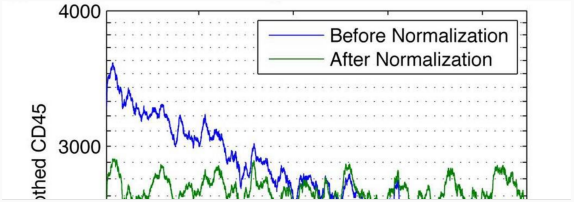
2 - Merge .fcs files together, analyse, then split apart later

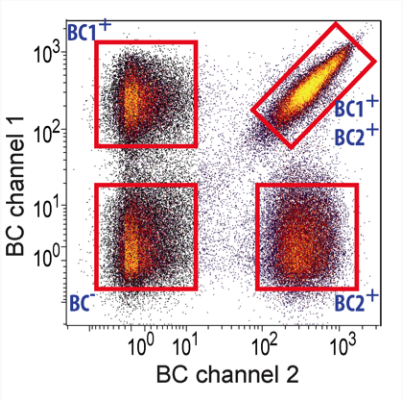
3 - Downsample if necessary

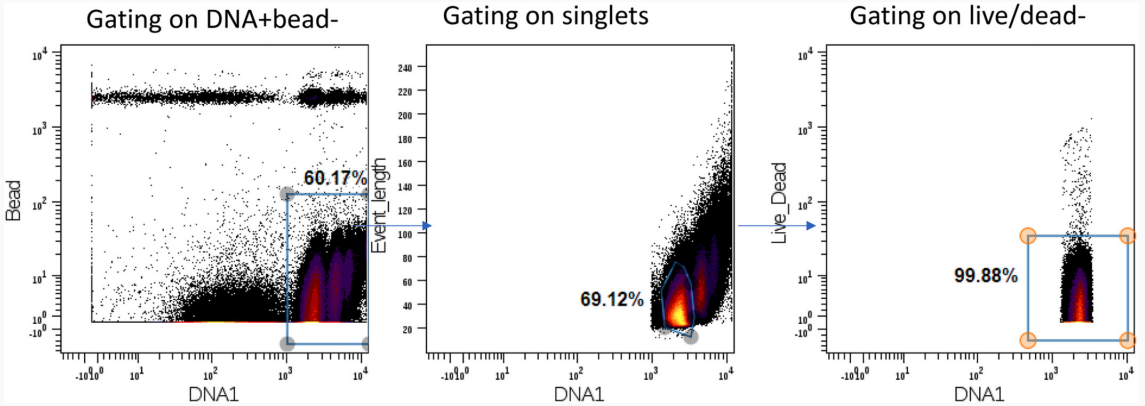
1 - or let FlowSOM do this for you (it uses logicle)

1 - Clean and transform your data

Bead normalization, de-barcoding, bead exclusion, doublet exclusion, dead cell exclusion

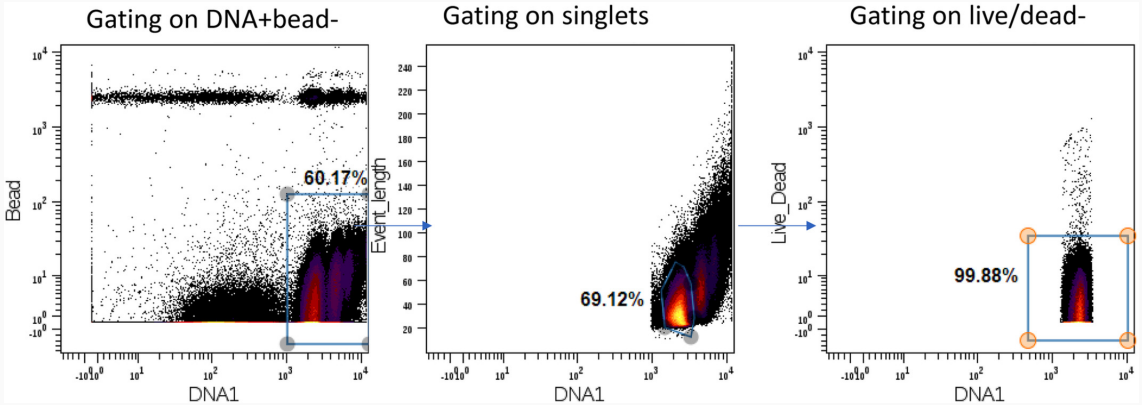






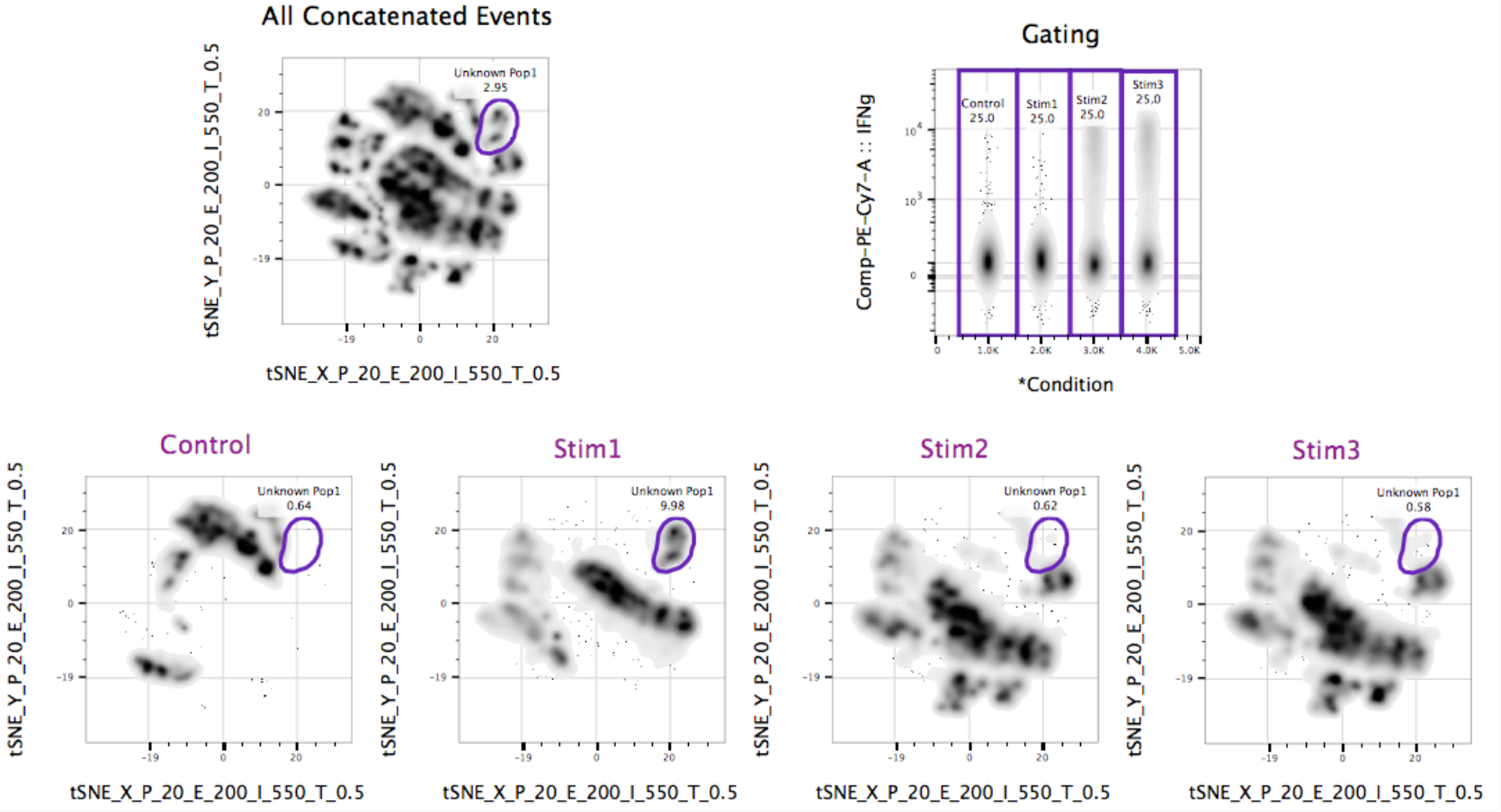
Check out the cytofclean R package: (https://github.com/JimboMahoney/cytofclean)

1 - Clean and transform your data



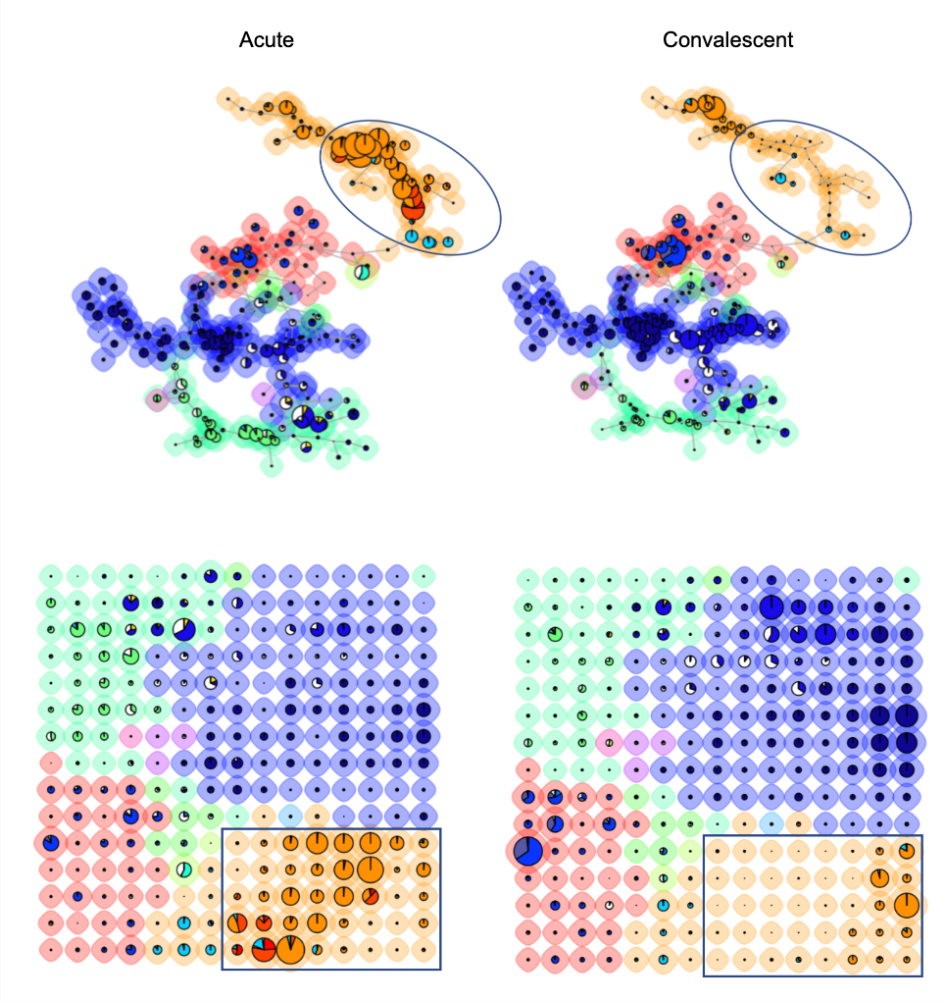
GET IMAGE FROM DINA SHOWING ARCSIN VS LOG TRANSFORMATION

2 - Merge .fcs files together

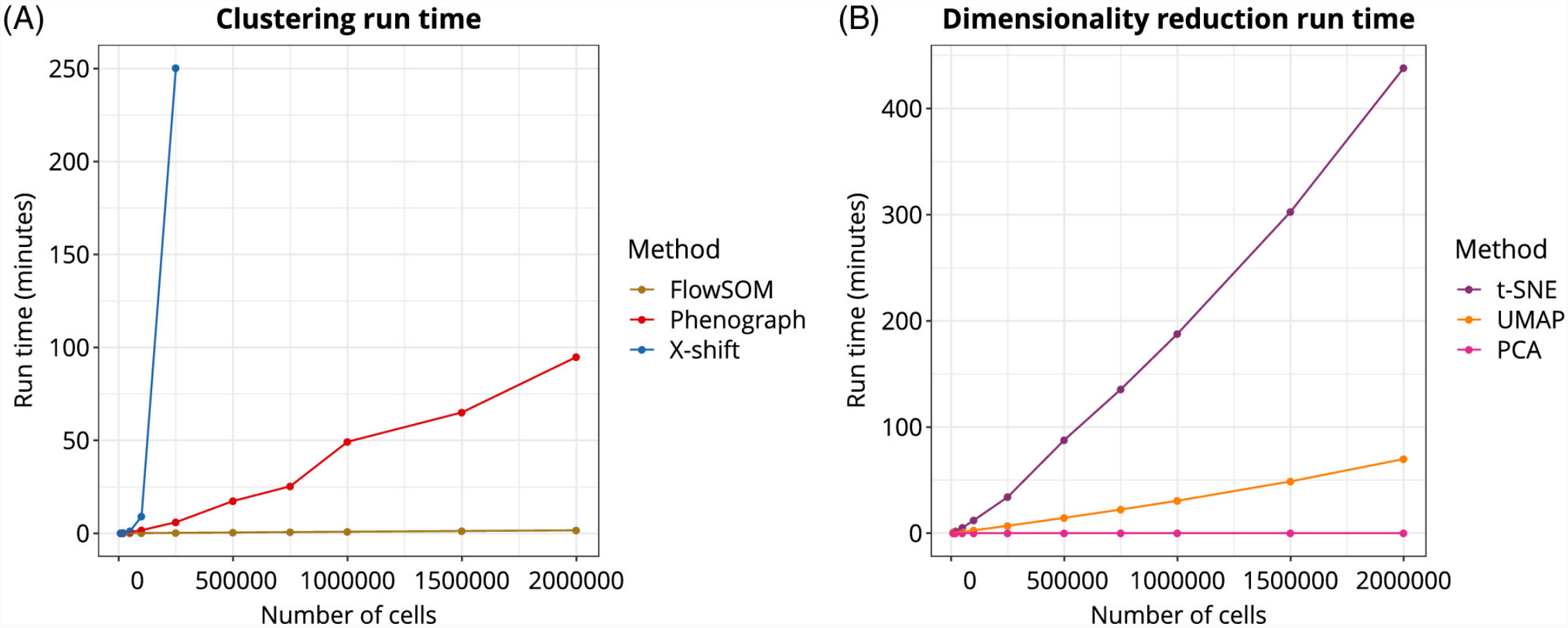


(https://www.flowjo.com/learn/flowjo-university/flowjo/tutorial/31)

2 - Merge .fcs files together



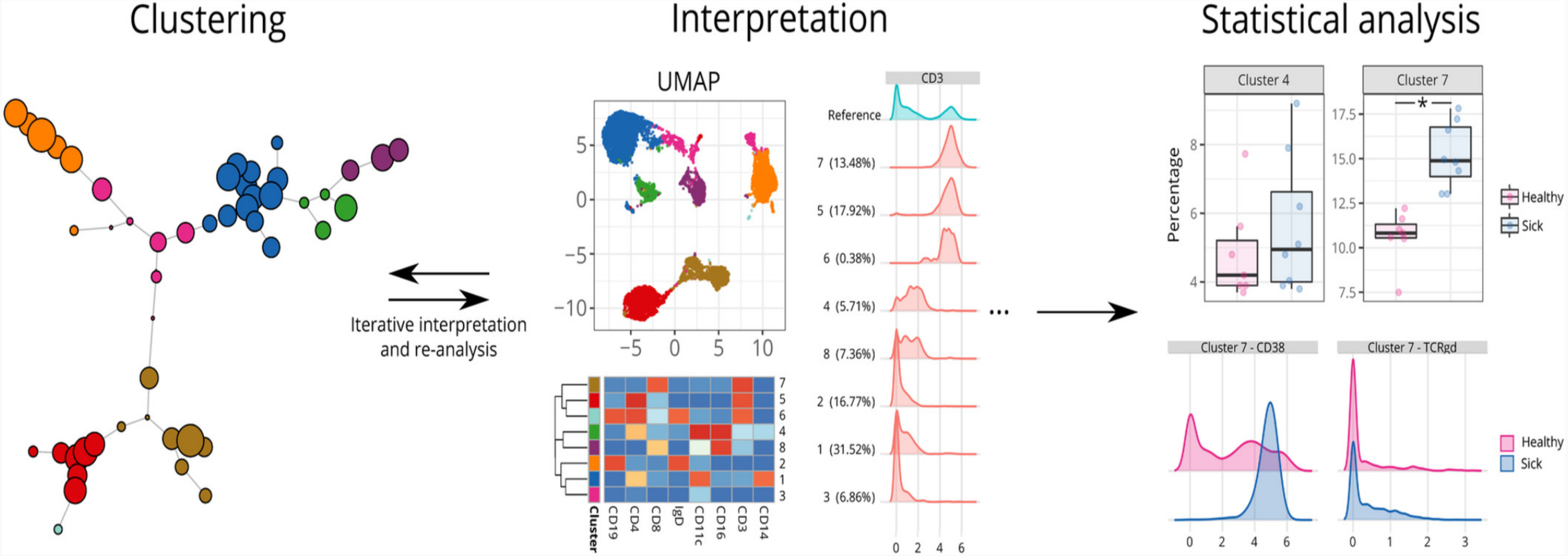
3 - Downsample if necessary



Pedersen & Olsen, 2019

The big picture

Overview of high-dimensional analysis



Pedersen & Olsen, 2019

Thanks for your attention!