Clustering Algorithms in Bioinformatics

Evangeline Kim
BINF6250 Module 8
The Roux Institute at Northeastern

General Overview

1. Introduction to Clustering
2. Applications in Bioinformatics
   a. scRNAseq
3. Basic Clustering Algorithms and Evaluation Metrics
   a. Hierarchical
   b. K-Means
   c. DBSCAN
4. Dimensionality Reduction
   a. PCA
   b. tSNE
   c. UMAP
5. Application and Conclusions

Introduction

What is Clustering?

Clustering is a method used for grouping a set of objects or data points into clusters based on their similarities.The goal is to ensure that data points within the same cluster are more similar to each other than to those in other clusters. This is then used to identify patterns and structures within the dataset.

Importance in Analyzing Biological Data

Hidden Relationships

Clustering different samples with similar gene expression might help find new common characteristics, discovering hidden relationships.

Subtypes

Clustering can help identify subtypes in genomic data! In cancer research, it might help distinguish between different cancer types which is useful when choosing treatments.

Biological Processes

Clustering can help map out pathways, showing how genes or proteins interact within a cell. This gives us a better understanding of biological processes.

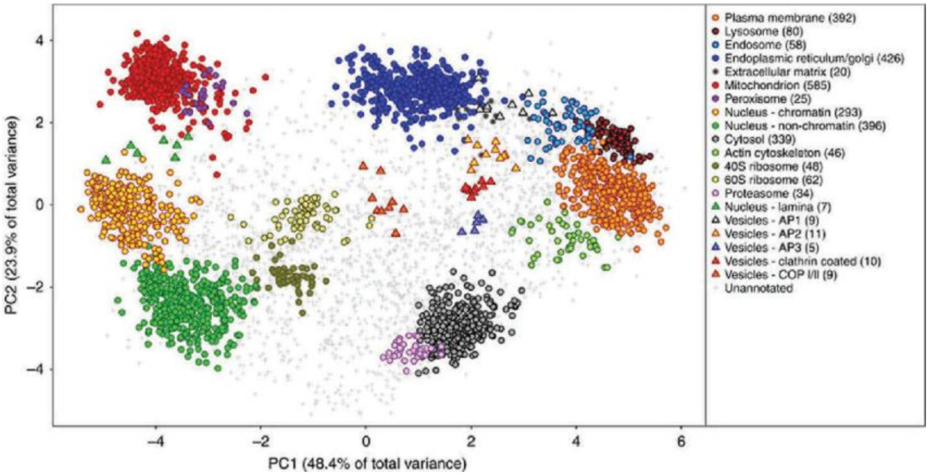
Biomarkers/Drugs

We can cluster gene expression changes that correlate with a disease's progression or even cell responses to various treatments. This can lead to new biomarker or drug discoveries!

Applications in Bioinformatics

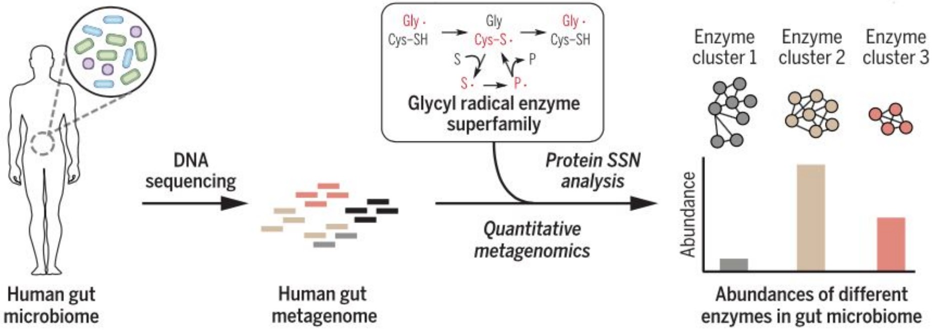
Some applications include…

- Gene Expression Analysis
  - Clustering genes based on expression profiles to discover genes that work together or are regulated by the same factors (more later).
- Proteomics
  - Grouping similar proteins to understand biological pathways.
- Phylogenetics
  - Using clustering to construct phylogenetic trees that represent relationships among species.



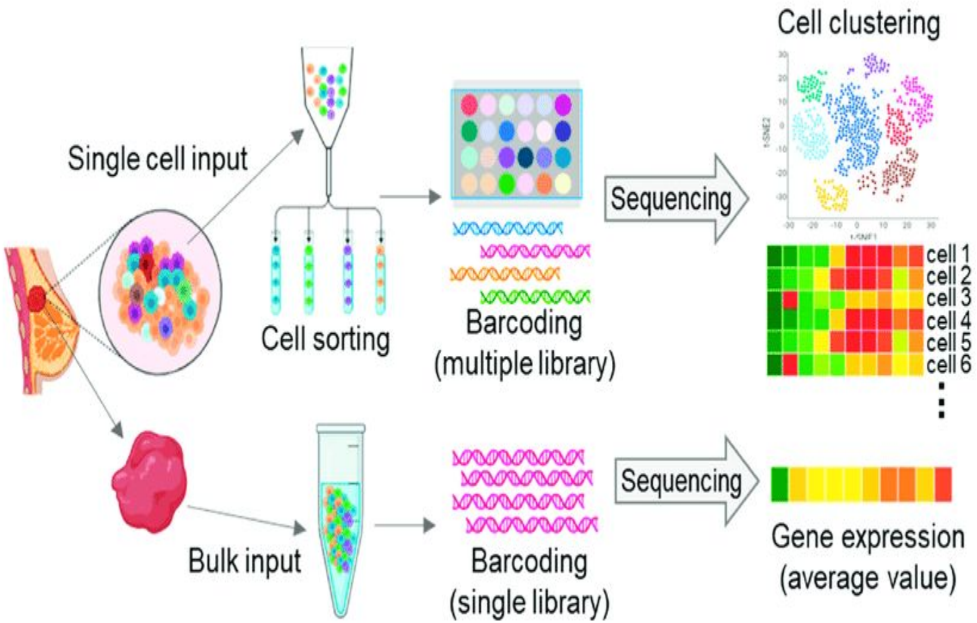
Applications cont.

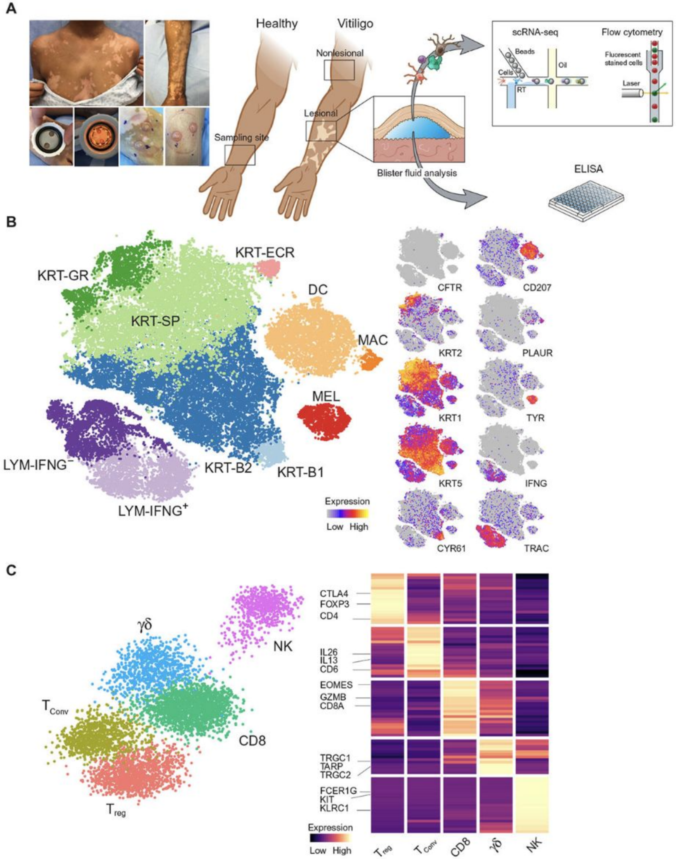
- Metagenomics
  - Clustering DNA from environmental samples to identify and compare the types of microorganisms (species) present in different environments.
- Single-cell RNA sequencing (scRNAseq)
  - Clustering data from *individual* cells to identify different cell types and their states within a complex tissue or population.
  - This technique allows one to study the diversity and functions of cells *one by one,* even when there are many types of cells in a sample.

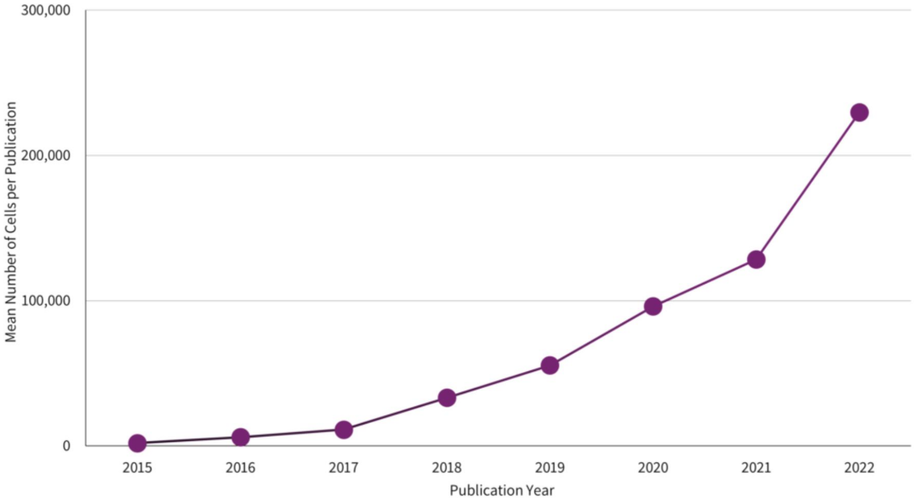


More scRNAseq. Why? Because I think it’s cool…

- Breaks down sample into individual cells. Can measure activity of each cell’s genes and behavior when “off” or “on”.
- Isolate cells → Capture RNA by cell → Sequence → Process and analyze!
- Very recent. A trending topic.



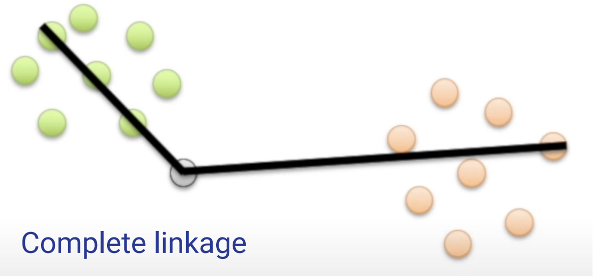

A
Healthy
Vitiligo
Nonlesional
Lesional
Sampling site
Blister fluid analysis
scRNA-seq
Beads
Cells
Oil
RT
Flow cytometry
Fluorescent stained cells
Laser
ELISA
B
KRT-GR
KRT-ECR
KRT-SP
DC
MAC
MEL
LYM-IFNG^{-}
LYM-IFNG^{+}
KRT-B2
KRT-B1
Expression
Low
High
CFTR
CD207
KRT2
PLAUR
KRT1
TYR
KRT5
IFNG
CYR61
TRAC
C
γδ
NK
T_{Conv}
CD8
T_{reg}
CTLA4
FOXP3
CD4
IL26
IL13
CD6
EOMES
GZMB
CD8A
TRGC1
TARP
TRGC2
FCER1G
KIT
KLRC1
Expression
Low
High
T_{reg}
T_{Conv}
CD8
γδ
NK


300,000
200,000
100,000
0
Mean Number of Cells per Publication
2015
2016
2017
2018
2019
2020
2021
2022
Publication Year

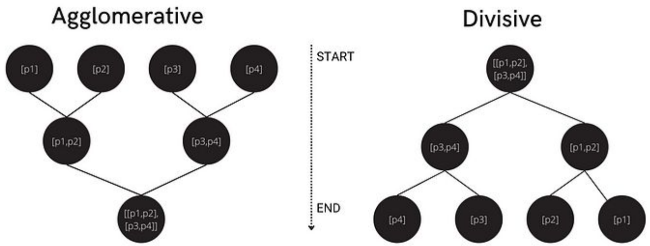
Basic Clustering Algorithms

Hierarchical Clustering

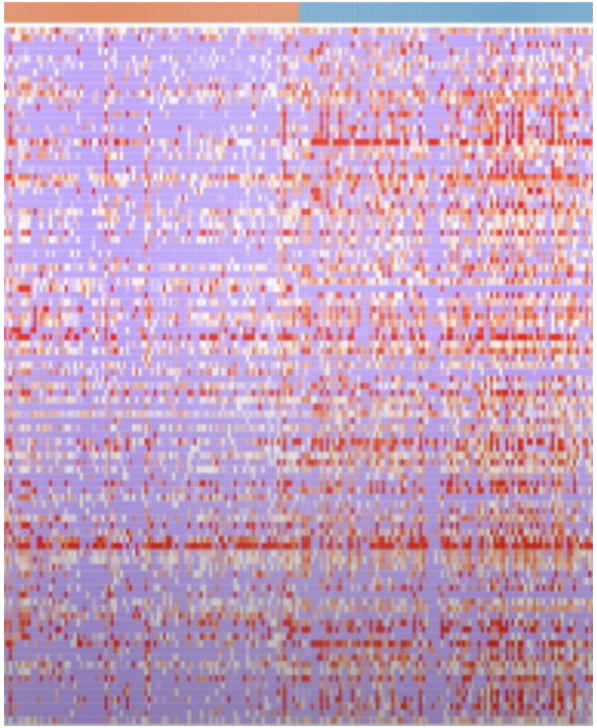
- Hierarchical clustering is based on the idea that nearby objects are more similar than further away objects. Clusters are formed based on their distance.
- Agglomerative vs. Divisive
- Different distance metrics determine how the clusters will be formed.
  - Euclidean, manhattan, cosine, etc.
- Linkage methods determine how the distance between clusters is calculated when merging them.
  - Average, single, complete, UPGMA, NJ, etc.



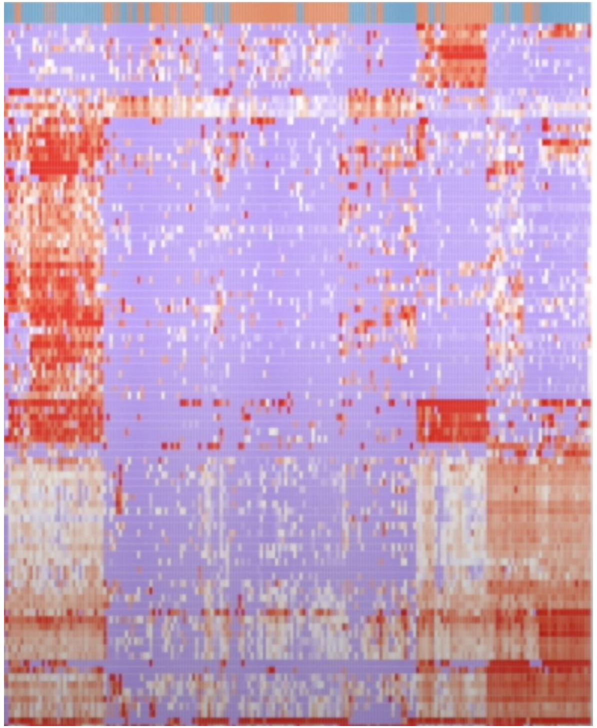
Complete linkage



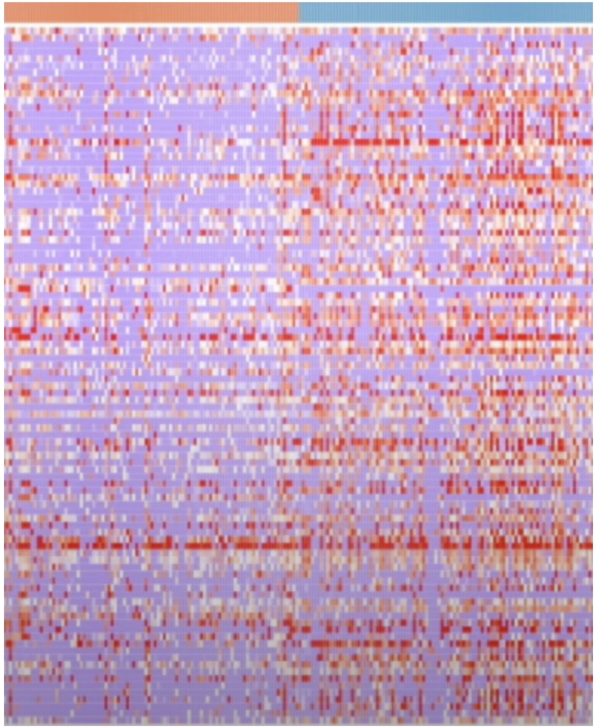
Without Hierarchical Clustering



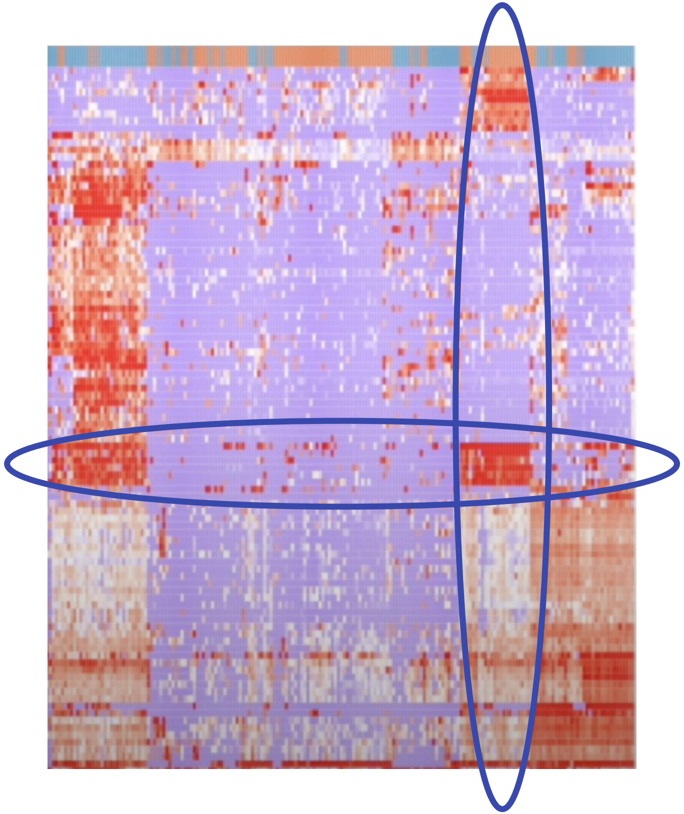
With Hierarchical Clustering



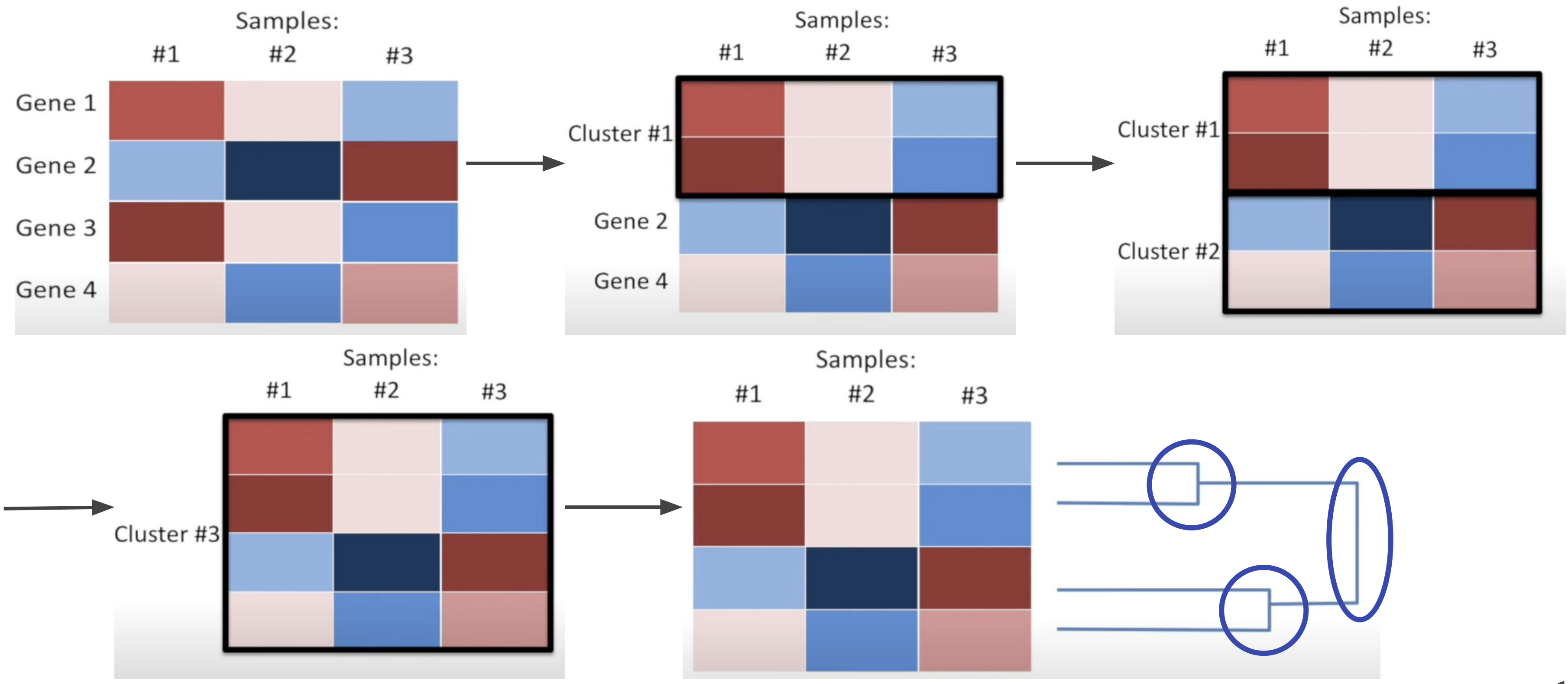
Without Hierarchical Clustering



With Hierarchical Clustering



Hierarchical Clustering



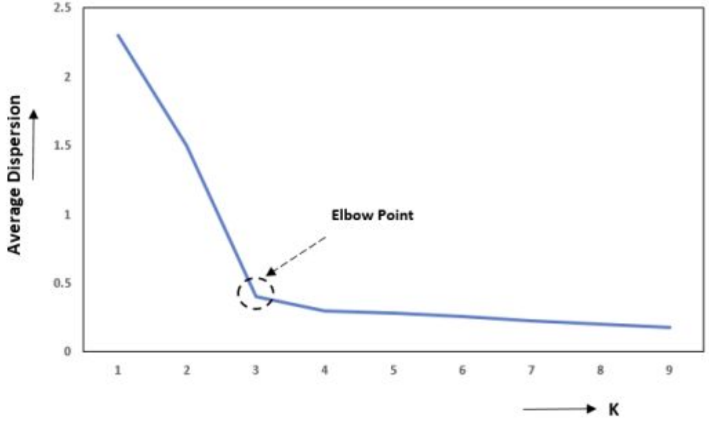
K-Means Clustering

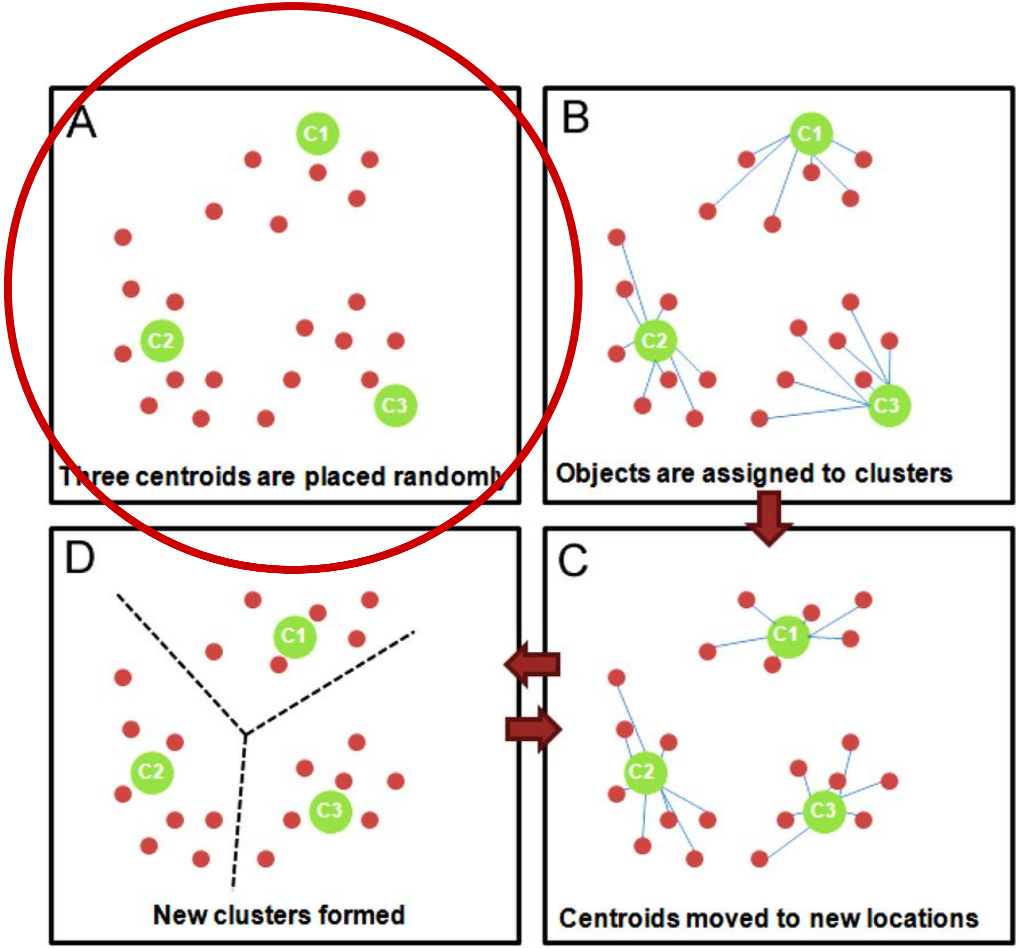
- This is a widely used algorithm that groups data into “K” clusters based on similarity. Unsupervised.
- The algorithm randomly selecting K data points as the initial centroids. Each data point is assigned to the nearest centroid based on a distance metric. Next, the centroids are updated by calculating the mean position of all data points assigned to each cluster. The previous steps are then repeated iteratively until either the centroids stabilize (convergence) or the max set number of iterations is reached. You can then repeat the entire k-means process until you get the best result with the lowest sum of variance.
- Elbow method to find optimal K

Choosing K and initializing the centroids

There are different methods to choose the optimal “K”

Elbow Method for selection of optimal “K” clusters





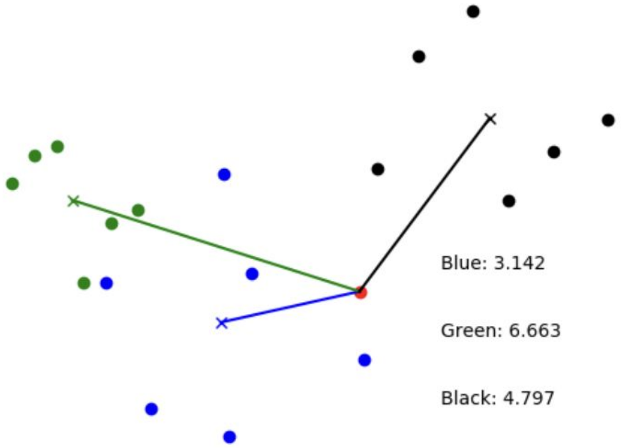
Three centroids are placed randomly

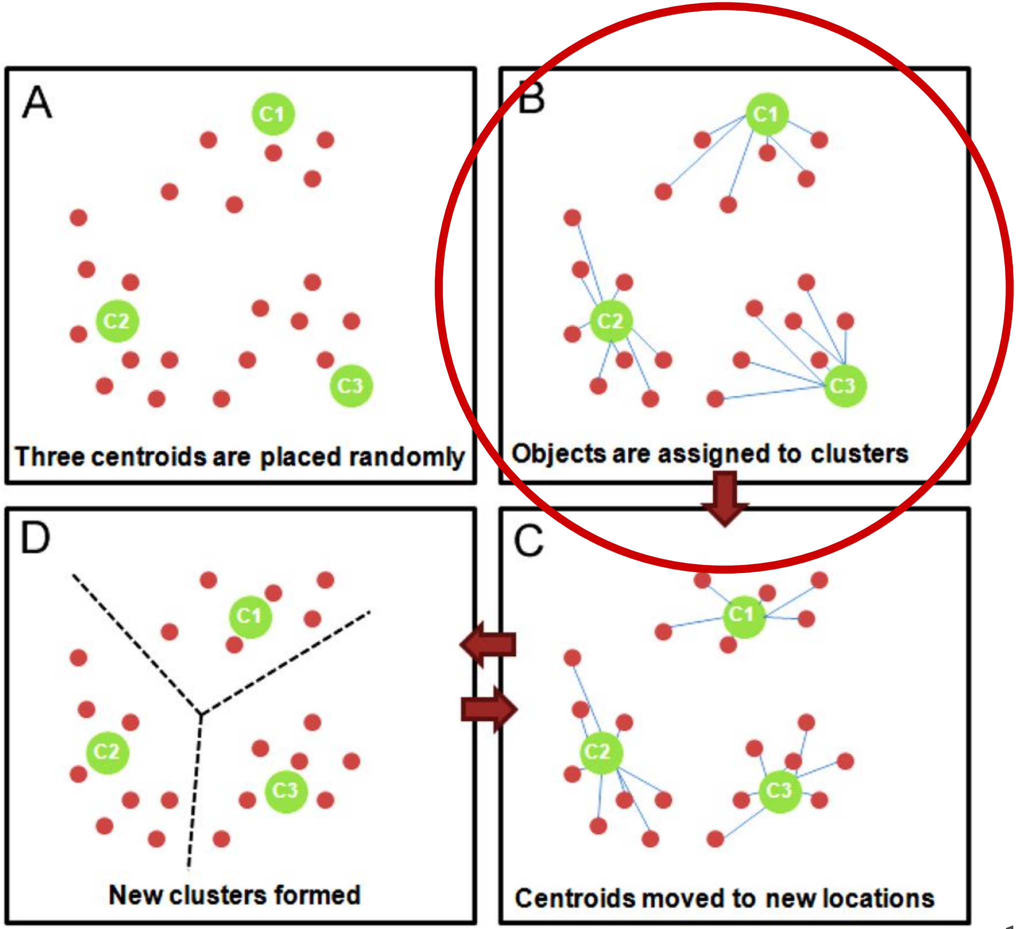
Objects are assigned to clusters

New clusters formed

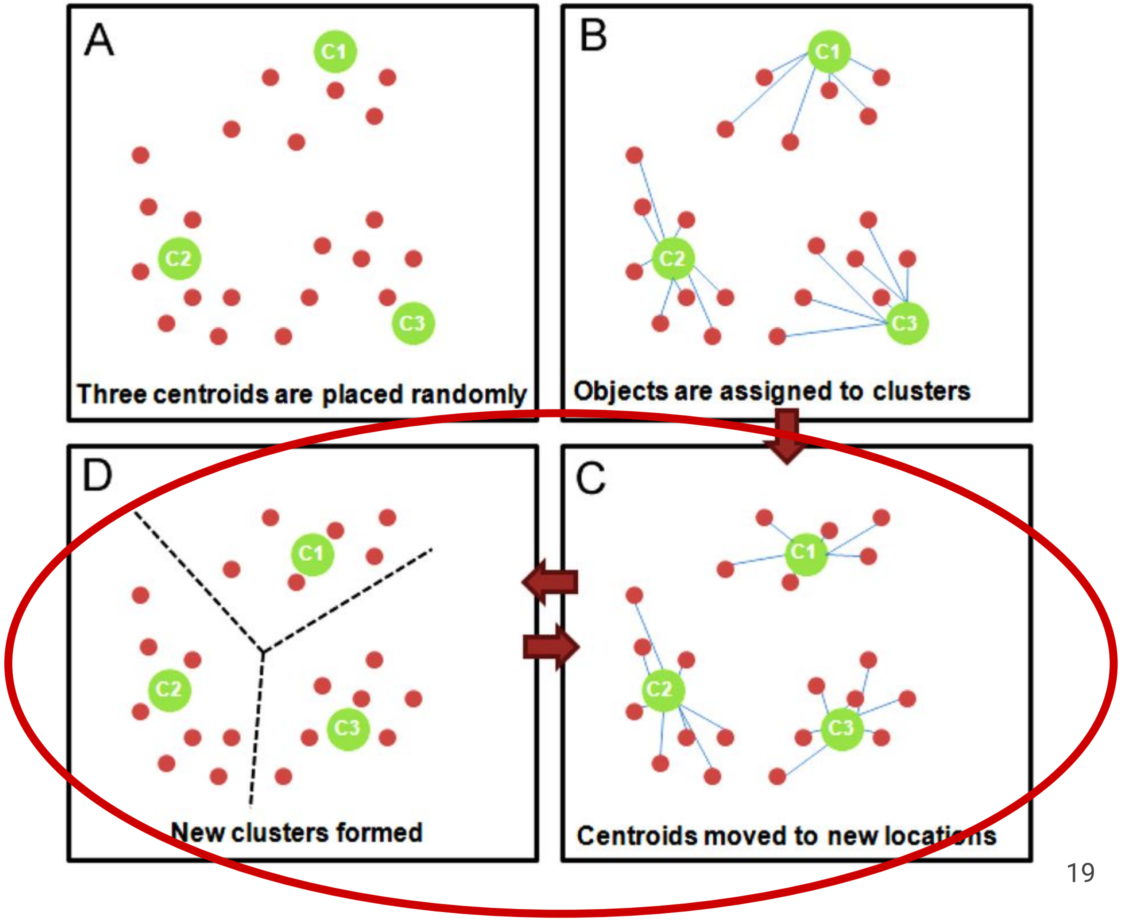
Centroids moved to new locations

Assign data points to the centroids to create the initial clusters. Usually based on euclidean distance, but other metrics can be used.




A
C1
C2
C3
Three centroids are placed randomly
B
Objects are assigned to clusters
C
Centroids moved to new locations
D
New clusters formed

Recalculating/updating the centroids to create the new clusters by computing the mean of all points assigned to that cluster


A
C1
C2
C3
Three centroids are placed randomly
B
C1
C2
C3
Objects are assigned to clusters
D
C1
C2
C3
New clusters formed
C
C1
C2
C3
Centroids moved to new locations

Say that we have 2 clusters…

Cluster 1 points: (1, 2), (2, 3), (3, 3)
Cluster 2 points: (8, 8), (9, 9), (10, 9)

We would calculate the new centroids as the mean of the points in each cluster…

CENTROID 1

$$\left(\frac{1+2+3}{3}, \frac{2+3+3}{3}\right) = (2, 2.67)$$

CENTROID 2

$$\left(\frac{8+9+10}{3}, \frac{8+9+9}{3}\right) = (9, 8.67)$$

Then we reassign those points to the new centroids to make the new clusters!

(1, 2) closer to (2, 2.67), so it stays in Cluster 1

(2, 3) closer to (2, 2.67), so it stays in Cluster 1

(3, 3) closer to (2, 2.67), so it stays in Cluster 1.

(8, 8) closer to (9, 8.67), so it stays in Cluster 2.

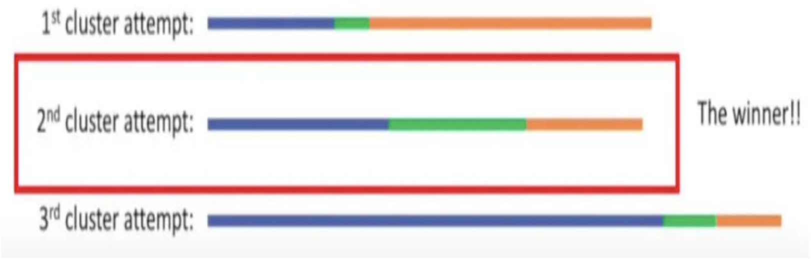
(9, 9) closer to (9, 8.67), so it stays in Cluster 2.

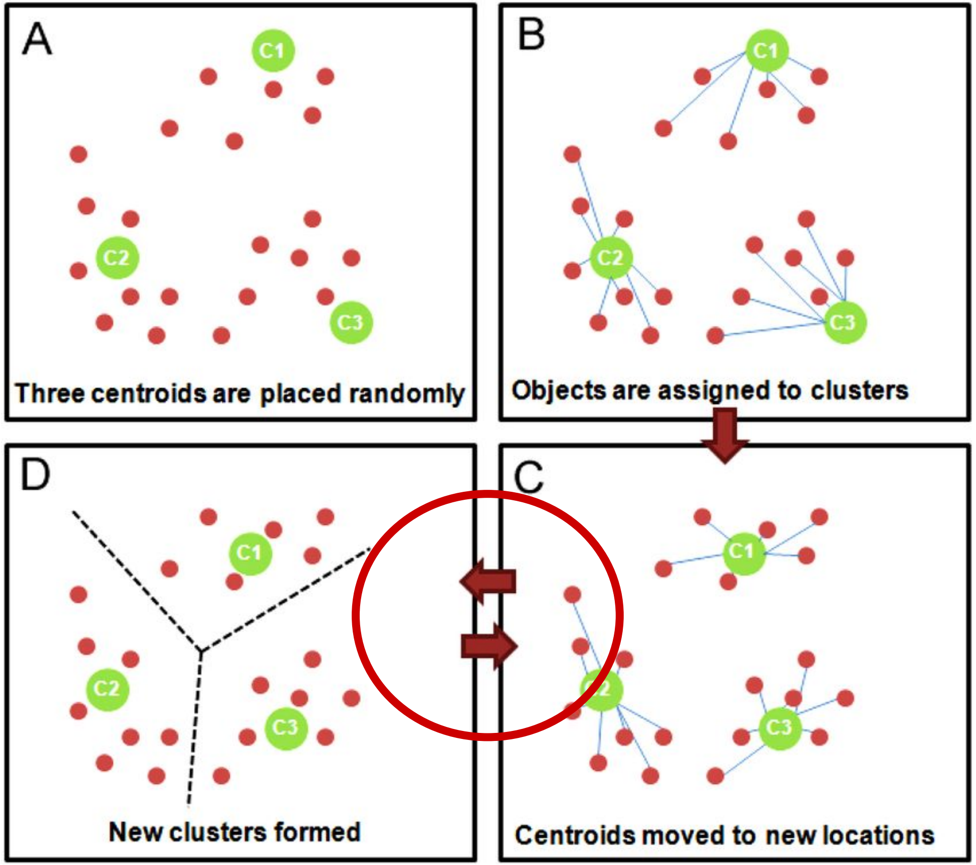
(10, 9) closer to (9, 8.67), so it stays in Cluster 2.

In this case, all points stayed in the same cluster (this is known as convergence)

Repeating the assignment and updating steps until the centroids stop changing (convergence)

You can repeat the entire k-means process until you get the best run/result with the lowest sum of variance.

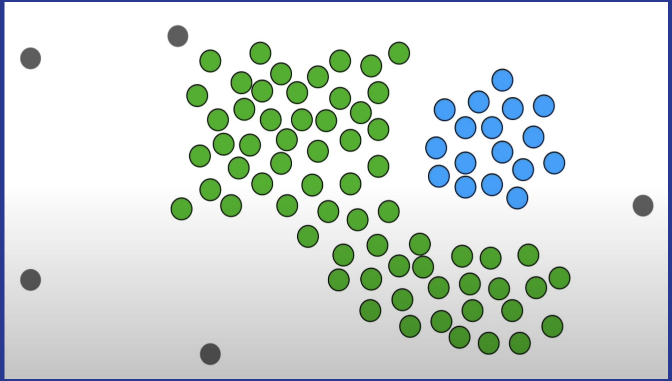



A
C1
C2
C3
Three centroids are placed randomly
B
C1
C2
C3
Objects are assigned to clusters
D
C1
C2
C3
New clusters formed
C
C1
C2
C3
Centroids moved to new locations

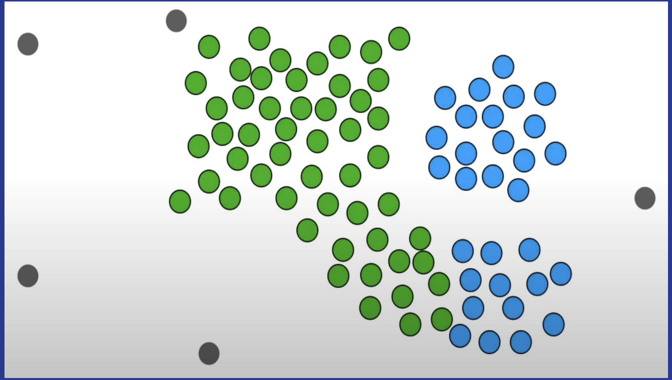
Fuzzy K-Means

- An extension of K-means but allows data points to belong to multiple clusters with different degrees of membership (probability score from 0-1). Centroids are now weighted means.
- Better to use this with complex/overlapping data, and often useful in bioinformatics. For example, in scRNAseq data, cells often have mixed characteristics which causes overlap. Similarly in gene expression, genes are involved in numerous biological processes and can belong to multiple clusters.
- Hard vs. Soft Clustering

But what if the clusters are nested and we keep getting weird results??



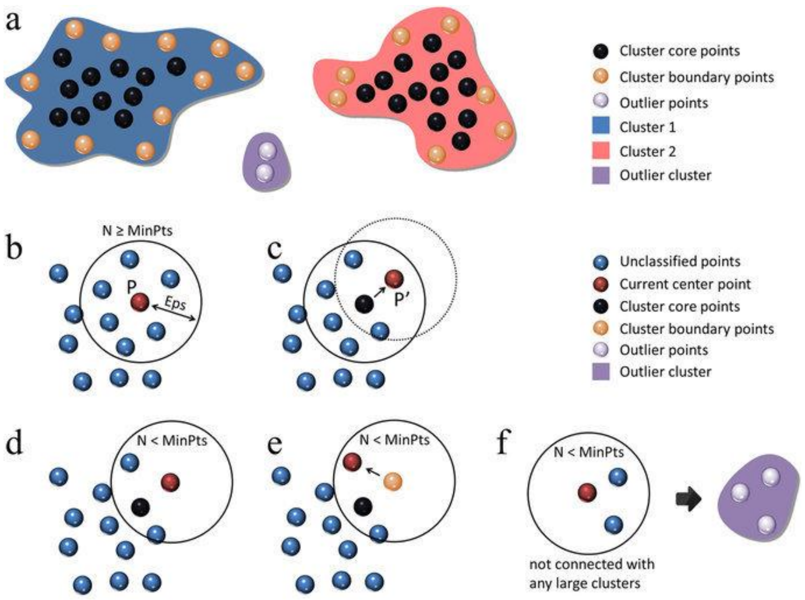
:D



:'(

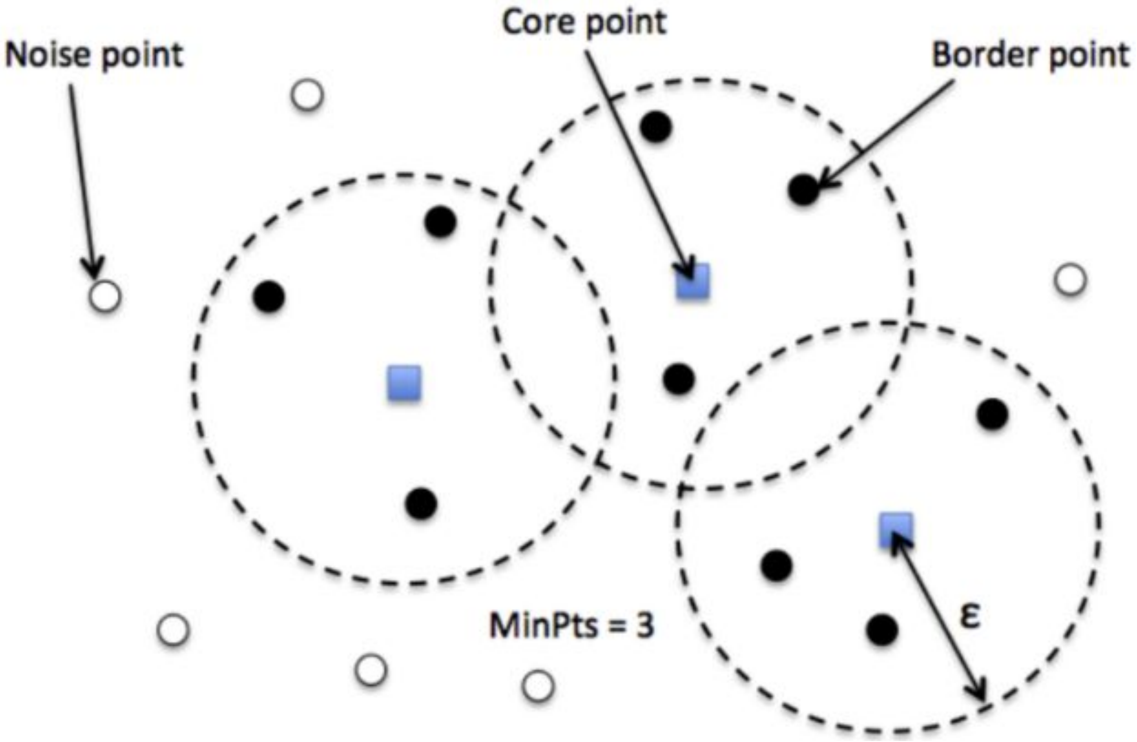
Density-Based Spatial Clustering of Applications with Noise (DBSCAN)

- DBSCAN is a density-based clustering algorithm that clusters data points based on their spatial density, allowing it to find clusters of non-uniform shapes and effectively deal with noise (outliers). Unlike K-means, which required the number of clusters to be predefined, DBSCAN can find the number of clusters automatically based on the density of the data points.

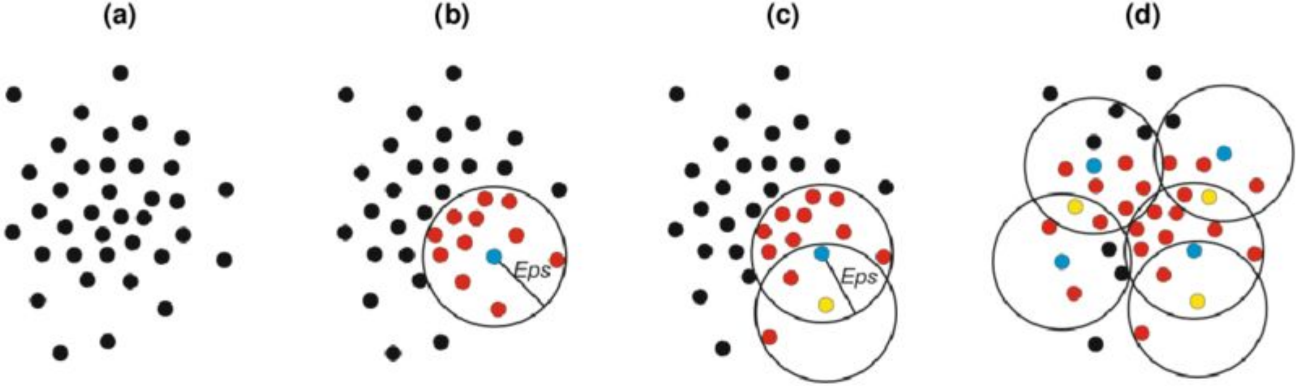


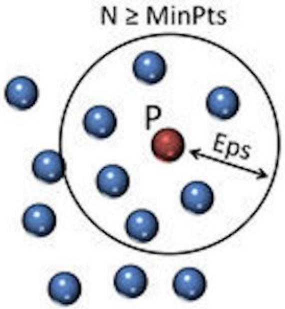
Terminology

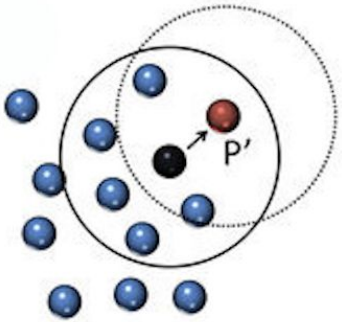
- Core Points
- Border Points
- Noise Points (outliers)
- **MinPts**
- **Epsilon radius (ε)**



We start by checking every point in the data set and deciding whether or not it is a core point. This is done by counting how many points fall within the epsilon radius. Points that have at least the *MinPts* overlapping the circle are core points!





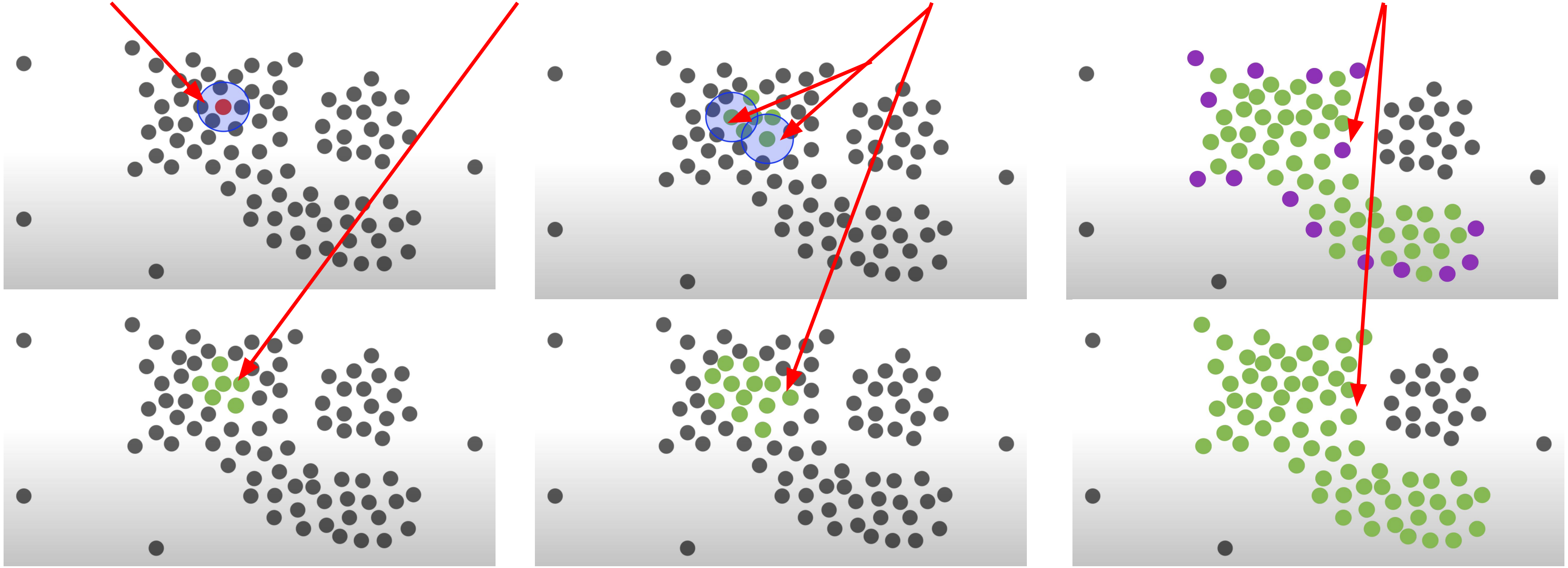


Randomly pick a core point and assign it to the first cluster

Core points within the epsilon circle are added to that cluster!

Neighboring core points join and extend the first cluster

Non-core points close to core points also join, but do not extend

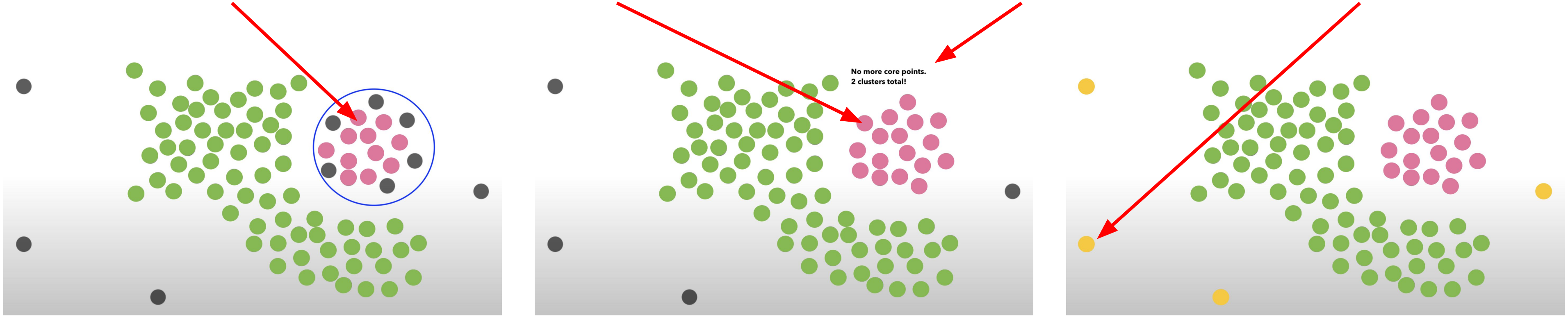


Now look at other core points that weren’t added to the first cluster!

These can form a second cluster along with the non-core points close by.

Once all core points are in clusters, we are done making new ones!

Any non-core points not in clusters are noise/outliers.



Because clusters are formed sequentially, if you have a non-core point close to multiple clusters, it would join the first cluster that was being formed because it is close to a core point.

Evaluation Metrics. AKA, Do my clusters suck?

- **Silhouette Score** - How well each point fits into its cluster compared to other points. Widely used. How compact/separated the clusters are. [-1, 1]
- **Adjusted Rand Index (ARI)** - compares predicted vs true labels and adjusts for random chance. Supervised evaluation. [-1, 1]
- **Davies-Bouldin Index (DBI)** - Checks if clusters are well separated/compact. Average similarity ratio b/w each cluster and the most similar cluster. Lower=better.
- **Inertia/Elbow method** from K-Means, etc.

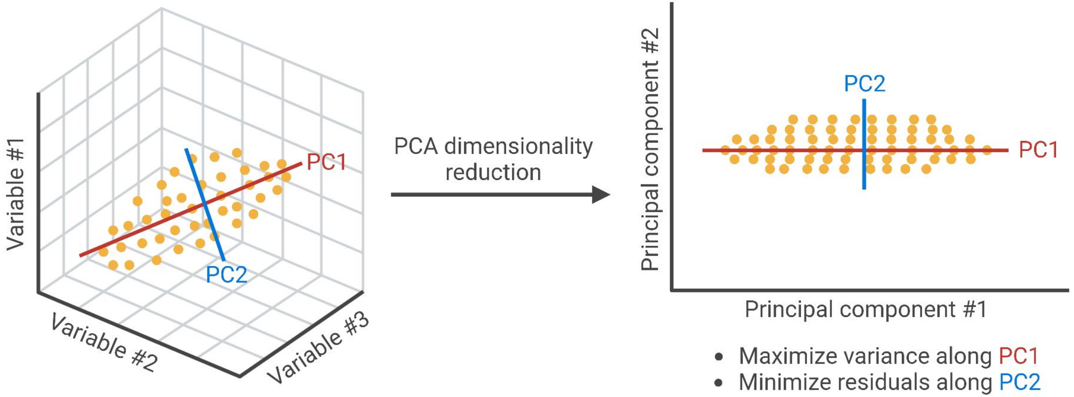
Dimensionality Reduction

What is it and why do we care?

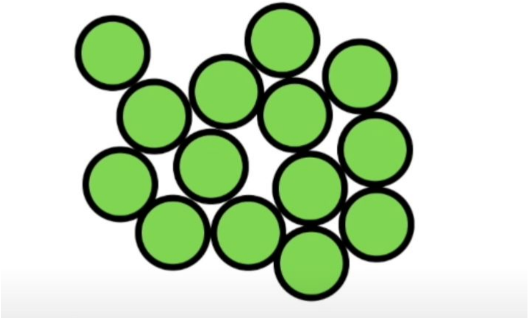
- Dimensionality reduction is a process that simplifies complex data by reducing the number of features in a dataset while keeping as much of its essential structure, information, and characteristics as possible. This transformation moves the data from a high-dimensional space to a lower-dimensional space, making it easier to visualize, analyze, and interpret.
  - The “curse of dimensionality”
  - Computational complexity
  - Improving batch effects
- Trade-offs
  - Interpretability
  - Information Loss
  - Need to choose the right method

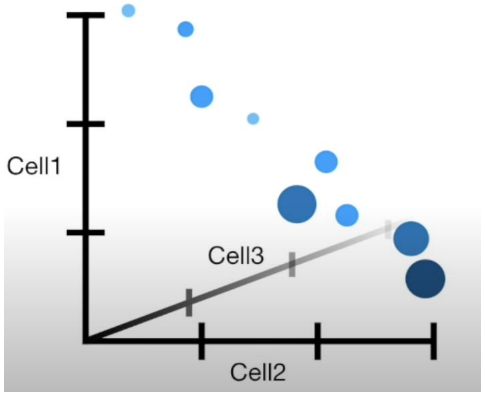
Principal Component Analysis (PCA)

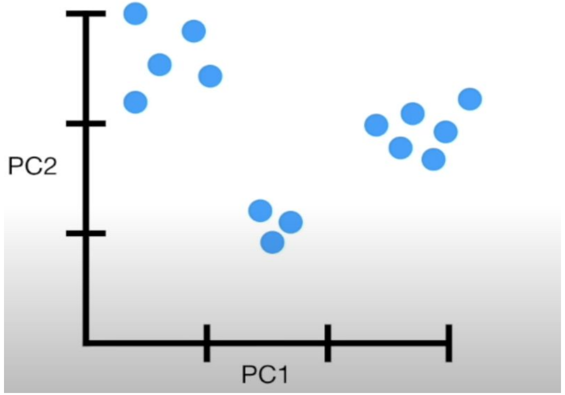
- Creates a new set of features (principal components/PCs) that are combinations of the original features.
- Measures variance and rearranges the data along axes where the variance is the largest. PCA tries to find the axes where the data varies the most. PC1 will have the highest variance, then PC2, 3, etc…

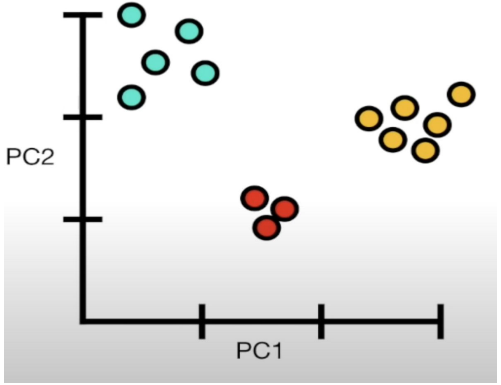


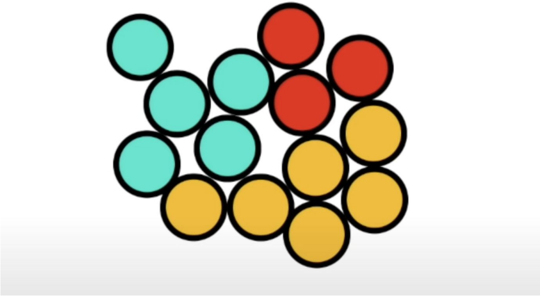
- Maximize variance along PC1
- Minimize residuals along PC2



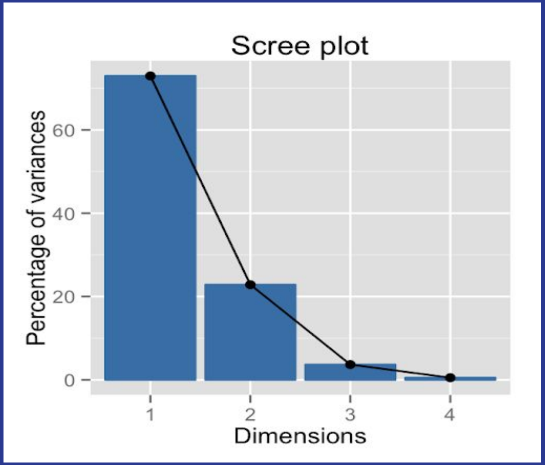

Cell1
Cell3
Cell2


PC2
PC1

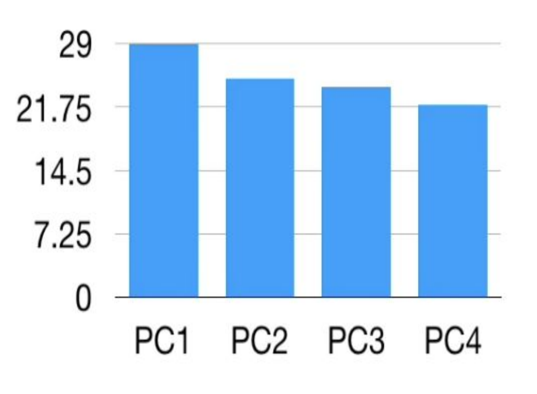

PC2
PC1



But what if the data isn't linear or the PCs aren't capturing enough variability?



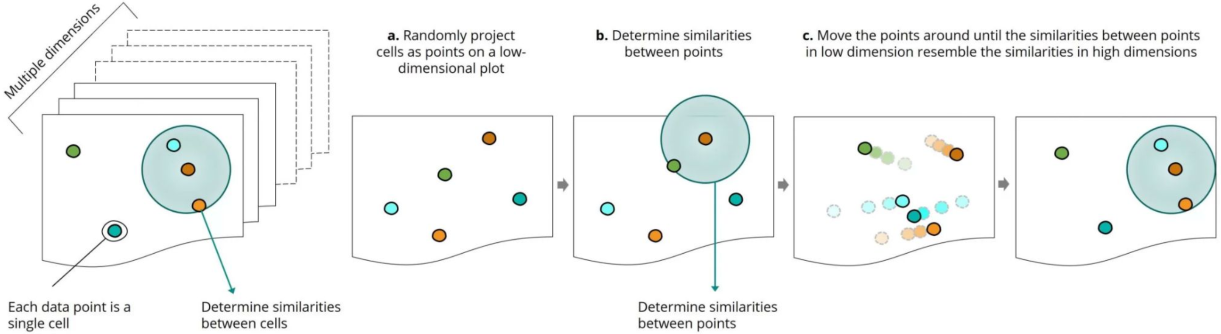
:D



:'(

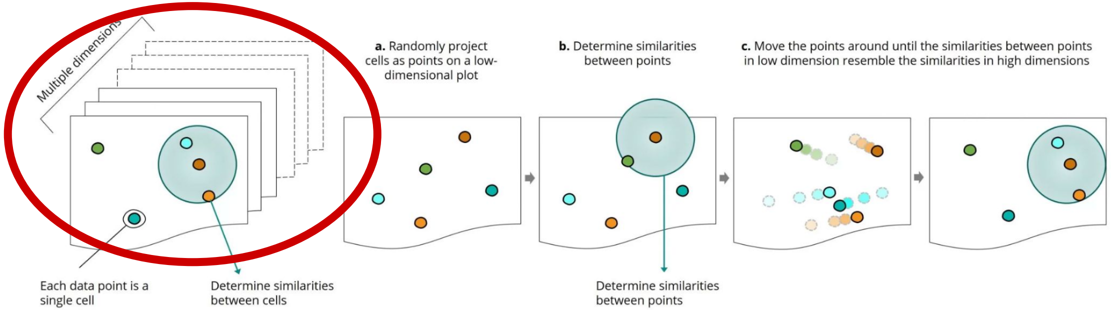
t-Distributed Stochastic Neighbor Embedding (t-SNE)

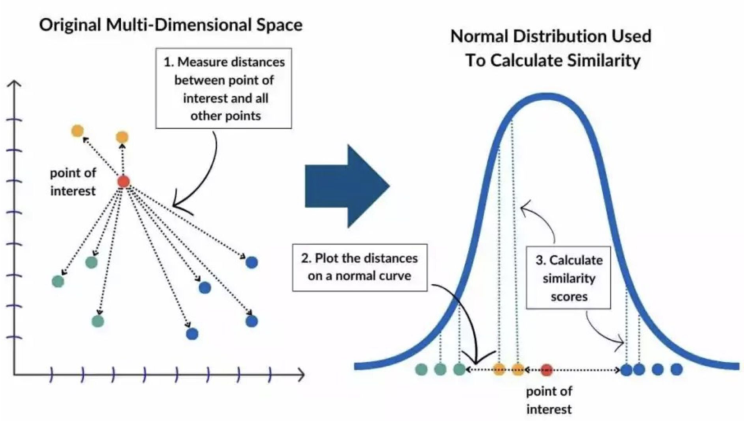
- Focuses on local similarities (usually distance based) between points to preserve local structure. Great for visualization in 2D/3D, especially if you're looking for clusters or patterns in data that might be non-linear
- Unlike PCA, t-SNE components correspond to the new dimensions that have been optimized to preserve the local structure between points.
- t-SNE wants to ensure that points close to each other in higher dimensions remain close to each other in the lower dimensions.



Multiple dimensions
a. Randomly project cells as points on a low-dimensional plot
b. Determine similarities between points
c. Move the points around until the similarities between points in low dimension resemble the similarities in high dimensions
Each data point is a single cell
Determine similarities between cells
Determine similarities between points

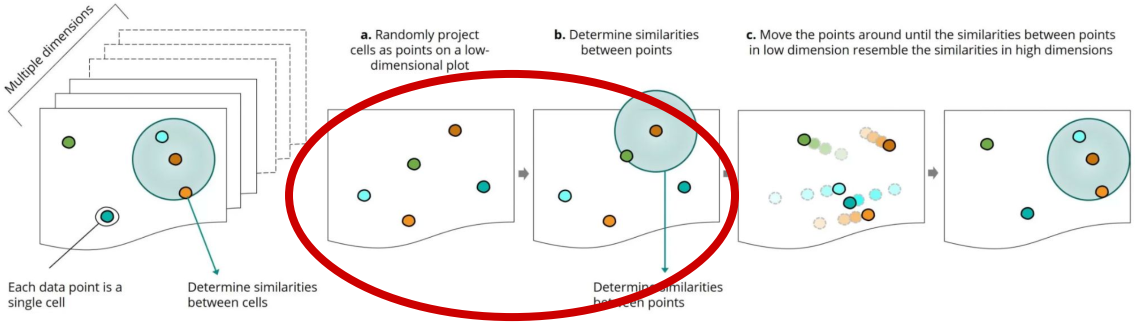
1. Calculate similarities in high dimensional space
2. Construct probability distributions

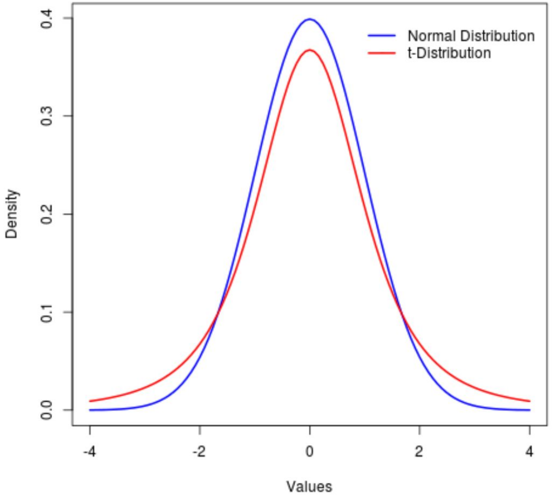

Multiple dimensions
a. Randomly project cells as points on a low-dimensional plot
b. Determine similarities between points
c. Move the points around until the similarities between points in low dimension resemble the similarities in high dimensions
Each data point is a single cell
Determine similarities between cells
Determine similarities between points


Original Multi-Dimensional Space
1. Measure distances between point of interest and all other points
point of interest
Normal Distribution Used To Calculate Similarity
2. Plot the distances on a normal curve
3. Calculate similarity scores
point of interest

t-SNE computes a probability distribution that reflects how similar each pair of points is in the high-dimensional space. t-SNE looks at the **neighbors** for each point. Then it calculates the probability of one point being a "**neighbor**" of another, based on distance. Closer points have higher probabilities of being neighbors.

3. Initialize random points in lower dimension
4. Calculate similarities in low dimensional space


Multiple dimensions
a. Randomly project cells as points on a low-dimensional plot
b. Determine similarities between points
c. Move the points around until the similarities between points in low dimension resemble the similarities in high dimensions
Each data point is a single cell
Determine similarities between cells
Determine similarities between points

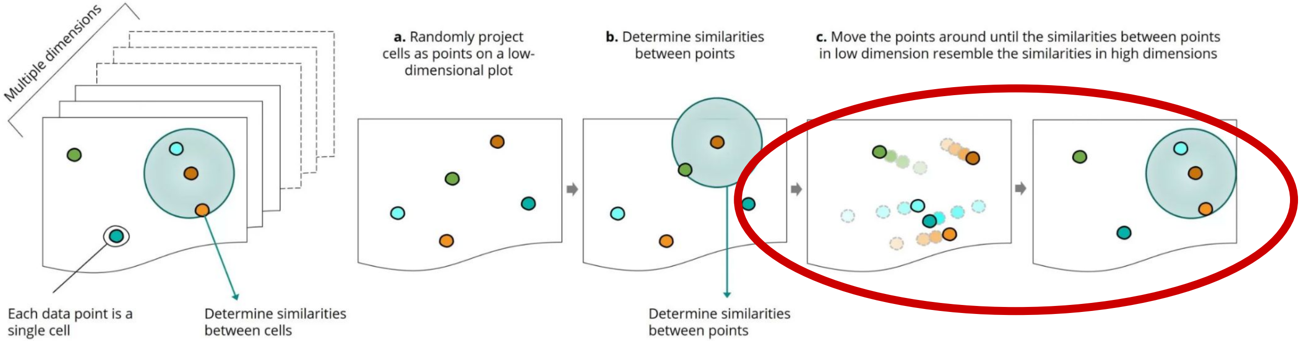


Initial **embedding** in the lower dimension. Randomly generated. Will later optimize and update these.

T-distribution used to compute similarities instead.

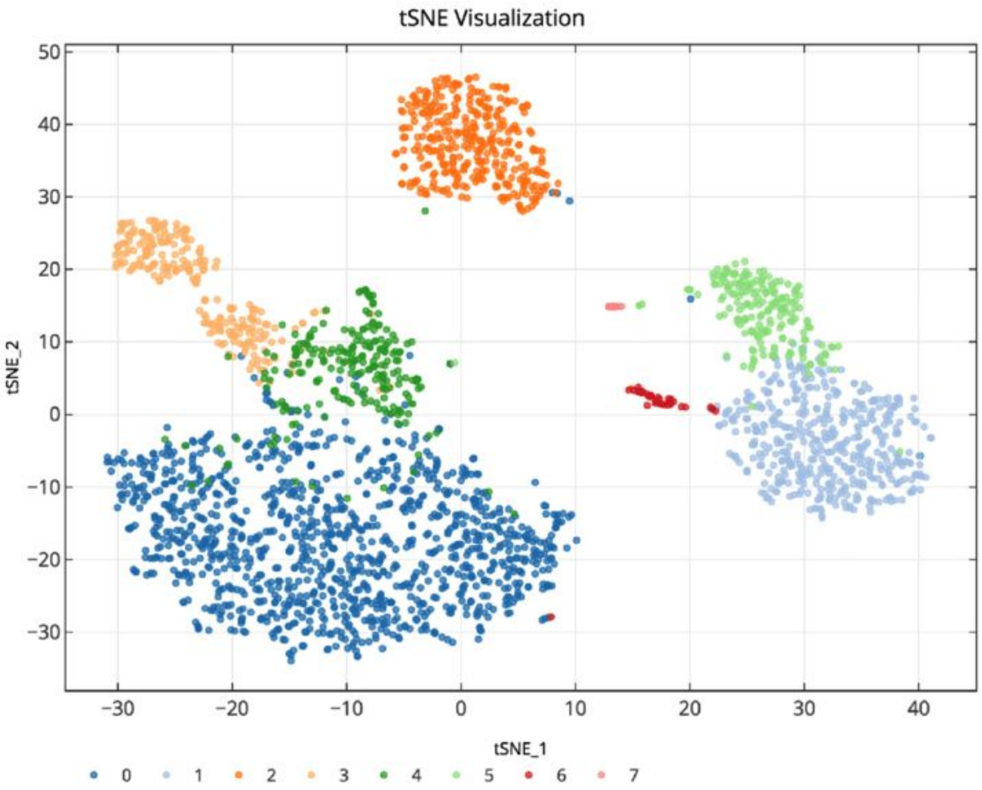
5. Optimize embedding by minimizing the Kullback-Leibler Divergence
6. Continue until convergence or max iterations reached

KL-divergence is a measure of how different the probability distributions are between the higher and lower dimensional spaces. Basically, t-SNE is using gradient descent to tweak the positions of points in the low-dimensional space, minimizing the difference between the high-dimensional and low-dimensional similarity distributions calculated earlier. Do this until convergence!


Multiple dimensions
a. Randomly project cells as points on a low-dimensional plot
b. Determine similarities between points
c. Move the points around until the similarities between points in low dimension resemble the similarities in high dimensions
Each data point is a single cell
Determine similarities between cells
Determine similarities between points

7. Pretty pictures in lower space!

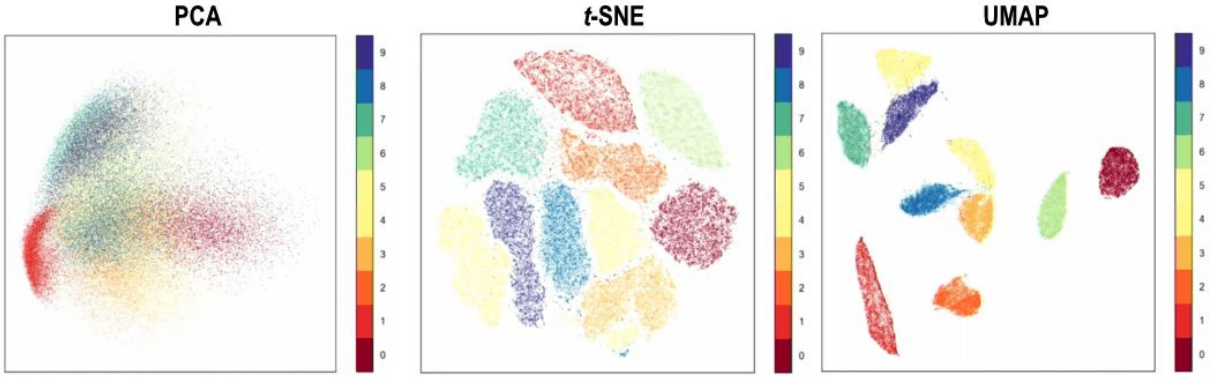
The final low-dimensional embedding where structure was preserved as much as possible.



tSNE1 and tSNE2 are the new optimized dimensions that preserve the structure..

Uniform Manifold Approximation and Projection (UMAP)

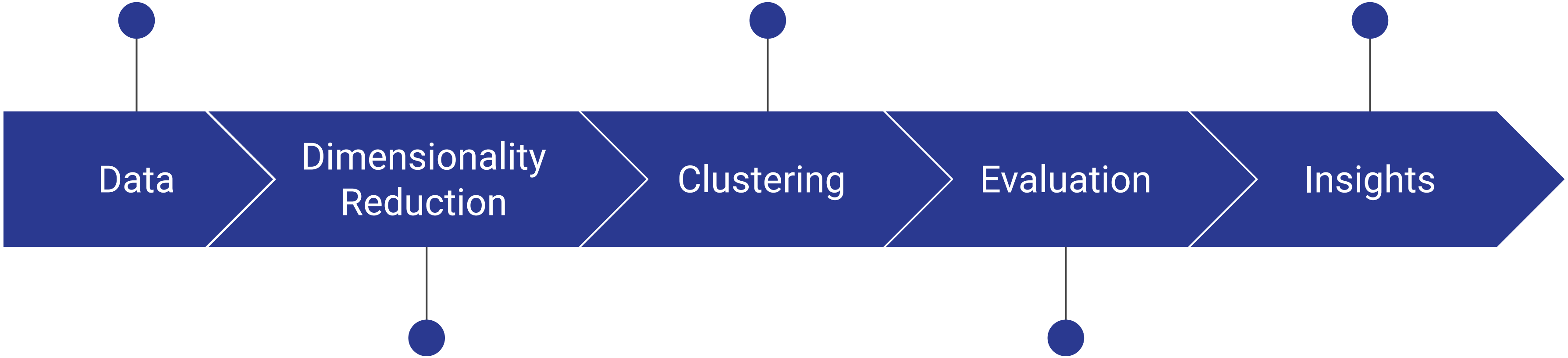
- Introduced by McInnes, Healy, and Melville in 2018 - very recent!
- UMAP is similar to t-SNE in that both construct graph representations in high dimensions and optimize a structurally similar low dimensional graph.
- t-SNE is great for visualization, but it focuses on local relationships and is less good at preserving global structure. It is also computationally intensive.
- UMAP is more flexible and can capture both local and global structures in the data. It's designed to be more scalable and can handle larger data-sets than t-SNE.



Step	t-SNE	UMAP
Similarity calculation	Calculates high-dimensional similarities (Euclidean, cosine, etc.) and Gaussian distribution.	High-dimensional similarities using Riemannian distances and Gaussian distribution.
Probability Distribution	Constructs the pairwise probability distribution	Constructs the pairwise probability distributions - a fuzzy simplicial set
Initializing Embedding	Initializing random points in lower dimension.	Initializing random points in lower dimension.
Low Dimension Similarities	Calculates low-dimensional similarities using a t-distribution.	Calculates low-dimensional similarities using a t-distribution.
Optimization	Optimizes by minimizing the KL-Divergence between high/low-dimensional distributions	Optimizes by minimizing cross-entropy loss between high-dimensional fuzzy simplicial set and low-dimensional set.
Refinement	N/A	Fine tunes by adjusting data point positions based on the structure and data connectivity.
Convergence	Iterate optimization until convergence/max iterations reached.	Iterate both optimization and refinement steps until convergence/max iterations reached.

Application and Conclusions

*Example of the general workflow when using clustering on your data.



Data

Clean/process your data, define objective, and decide on a workflow.

Dimensionality Reduction

Does the data require feature reduction to reduce complexity or improve visualizations? Choose your method(s).

Clustering

Choose your favorite clustering algorithm (or just test a bunch) and run it on the reduced data.

Evaluation

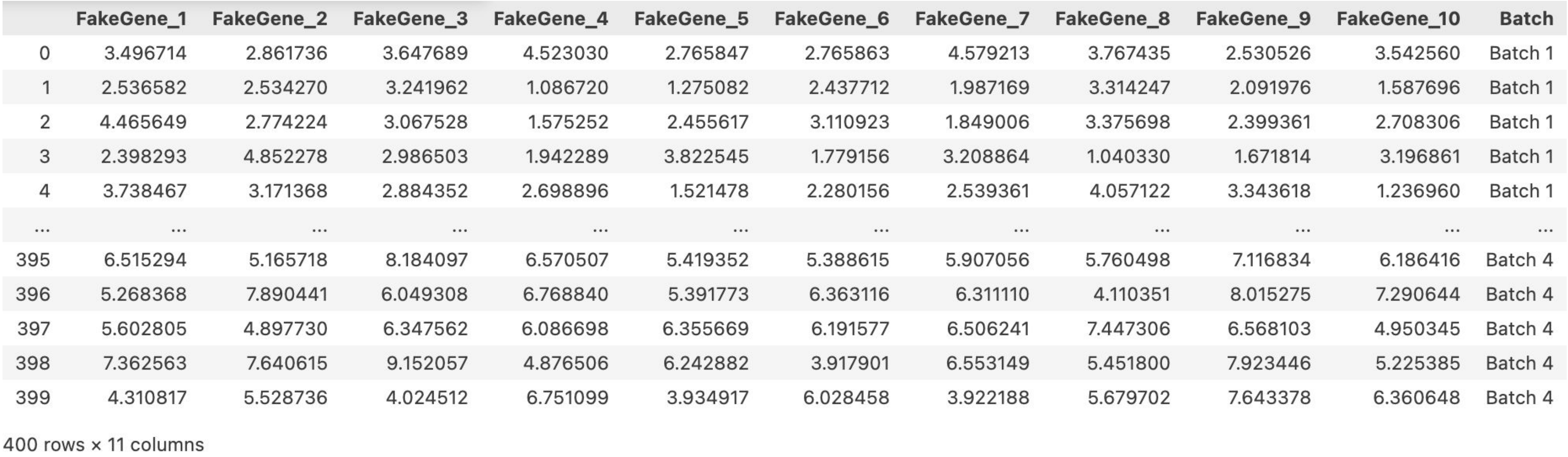
Use evaluation metrics to discern the quality of your clusters. What are the batch effects like? etc.

Insights

What can you get out of these results? Did it address your original objective?

Example Implementation for those interested…

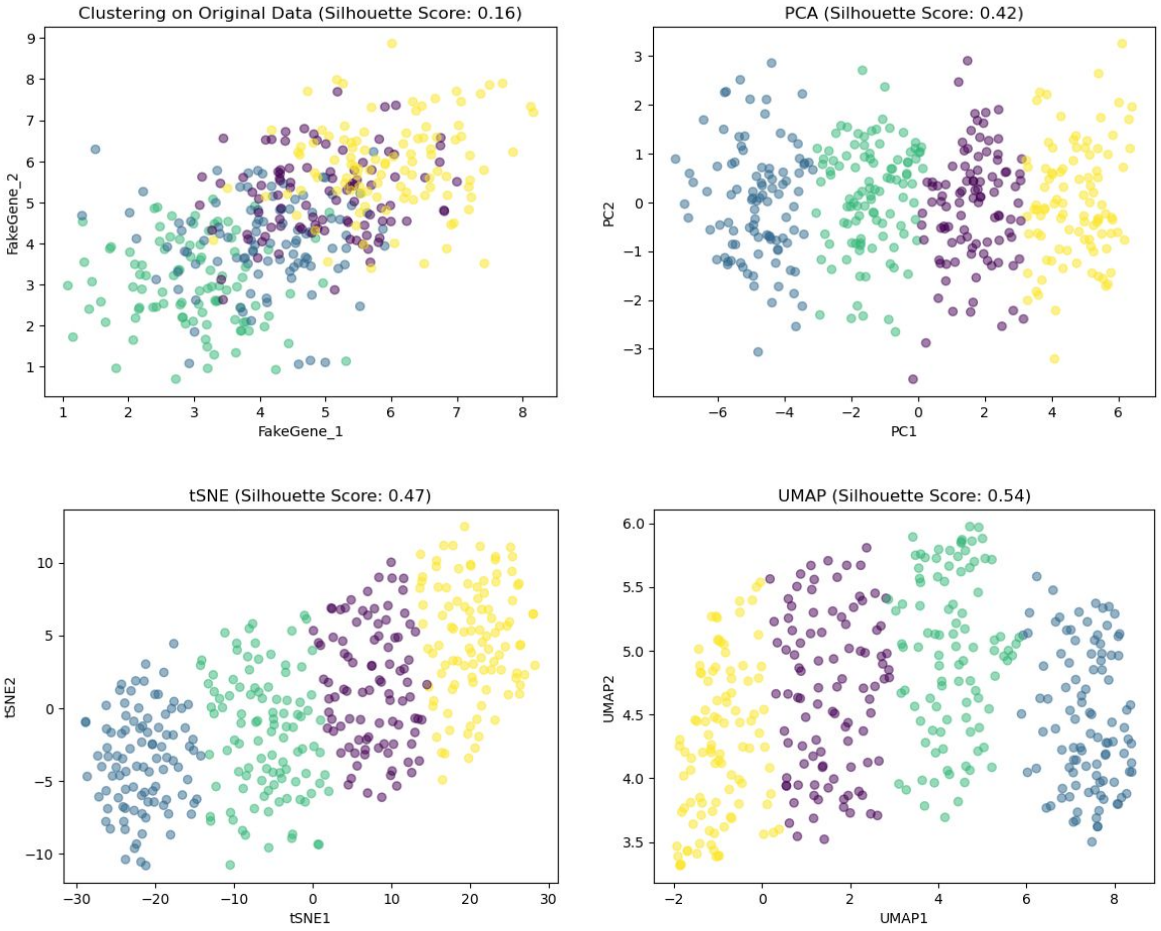
- This is just a quick and simple example of using clustering with PCA, tSNE, and UMAP on fake generated genetic data (following the workflow from the previous slide). I used K-Means as the clustering algorithm and silhouette score as an evaluation metric. Scikit-learn was used, so no complicated calculations were done.
- To view the full script on Colab, you can click the link below, otherwise, just continue on to the next slides!
    - https://colab.research.google.com/drive/1IRYV2ZHE6VNrr97aKP4C-UuKJdwquw8N?usp=sharing



	FakeGene_1	FakeGene_2	FakeGene_3	FakeGene_4	FakeGene_5	FakeGene_6	FakeGene_7	FakeGene_8	FakeGene_9	FakeGene_10	Batch
0	3.496714	2.861736	3.647689	4.523030	2.765847	2.765863	4.579213	3.767435	2.530526	3.542560	Batch 1
1	2.536582	2.534270	3.241962	1.086720	1.275082	2.437712	1.987169	3.314247	2.091976	1.587696	Batch 1
2	4.465649	2.774224	3.067528	1.575252	2.455617	3.110923	1.849006	3.375698	2.399361	2.708306	Batch 1
3	2.398293	4.852278	2.986503	1.942289	3.822545	1.779156	3.208864	1.040330	1.671814	3.196861	Batch 1
4	3.738467	3.171368	2.884352	2.698896	1.521478	2.280156	2.539361	4.057122	3.343618	1.236960	Batch 1
...	...	...	...	...	...	...	...	...	...	...	...
395	6.515294	5.165718	8.184097	6.570507	5.419352	5.388615	5.907056	5.760498	7.116834	6.186416	Batch 4
396	5.268368	7.890441	6.049308	6.768840	5.391773	6.363116	6.311110	4.110351	8.015275	7.290644	Batch 4
397	5.602805	4.897730	6.347562	6.086698	6.355669	6.191577	6.506241	7.447306	6.568103	4.950345	Batch 4
398	7.362563	7.640615	9.152057	4.876506	6.242882	3.917901	6.553149	5.451800	7.923446	5.225385	Batch 4
399	4.310817	5.528736	4.024512	6.751099	3.934917	6.028458	3.922188	5.679702	7.643378	6.360648	Batch 4

400 rows × 11 columns

I used random data to create a fake dataset of gene expression levels for 400 samples. They are split into 4 batches with their own noise patterns. Each batch has 100 samples, and there are 10 genes in total. In this case, the batches might refer to different cell types or test groups or species or etc. This is just for an example, so please note that data does not normally look this evenly distributed.



Clustering on Original Data (Silhouette Score: 0.16)

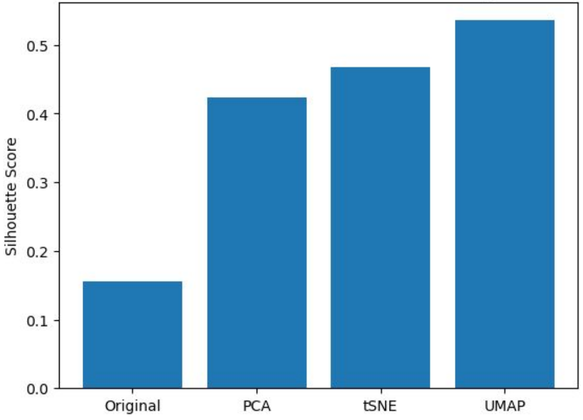
PCA (Silhouette Score: 0.42)

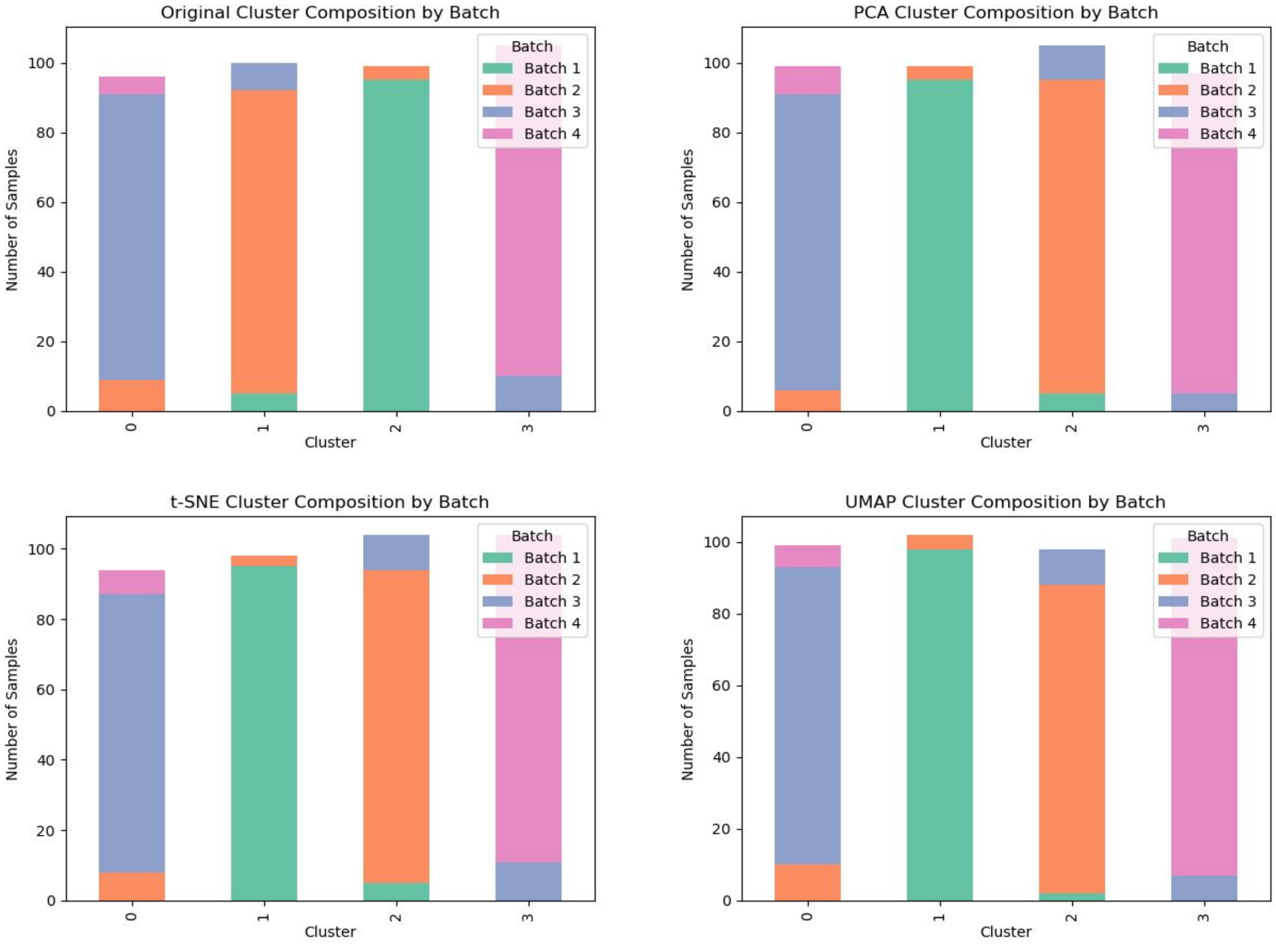
tSNE (Silhouette Score: 0.47)

UMAP (Silhouette Score: 0.54)

```
   Method  Silhouette Score
 Original          0.155652
      PCA          0.424108
     tSNE          0.467694
     UMAP          0.535558
```

Silhouette Scores for KMeans Clustering





Cluster Composition:

Depending on the nature of the batches (are they experiment groups, cell types, etc.), dimensionality reduction can significantly influence data analysis by mitigating batch effects or helping to distinguish between different biological conditions such as cell types. In genetic research, these scores are often visualized as heatmaps.

References:

- https://www.parsebiosciences.com/blog/publication-trends-of-scrna-seq-research/ - scRNAseq trends
- https://www.science.org/doi/10.1126/scitranslmed.abd8995- scRNAseq paper
- https://www.bioinformaticscrashcourse.com/9.1_Clustering.html - Clustering
- https://neptune.ai/blog/clustering-algorithms - Clustering algorithms
- https://www.youtube.com/watch?v=7xHsRkOdVwo&t=339s - Hierarchical clustering
- https://www.youtube.com/watch?v=RDZUdRSDOok - DBSCAN overview
- https://medium.com/@aastha.code/dimensionality-reduction-pca-t-sne-and-umap-41d499da2df2#:~:text=Comparison%20of%20PCA%2C%20t%2DSNE%2C%20and%20UMAP&text=PCA%3A%20Linear%2C%20fast%2C%20good,preserves%20local%20and%20global%20structure. - Dimensionality reduction
- https://aurigait.com/blog/blog-easy-explanation-of-dimensionality-reduction-and-techniques/ - Dimensionality reduction techniques
- https://www.youtube.com/watch?v=HMOI_lkzW08 - PCA main concepts
- https://www.scdiscoveries.com/blog/knowledge/what-is-t-sne-plot/ - More on t-SNE plots
- https://siegel.work/blog/tSNE/ - t-SNE probability distribution