

# Dimensionality reduction

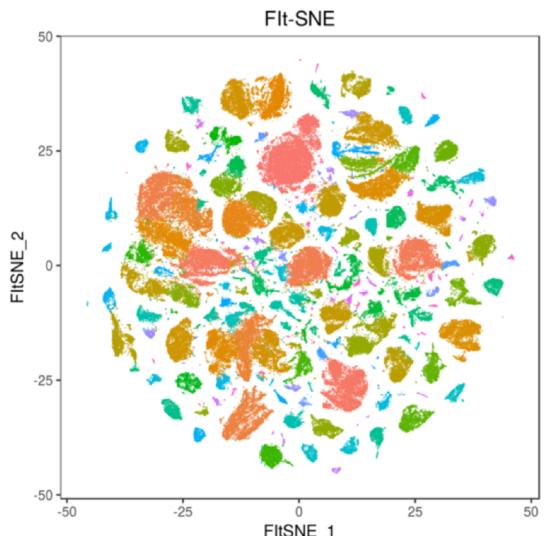
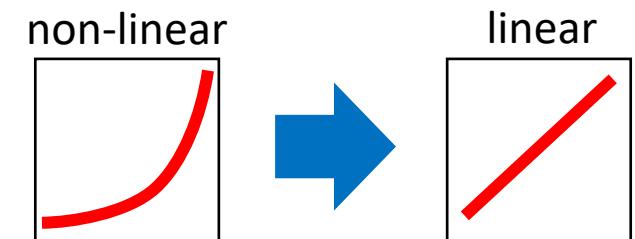
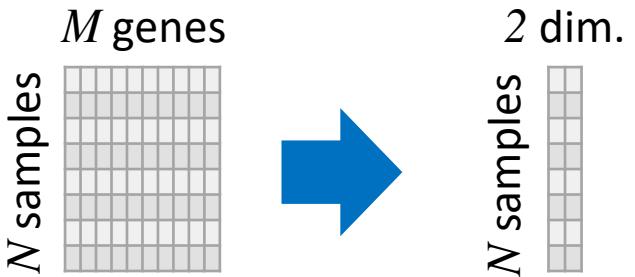
---

Paulo Czarnewski  
*Bioinformatician*  
[paulo.czarnowski@nbis.se](mailto:paulo.czarnowski@nbis.se)

*National Bioinformatics Infrastructure Sweden (NBIS) – nbis.se  
Science for Life Laboratory (SciLifeLab)*

# Why dimensionality reduction?

- Simplify complexity, so it becomes easier to work with.  
Reduce number of features (genes)  
In some: Transform non-linear relationships to linear
- “Remove” redundancies in the data
- Identify the most relevant information (find and filter noise)
- Reduce computational time for downstream procedures
- Facilitate clustering, since some algorithms struggle with too many dimensions
- Data visualization



... and more ...

# Some dimensionality reduction algorithms

They can be divided into 2 major groups:

- Matrix Factorization (including autoencoders)
- Graph-based

→ PCA	linear	Matrix Factorization		
MDS	non-linear	Matrix Factorization		
Sparce NNMF	non-linear	Matrix Factorization	2010	<a href="https://pdfs.semanticscholar.org/664d/40258f12ad28ed0b7d4c272935ad72a150db.pdf">https://pdfs.semanticscholar.org/664d/40258f12ad28ed0b7d4c272935ad72a150db.pdf</a>
cPCA	non-linear	Matrix Factorization	2018	<a href="https://doi.org/10.1038/s41467-018-04608-8">https://doi.org/10.1038/s41467-018-04608-8</a>
ZIFA	non-linear	Matrix Factorization	2015	<a href="https://doi.org/10.1186/s13059-015-0805-z">https://doi.org/10.1186/s13059-015-0805-z</a>
ZINB-WaVE	non-linear	Matrix Factorization	2018	<a href="https://doi.org/10.1038/s41467-017-02554-5">https://doi.org/10.1038/s41467-017-02554-5</a>

→ Isomap	non-linear	graph-based	2000	<a href="https://doi.org/10.1126/science.290.5500.2319">10.1126/science.290.5500.2319</a>
Diffusion maps	non-linear	graph-based	2005	<a href="https://doi.org/10.1073/pnas.0500334102">https://doi.org/10.1073/pnas.0500334102</a>
→ t-SNE	non-linear	graph-based	2008	<a href="https://lvdmaaten.github.io/publications/papers/JMLR_2008.pdf">https://lvdmaaten.github.io/publications/papers/JMLR_2008.pdf</a>
- BH t-SNE	non-linear	graph-based	2014	<a href="https://lvdmaaten.github.io/publications/papers/JMLR_2014.pdf">https://lvdmaaten.github.io/publications/papers/JMLR_2014.pdf</a>
- Flt-SNE	non-linear	graph-based	2017	<a href="https://arxiv.org/abs/1712.09005">arXiv:1712.09005</a>
LargeVis	non-linear	graph-based	2018	<a href="https://arxiv.org/abs/1602.00370">arXiv:1602.00370</a>
→ UMAP	non-linear	graph-based	2018	<a href="https://arxiv.org/abs/1802.03426">arXiv:1802.03426</a>
PHATE	non-linear	graph-based		<a href="https://www.biorxiv.org/content/biorxiv/early/2018/06/28/120378.full.pdf">https://www.biorxiv.org/content/biorxiv/early/2018/06/28/120378.full.pdf</a>

→ scvis	non-linear	Autoencoder (MF)	2018	<a href="https://doi.org/10.1038/s41467-018-04368-5">https://doi.org/10.1038/s41467-018-04368-5</a>
VASC	non-linear	Autoencoder (MF)	2018	<a href="https://doi.org/10.1016/j.gpb.2018.08.003">https://doi.org/10.1016/j.gpb.2018.08.003</a>

... and many more

# PCA

---

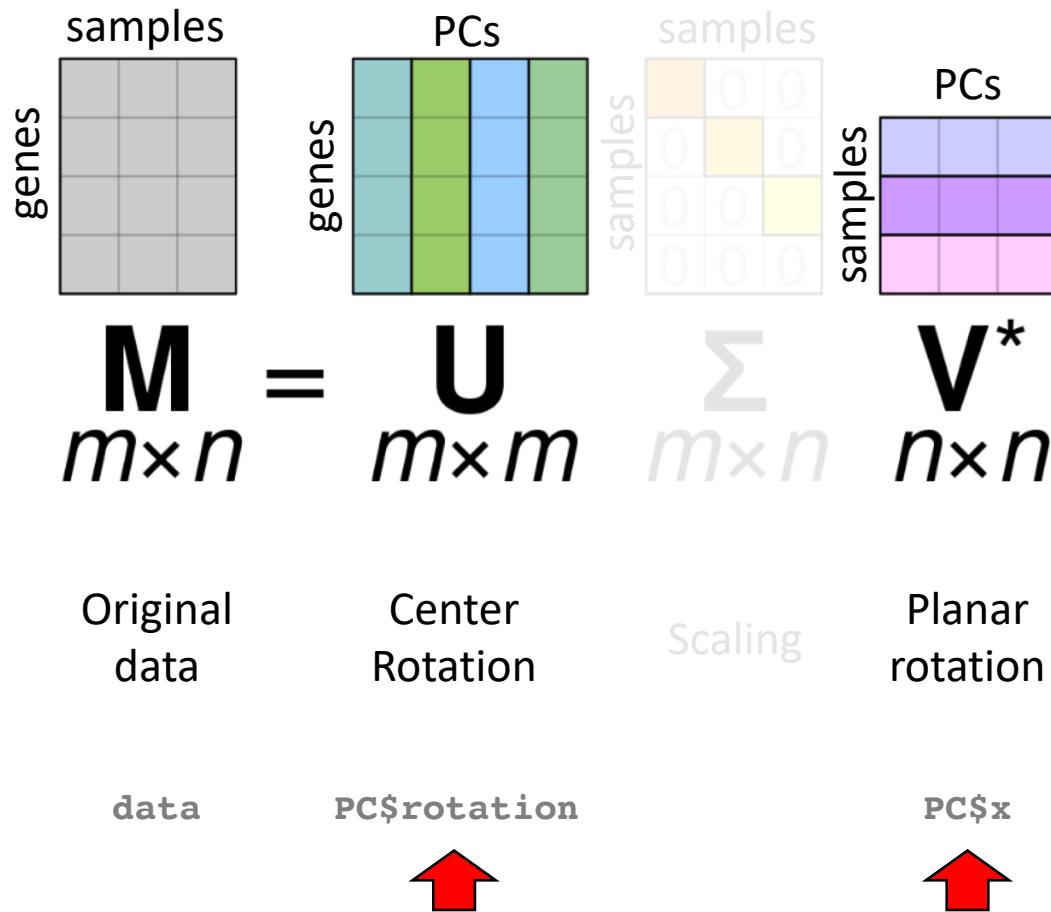
Principal Component Analysis

# How PCA works

It is a LINEAR algebraic method of dimensionality reduction.

It is a case inside Singular Value Decomposition (SVD) method (data compression)

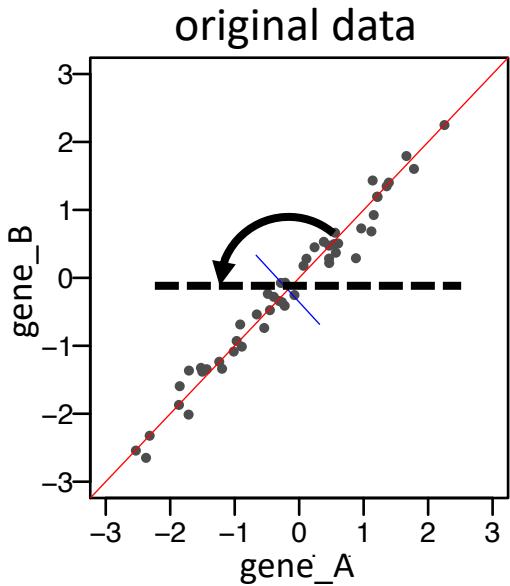
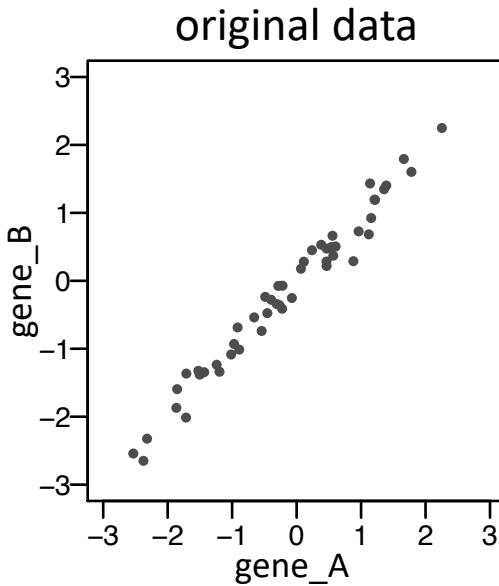
Any matrix can be decomposed as a multiplication of other matrices (Matrix Factorization).



```
PC <- prcomp( data )
PC$|
```

	sdev
	rotation
	center
	scale
	x

# How PCA works



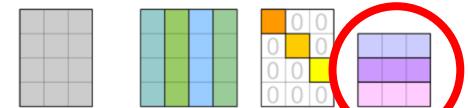
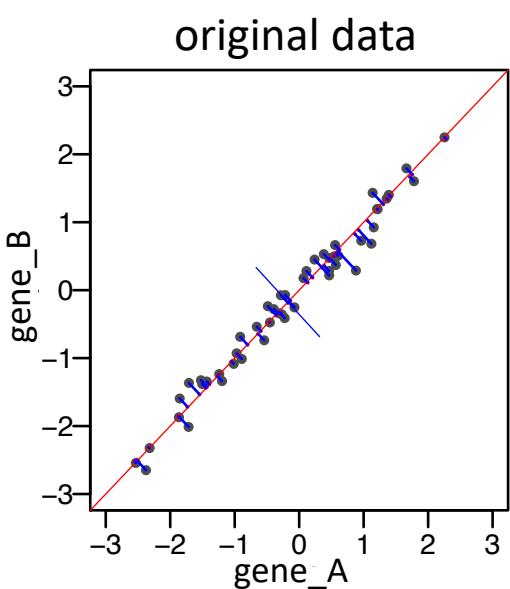
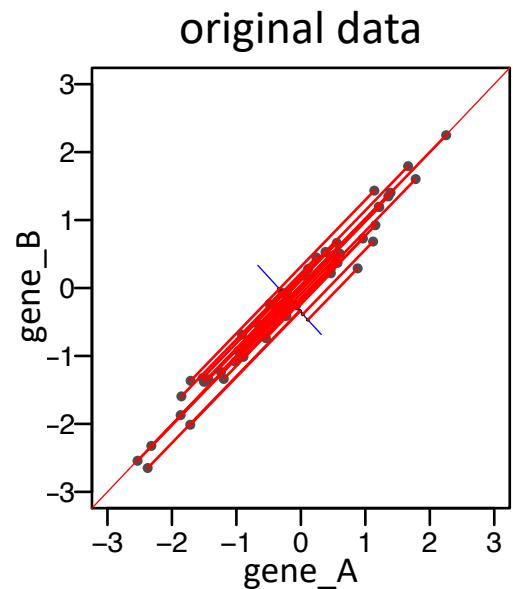
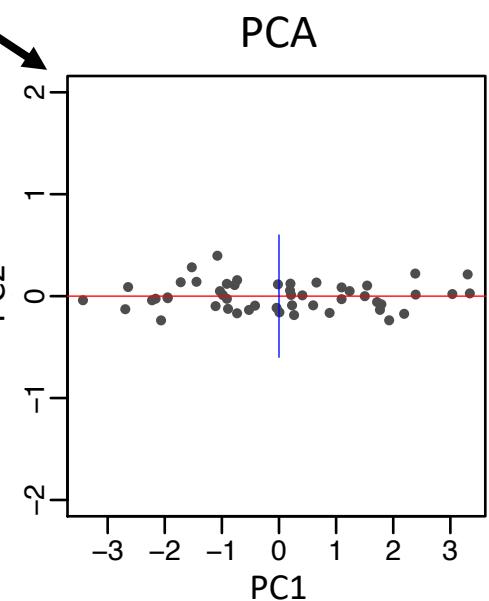
$$M_{m \times n} = U_{m \times m} \Sigma_{m \times n} V^*_{n \times n}$$


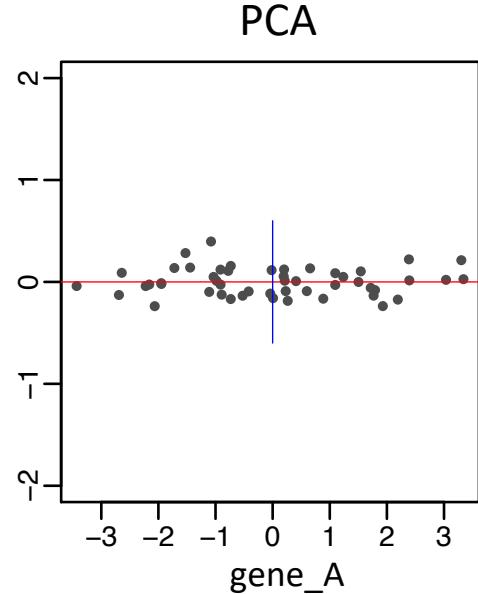
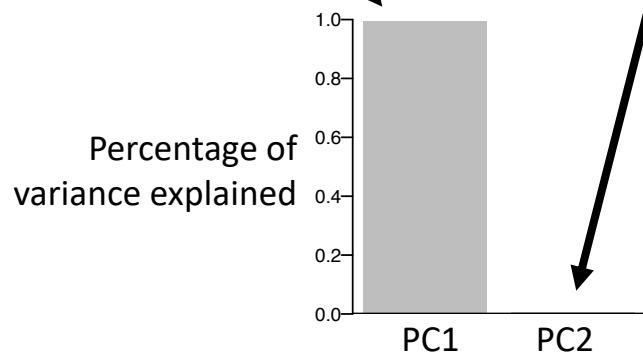
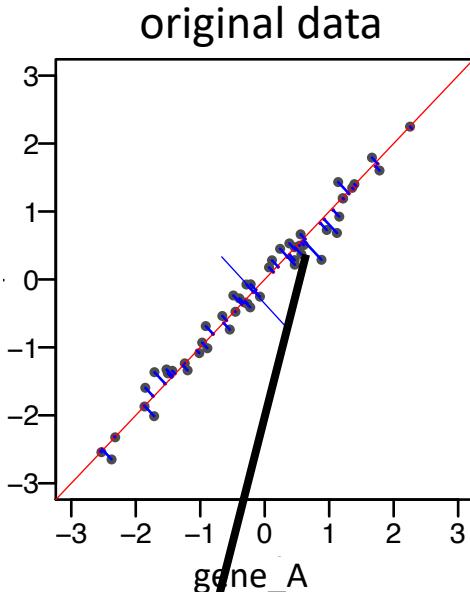
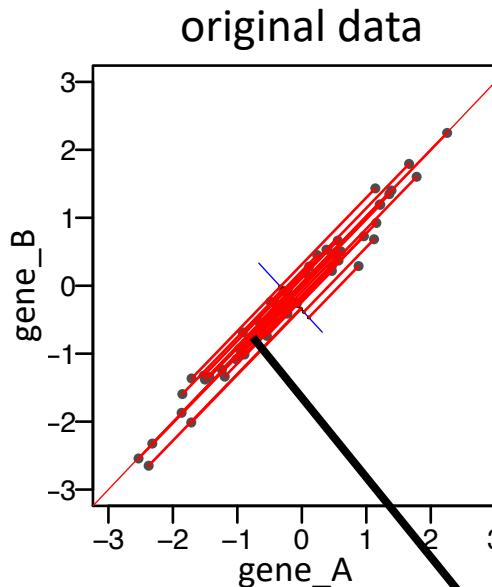
Diagram illustrating the singular value decomposition (SVD) components and their dimensions:

- $M$ :  $m \times n$
- $U$ :  $m \times m$
- $\Sigma$ :  $m \times n$
- $V^*$ :  $n \times n$

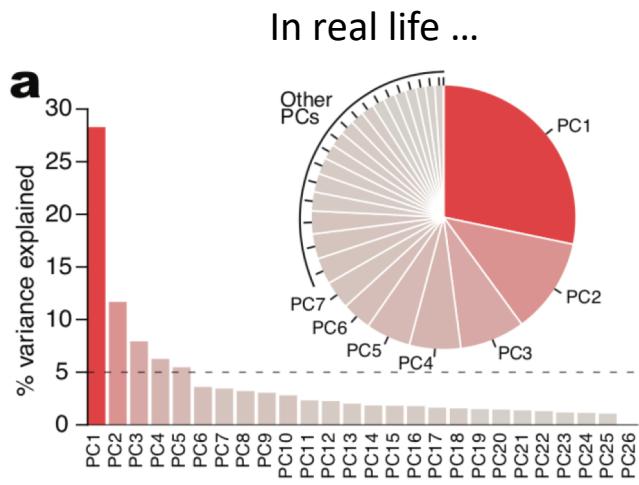
rotation



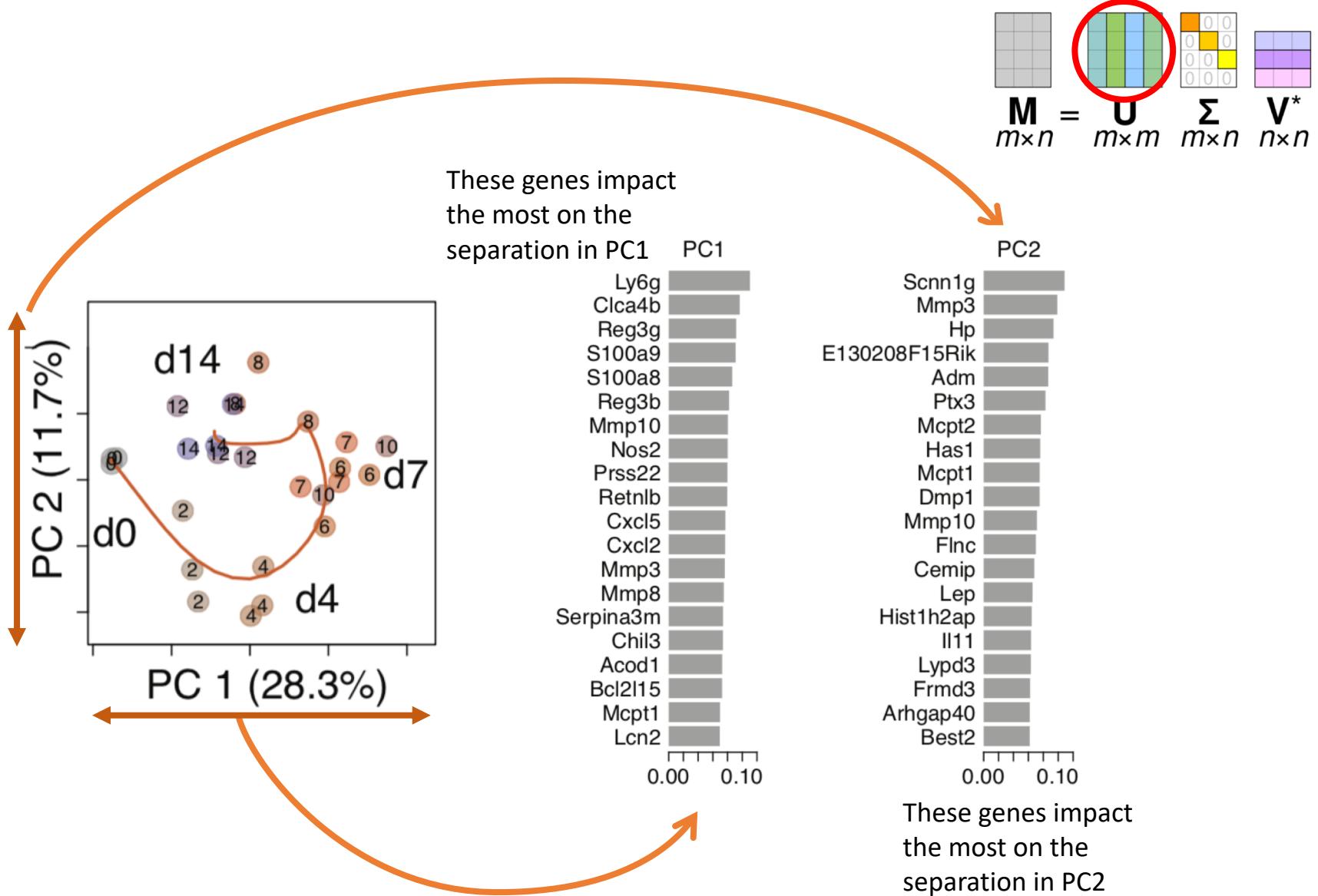
# PCs retain variance from the data



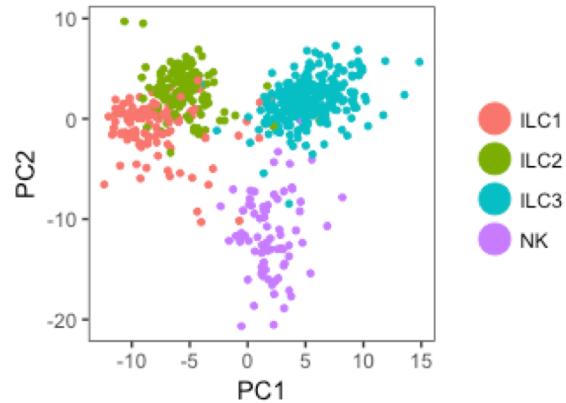
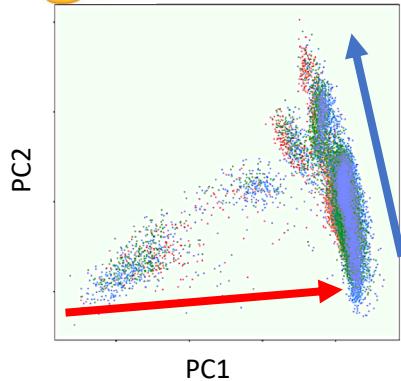
- PC1 explains >98% of the variance in this dataset
- PC1 represents 2 genes (“removing” redundancy)
- PC2 is nearly insignificant (can be disregarded)



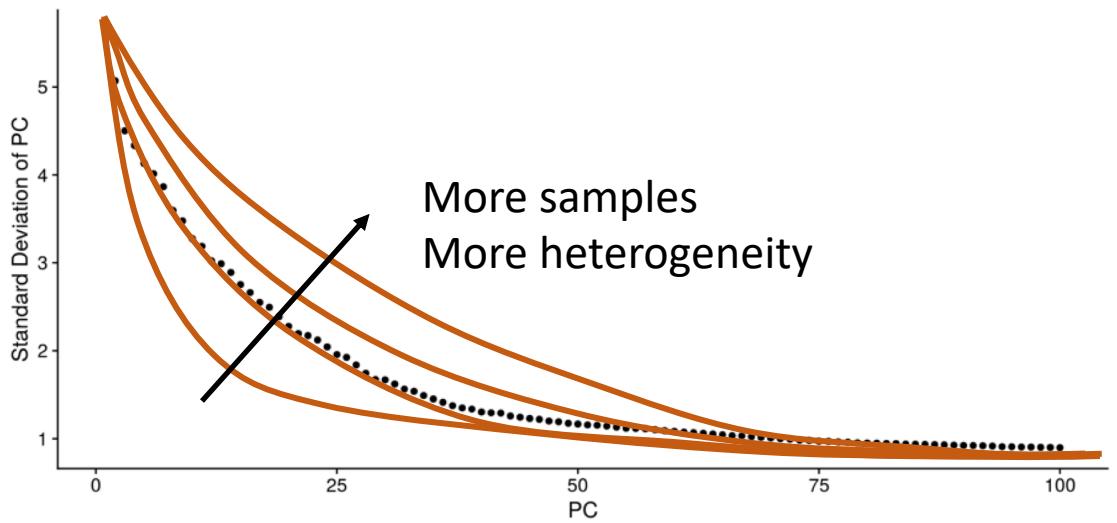
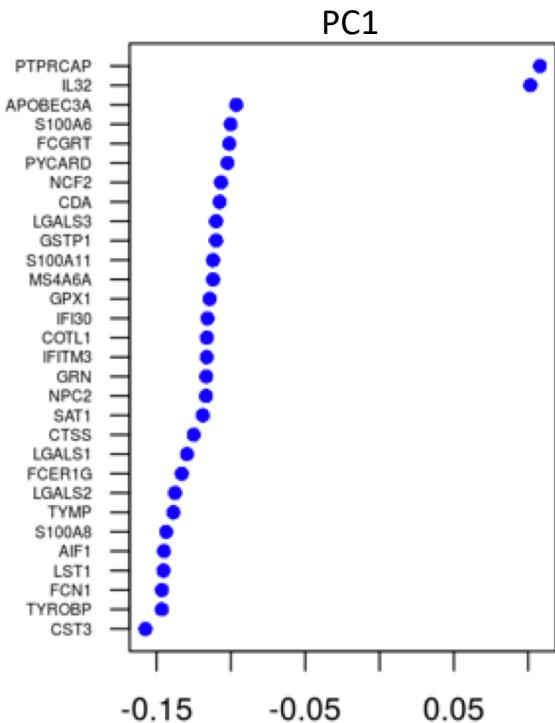
# Genes contribute differently to each PC



# PCA in single cell data



- PC1 and PC2 are commonly correlated to **sequencing depth** and cell **heterogeneity/complexity** (but not always ...)



# PCA: summary

## To keep in mind:

- It is a LINEAR method of dimensionality reduction
- It is an interpretable dimensionality reduction
- Data is usually SCALED prior to PCA (Z-score | see ScaleData in the Seurat)
- The TOP principal components contain higher variance from the data
- Can be used as FILTERING, by selecting only the top significant PCs
  - PCs that explain at least 1% of variance
  - Jackstraw of significant p-values
  - The first 5-10 PCs

## Problems:

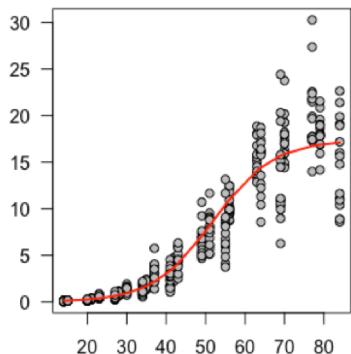
- It performs poorly to separate cells in 0-inflated data types (because of its non-linearity nature)
- Cell sizes and sequencing depth are usually captured in the top principal components

# A very brief intro to graphs

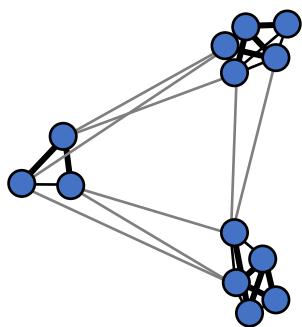
---

The fundamental piece for efficient non-linear dimensionality reduction algorithms

# Graphs



This is a PLOT



This is GRAPH  
(a.k.a. network)

- Each dot is a cell (or a gene)
- Each line represents a connection between 2 cells
- Each connection can be weighted as a proximity between cells
  - Correlation (high and positive)
  - Euclidean distance (low)
  - etc.

Graph-based dimensionality reduction algorithms can be divided into 2 main steps:

1. Construct a weighted graph based on the top  $k$  connections  
(a.k.a.  $k$ -nearest neighbors, KNN)
2. The low dimensional layout of the graph is computed and optimized

# t-SNE

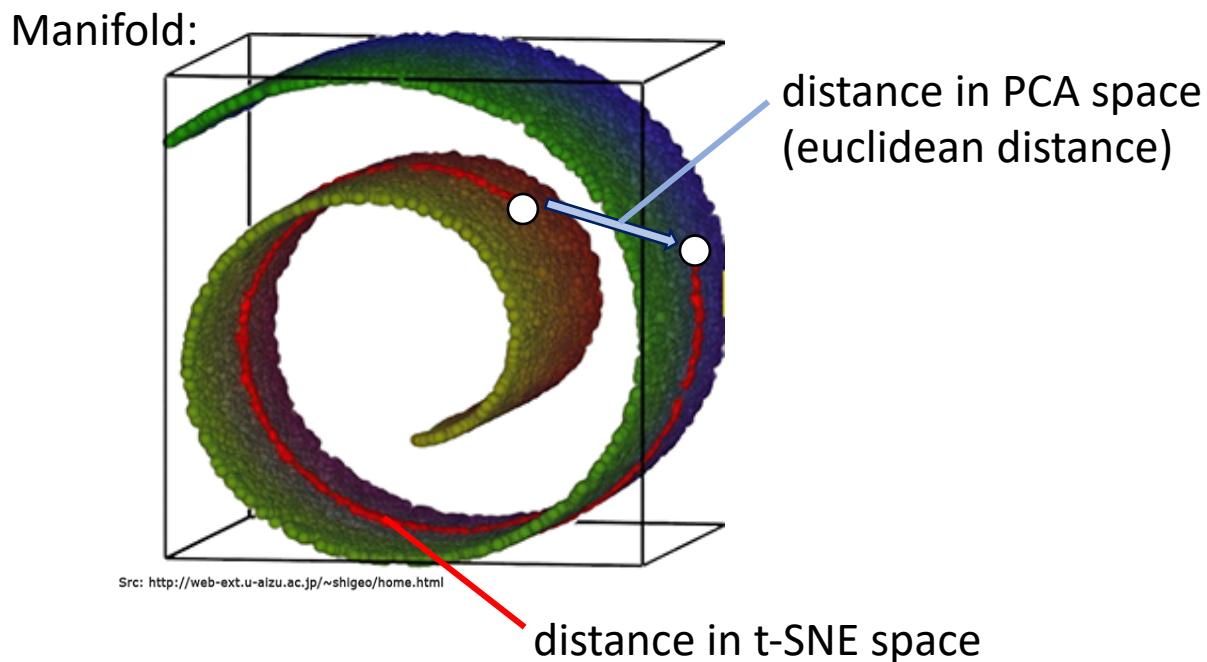
---

t-distributed Stochastic Neighborhood  
Embedding

# How t-SNE works

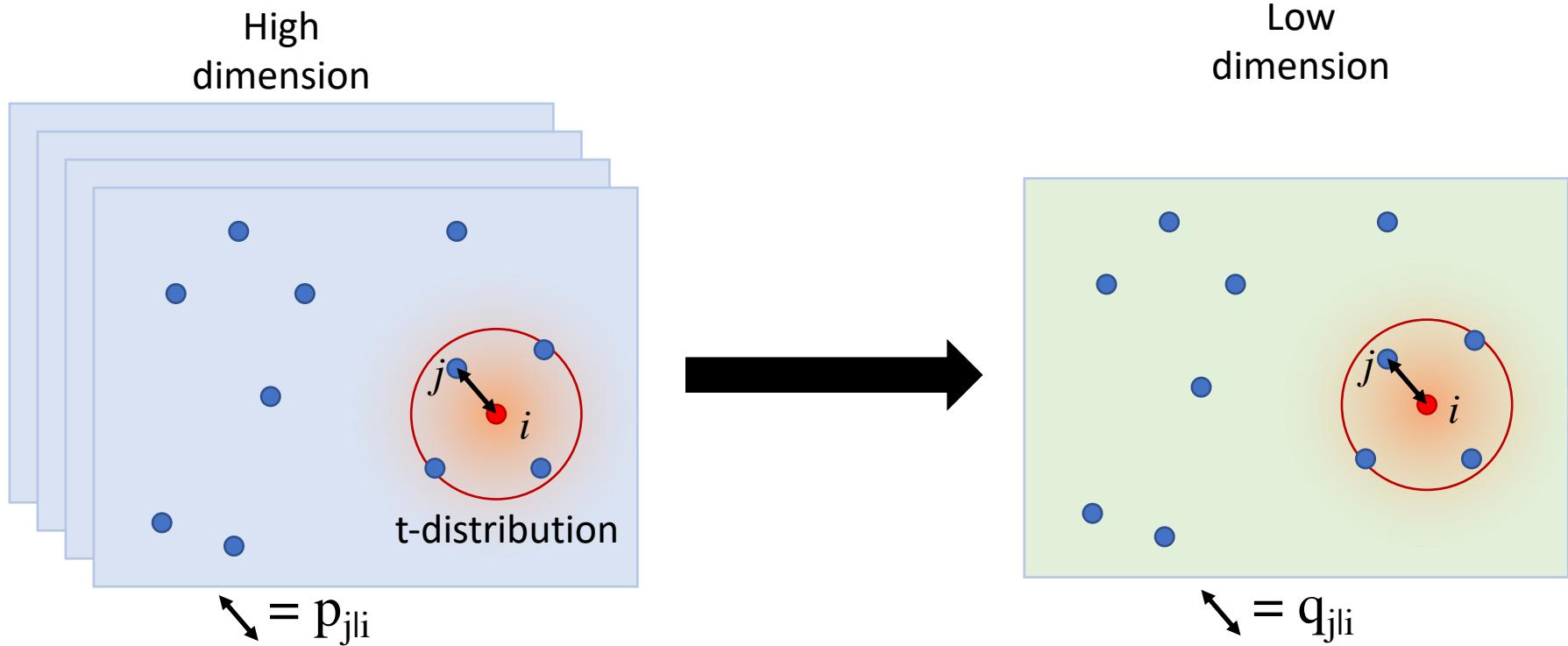
It is a graph-based NON-LINEAR dimensionality reduction

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.  
[https://lvdmaaten.github.io/publications/papers/JMLR\\_2008.pdf](https://lvdmaaten.github.io/publications/papers/JMLR_2008.pdf)

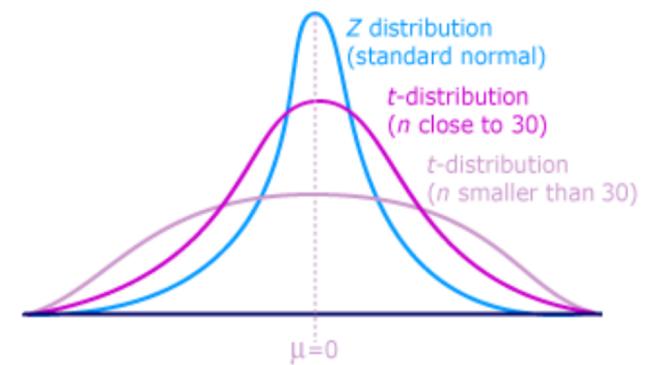


In other words, t-SNE calculates the distances based on the distance to the neighbor cell

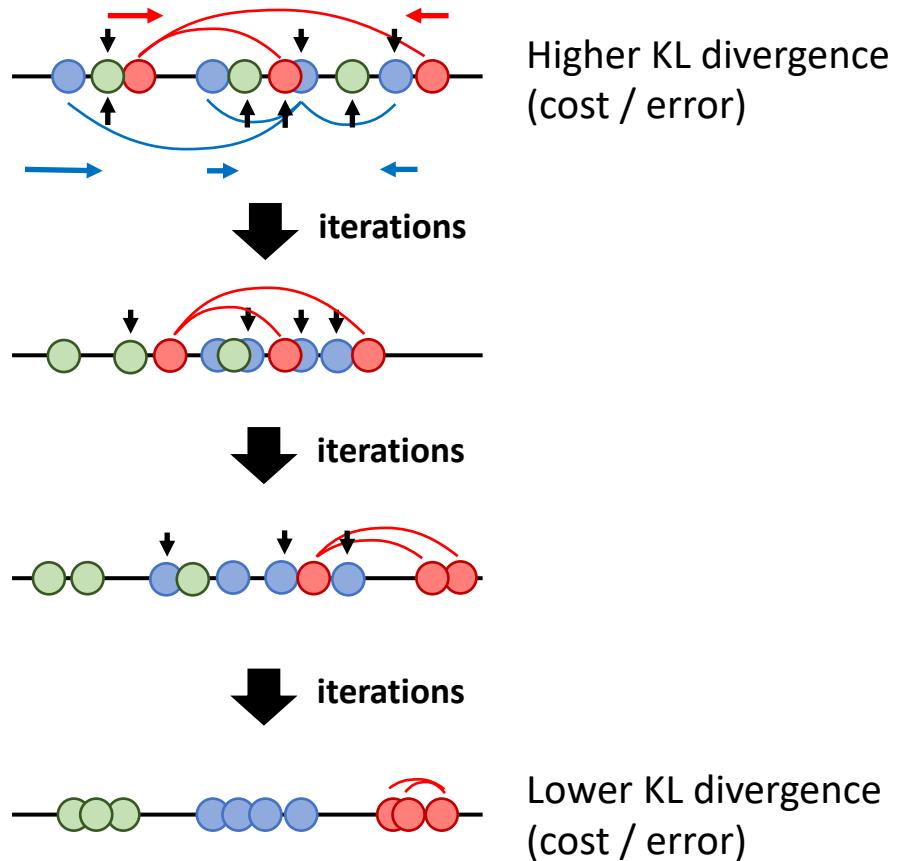
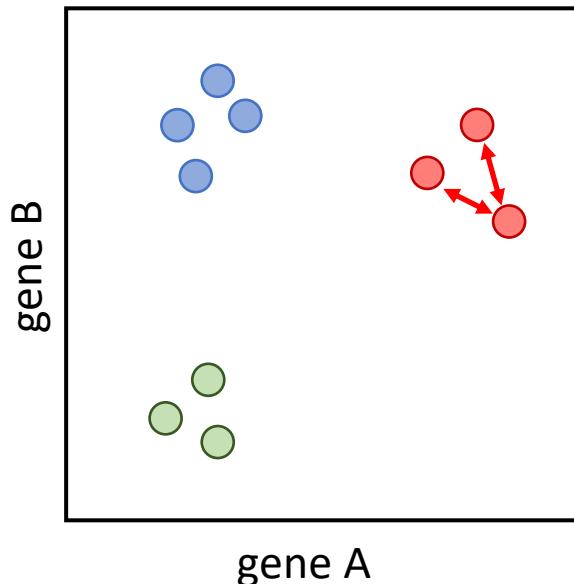
# How t-SNE works



$p_{j|i}$  and  $q_{j|i}$  measure the conditional probability that a point  $i$  would pick point  $j$  as its nearest neighbor, in high ( $p$ ) and low ( $q$ ) dimensional space respectively.



# How t-SNE works



The same concept applies to embedding into 2 dimensions

# t-SNE hyper-parameters

- Barnes-Hut's tSNE implementation -  $O(n \log n)$   
Rtsne & Seurat & viSNE (MATLAB)

Laurens van der Maaten (2014). Accelerating t-SNE using tree-based algorithms. *Journal of Machine Learning Research*, 15(1):3221–3245.

[https://lvdmaaten.github.io/publications/papers/JMLR\\_2014.pdf](https://lvdmaaten.github.io/publications/papers/JMLR_2014.pdf)

The definition of the t-SNE and the chances of converging correctly depends on the hyper-parameters (“tuning” parameters).

t-SNE has 10 hyper-parameters that can be optimized for your specific data.

The most common hyper-parameters are:

- Perplexity
- Number of iterations
- Learning rate
- Theta (for BH t-SNE)

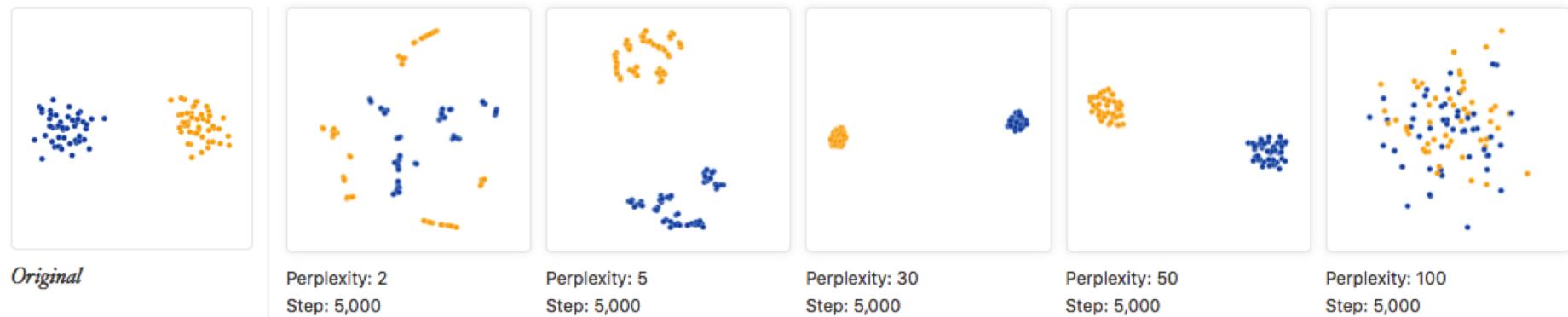
Check this link: <https://distill.pub/2016/misread-tsne/>

# t-SNE hyper-parameters

## Number of iterations



## Perplexity

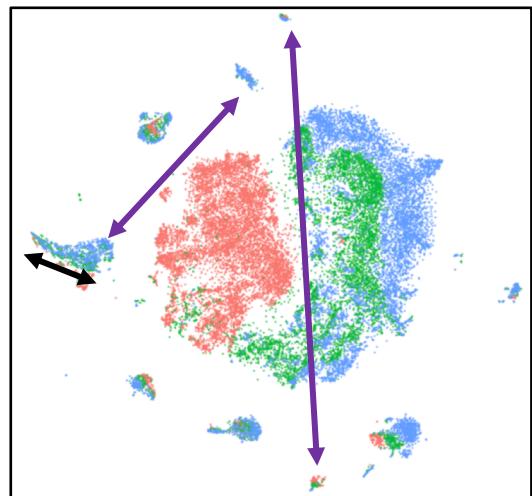


# Important notes about t-SNE

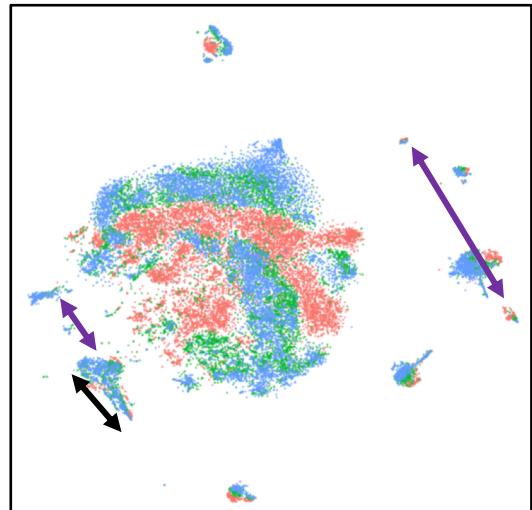
- Unlike PCA, it is a stochastic algorithm, so it will never produce the same output (unless you use a `seed( )` to lock the random estimators).
- The cost function never reaches the minima, and it is not an indicator how good the graph looks.
- The cost function in t-SNE minimizes the distance between similar points (and ignore the distant ones – local embedding)  
 The distances within a group are slightly meaningful, but not between groups!
- To add more samples, you need to re-run the algorithm from start.

Same data with 2 different runs and nearly identical KL divergence

Converged successfully



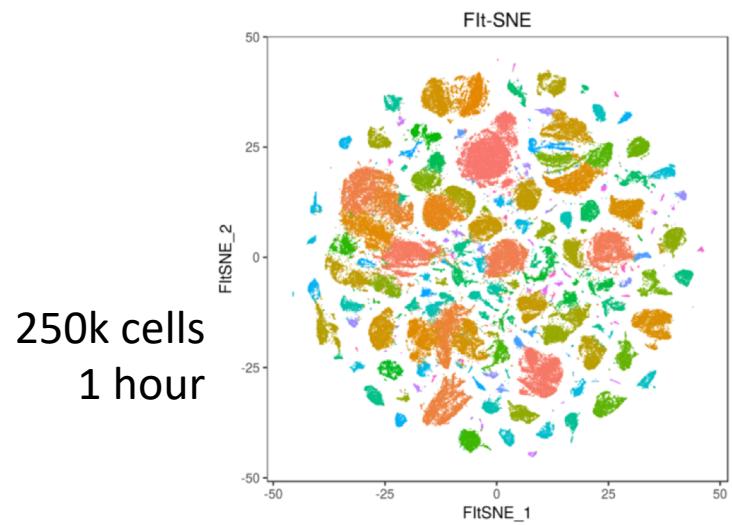
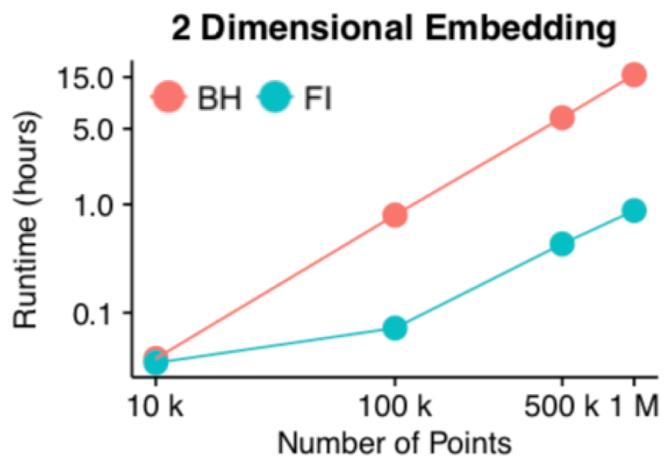
Failed to converge



# Efficient t-SNE implementation

- Fast Fourier Transform-accelerated Interpolation-based t-SNE -  $O(n)$

George C. Linderman, Manas Rachh, Jeremy G. Hoskins, Stefan Steinerberger, Yuval Kluger. **Efficient Algorithms for t-distributed Stochastic Neighborhood Embedding.** 2017  
<https://arxiv.org/pdf/1712.09005.pdf>



# t-SNE: summary

## To keep in mind:

- It is a NON-LINEAR method of dimensionality reduction
- It is the current GOLD-STANDARD method in single cell data (including scRNA-seq)
- Can be run from the top PCs (e.g.: PC1 to PC10)

## Problems:

- It does not learn an explicit function to map new points
- Its cost function is not convex – This means that the optimal t-SNE cannot be computed
- Too many hyper-parameters to be defined empirically (dataset-specific)
- It does not preserve a global data structure (only local)

# UMAP

---

Uniform Manifold Approximation and  
projection

# UMAP

It is a graph-based NON-LINEAR dimensionality reduction

Leland McInnes, John Healy, James Melville. **UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction.** *arXiv:1802.03426* Feb 2018

<https://arxiv.org/abs/1802.03426>

Dimensionality reduction for visualizing single-cell data using UMAP

Etienne Becht<sup>1</sup>, Leland McInnes<sup>2</sup> , John Healy<sup>2</sup>, Charles-Antoine Dutertre<sup>1</sup>, Immanuel W H Kwok<sup>1</sup>, Lai Guan Ng<sup>1</sup>, Florent Ginhoux<sup>1</sup>  & Evan W Newell<sup>1,3</sup> 

It is very efficient -  $O(n)$

It takes both the LOCAL and GLOBAL embeddings into consideration

It stores information about mapping distances for new points

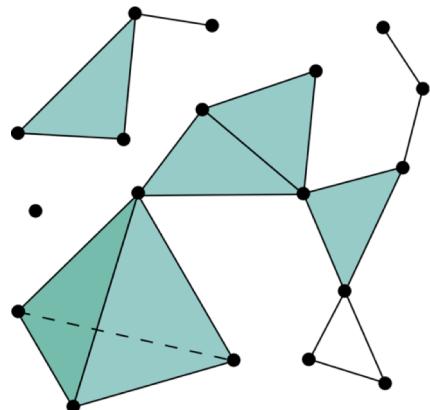
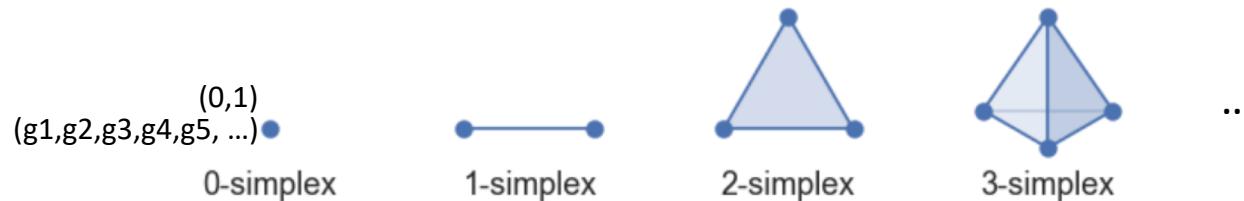
scalable, incremental

# How UMAP works

It is based on topological structures in multidimensional space (simplices)

Points are connected with a line (edge) if the distance between them is below a threshold:

- Any distance metric can be used (euclidean)

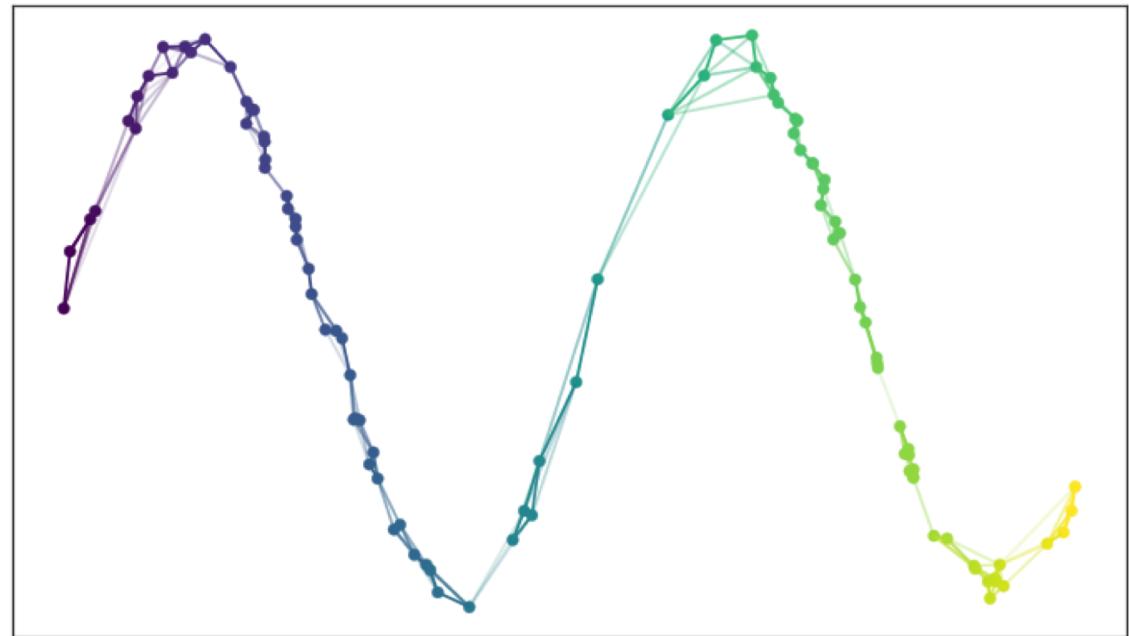
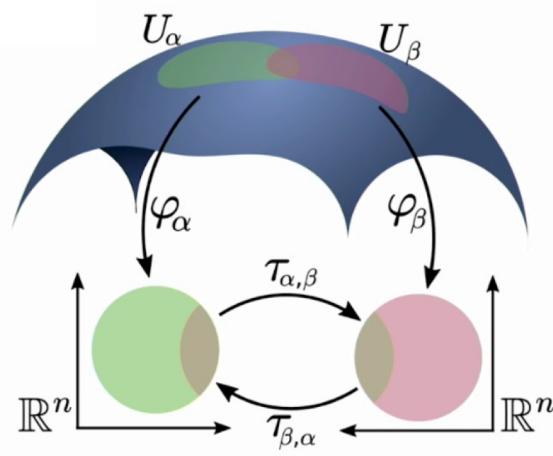


This way, by constructing the simplicial complexes beforehand allows UMAP to calculate the relative point distances in the lower dimension

(instead of randomly as in tSNE)

# How UMAP works

The distance in the manifold are the same, but not in the REAL space.

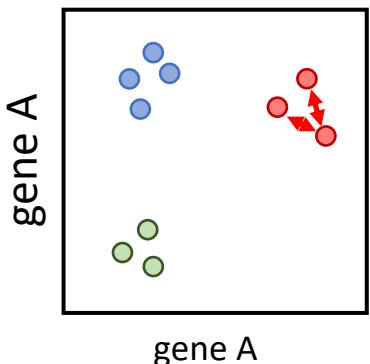


The distance is now “variable” in the REAL space for each point (t-SNE was fixed)

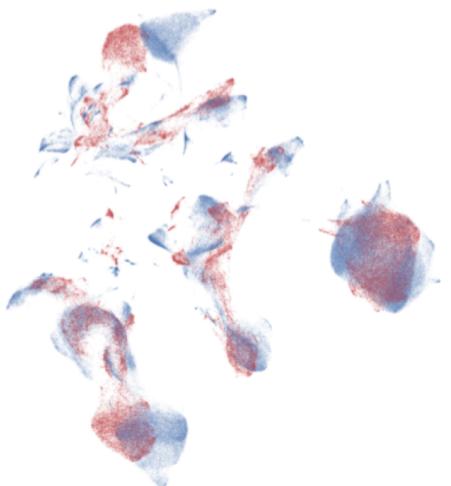
McInnes et al 2018

[https://umap-learn.readthedocs.io/en/latest/how\\_umap\\_works.html](https://umap-learn.readthedocs.io/en/latest/how_umap_works.html)

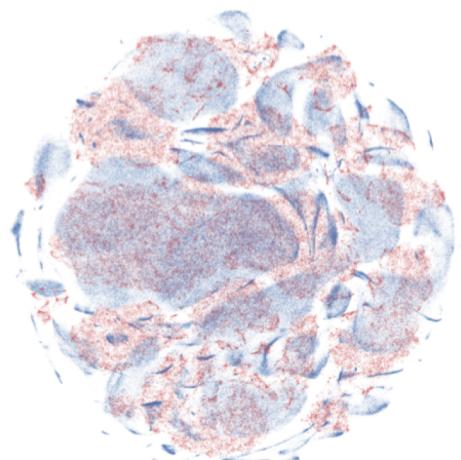
# UMAP



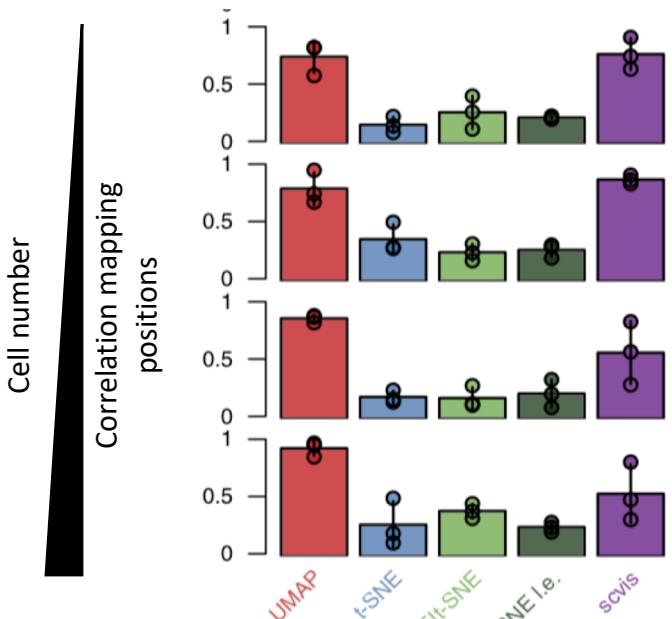
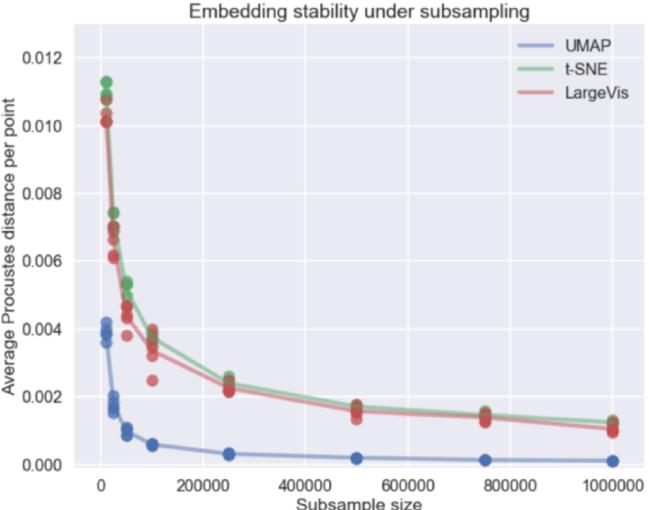
Since UMAP learns the global data structure and is less dependent on random initiators (like t-SNE), it can recreate low dimensional embedding regardless of the dataset size.



(a) UMAP



(b) t-SNE

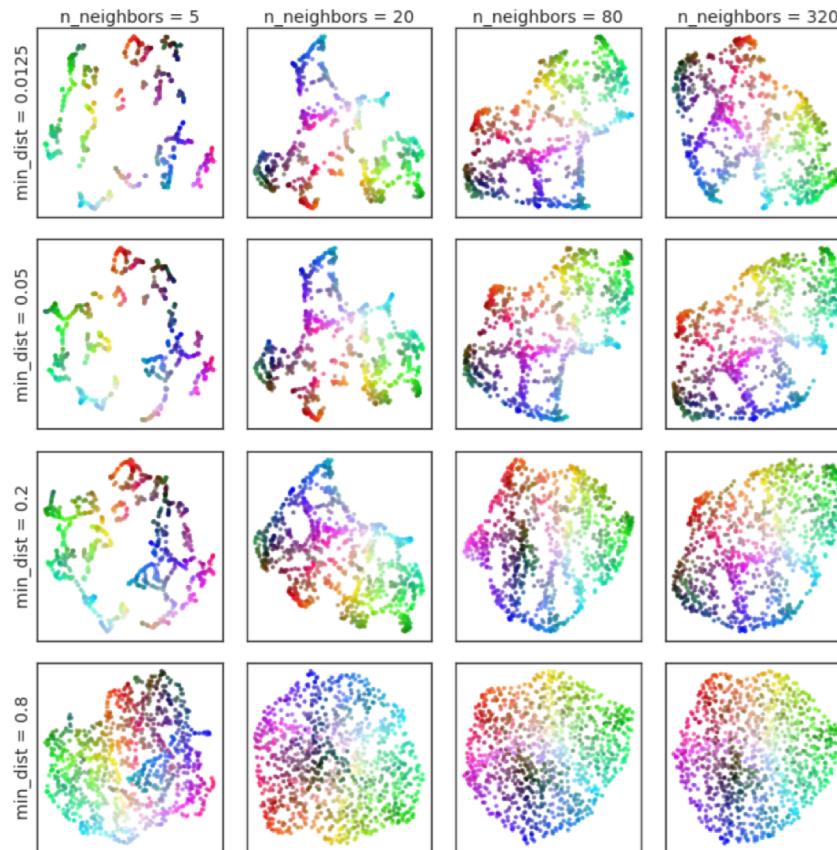


# UMAP hyper-parameters

UMAP assumes that there is a manifold in the dataset, it could also tend to cluster noise.

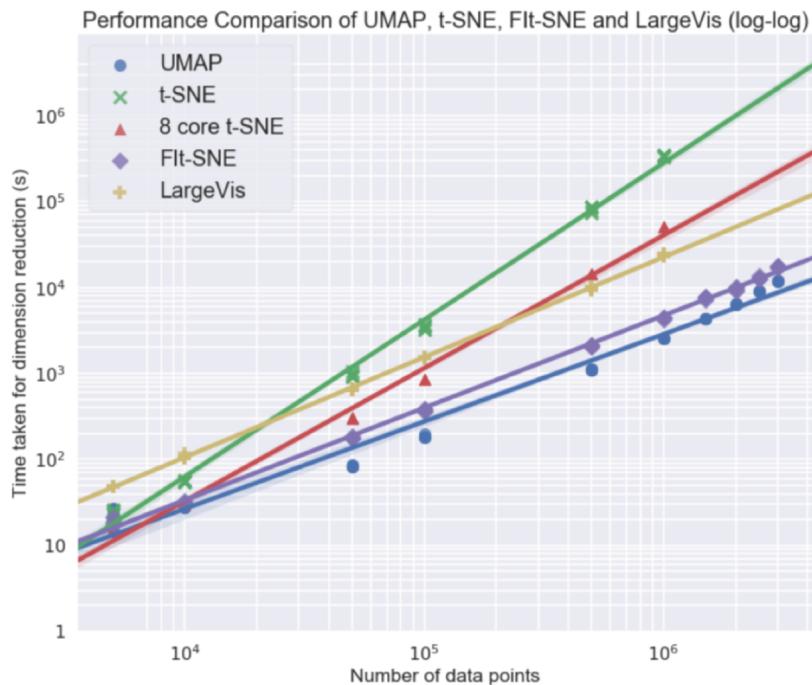
As for t-SNE, checking the parameters is also important.

Embedding of random noise

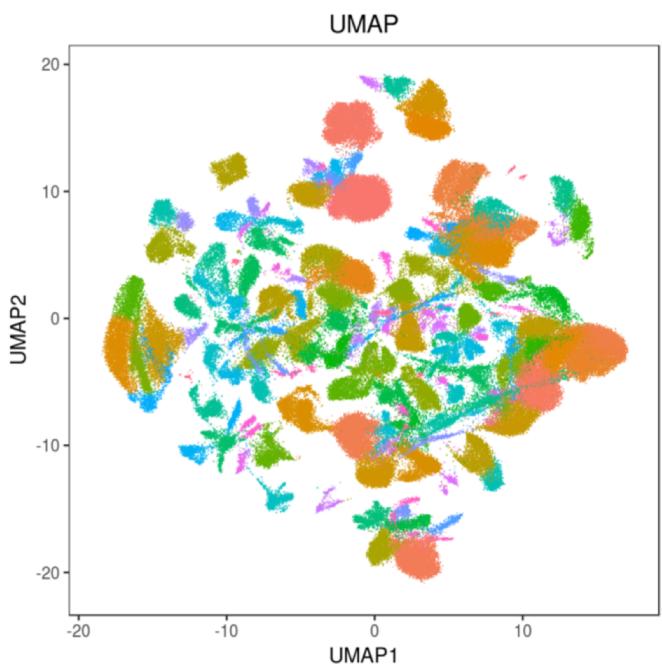


# UMAP hyper-parameters

UMAP's mathematical tricks allows much faster computations compared to current state-of-the-art methods.



250k cells  
7 min



# UMAP: summary

## To keep in mind:

- It is a NON-LINEAR graph-based method of dimensionality reduction
- Very efficient -  $O(n)$
- Can be run from the top PCs (e.g.: PC1 to PC10)
- Can use any distance metrics!
- Can integrate between different data types (text, numbers, classes)
- It is no longer completely stochastic as t-SNE
- Defines both LOCAL and GLOBAL distances
- Can be applied to new data points

# SCVIS and VASC

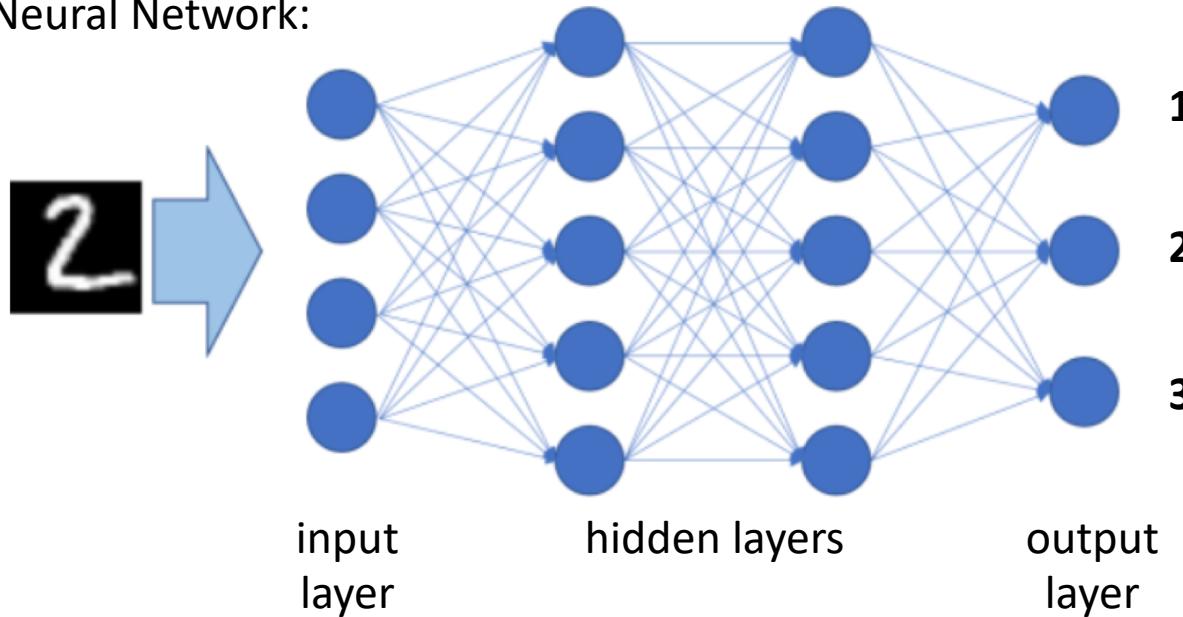
---

Autoencoder-based dim. reduction

# How SCVIS and VASC work

It is a autoencoder-based NON-LINEAR dimensionality reduction

Neural Network:

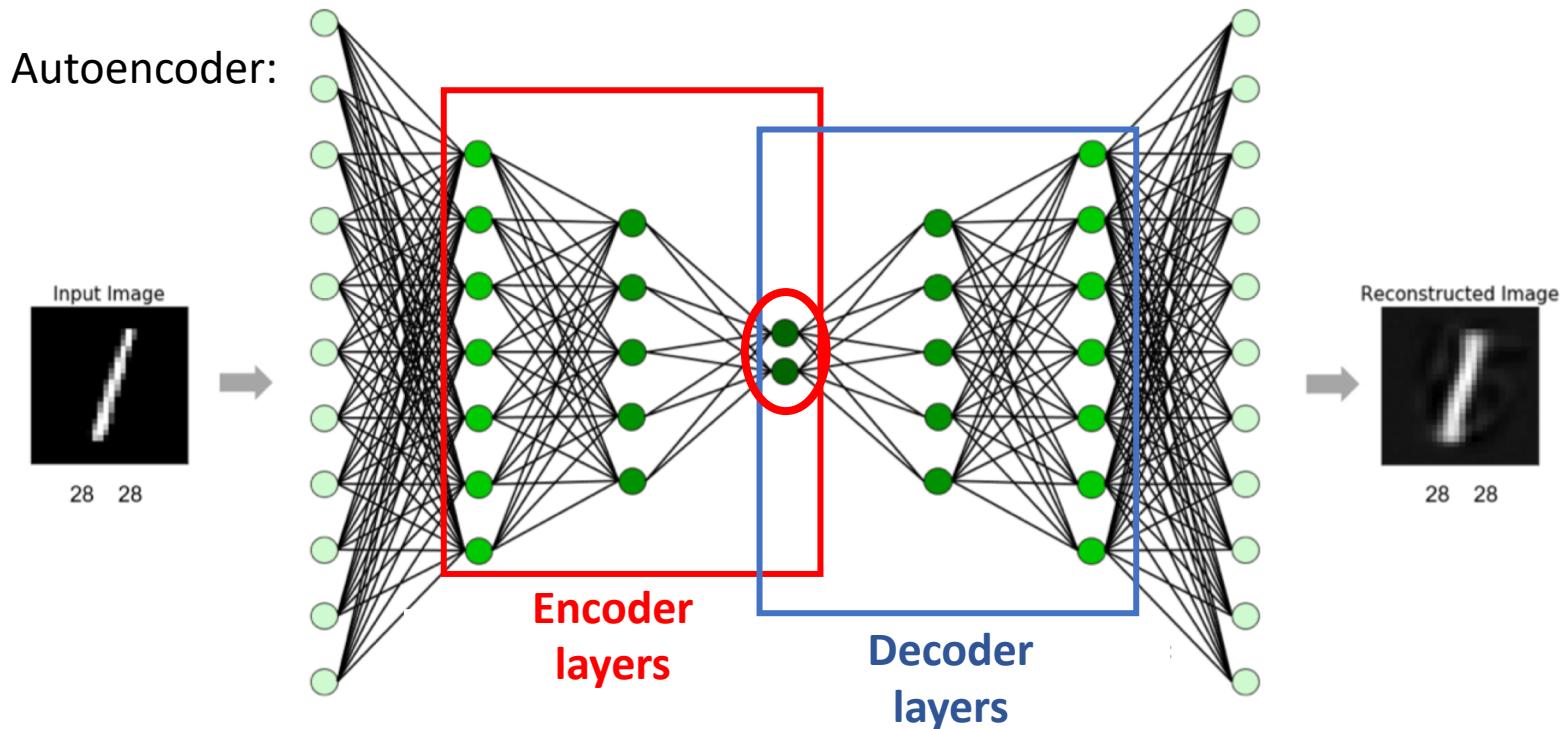


Each hidden layers in the NN captures a specific pattern of the data (as a numeric matrix)

The NN is then trained (adjusted) so that the input matches the desired known output.

# How SCVIS and VASC work

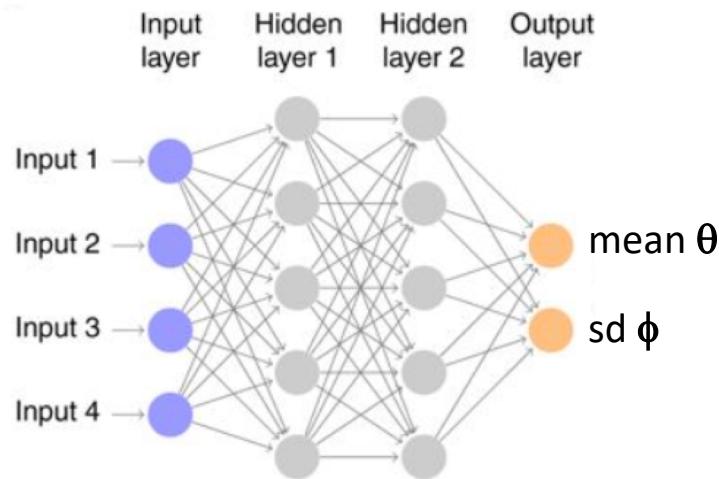
SCVIS / VASC use a special kind of neural network called autoencoder



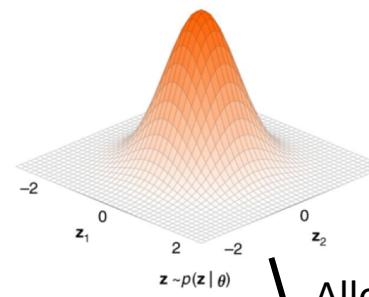
The autoencoder can be constructed with bottleneck with only two “neurons” which will represent your data in a reduced space.

# How SCVIS and VASC work

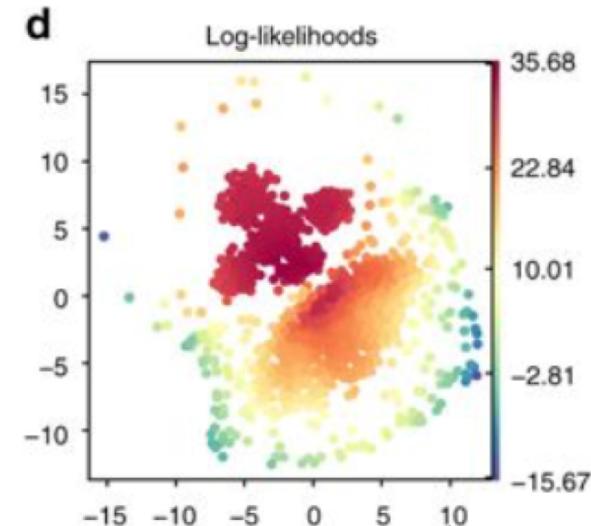
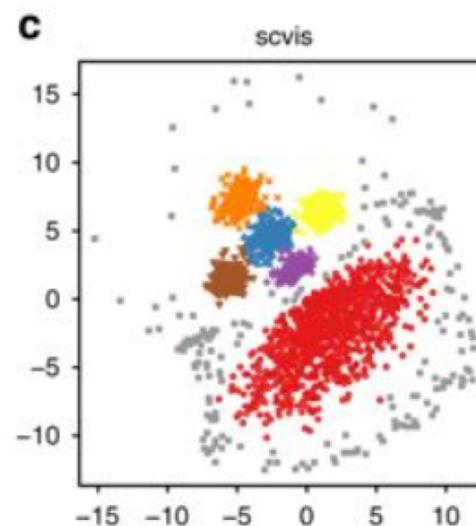
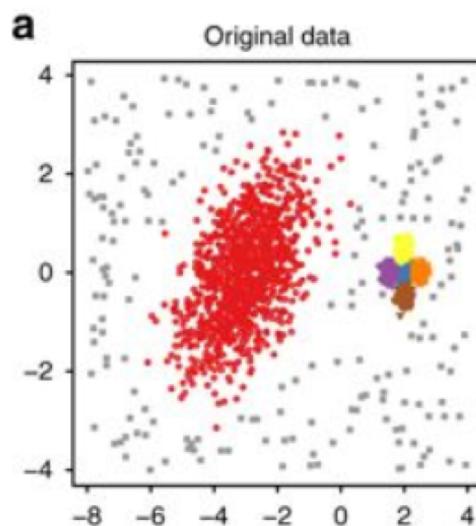
SCVIS and VASC are based specifically in **variational autoencoders**



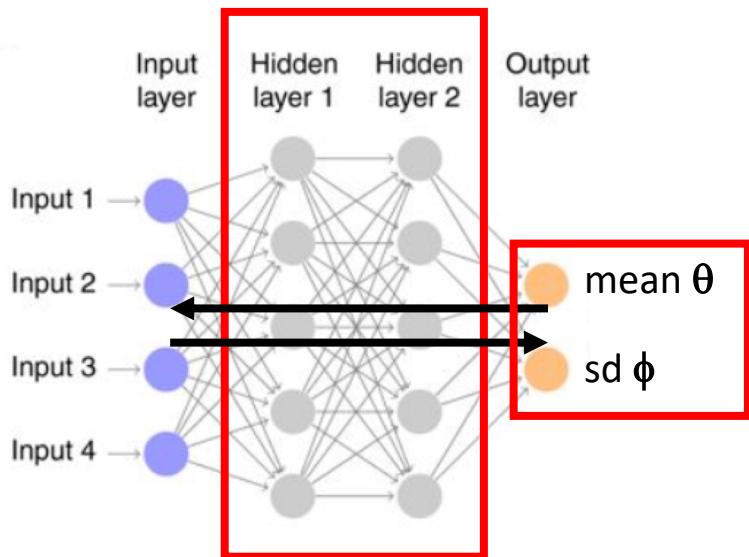
SCVIS and VASC calculates the probability that a cell came from a 2-D normal distribution with mean  $\theta$  and sd  $\phi$ . This way, every cell cluster has its own mean and sd.



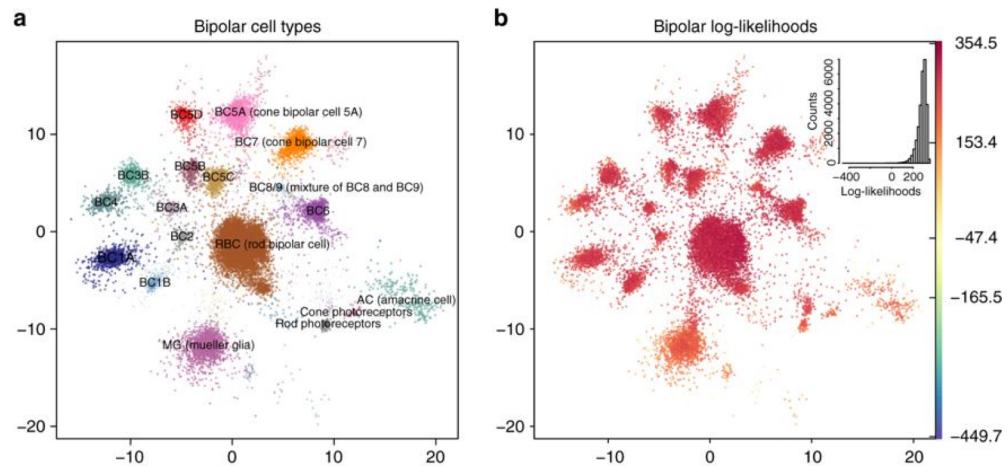
Allows measuring uncertainty



# How SCVIS and VASC work



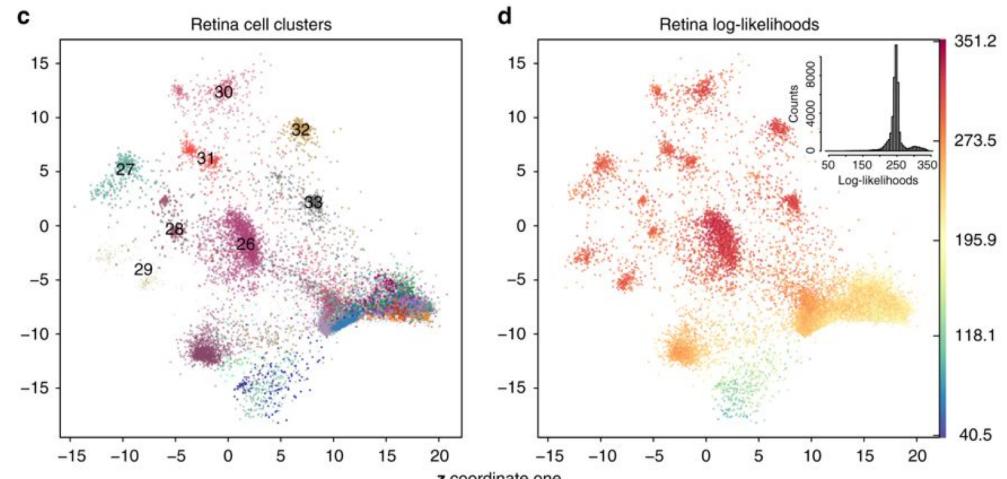
TRAINING dataset -> construction of the scvis map



Training a neural layer to assign cell into low dimensional space has some advantages:

1. Allows assignment of new cells to the low dimension
2. Allows reconstruction of the dataset from the low dimension
3. It can do the above with a measure of uncertainty

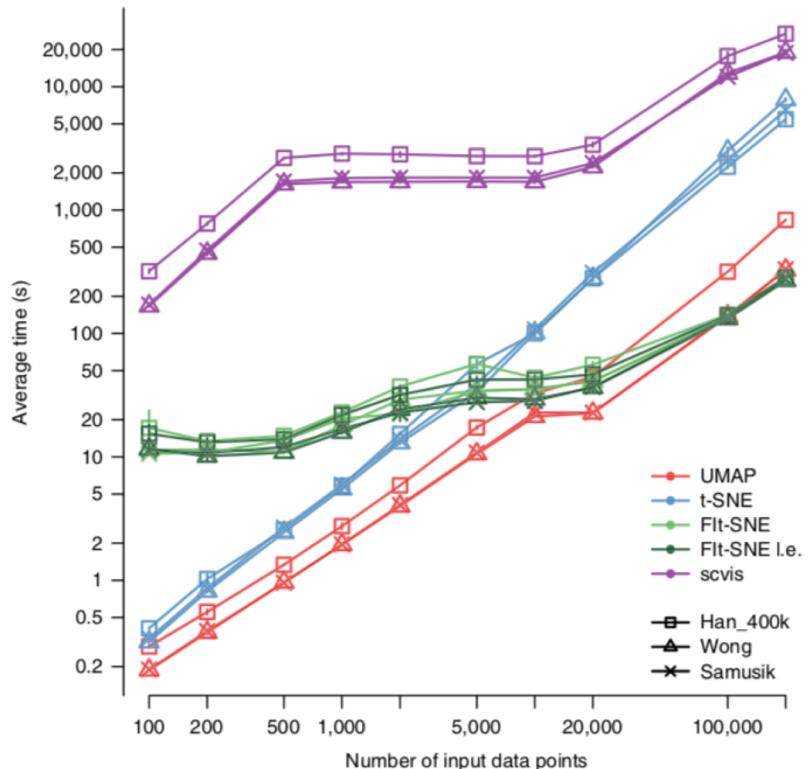
NEW dataset -> assignment of cells to the map above



# SCVIS and VASC: summary

## To keep in mind:

- It is a NON-LINEAR autoencoder-based method of dimensionality reduction
- Training SCVIS and VASC are computationally expensive -  $O(TN^2D + TN^2d)$
- However, assigning new points is very fast -  $O(N)$
- Can assign new samples (even from other datasets) into the lower dimensions – data integration
- Convenient for big datasets i.e.  $> 100.000$  cells



# Wrap-up

---

So which one should you use?

# Single cell workflows

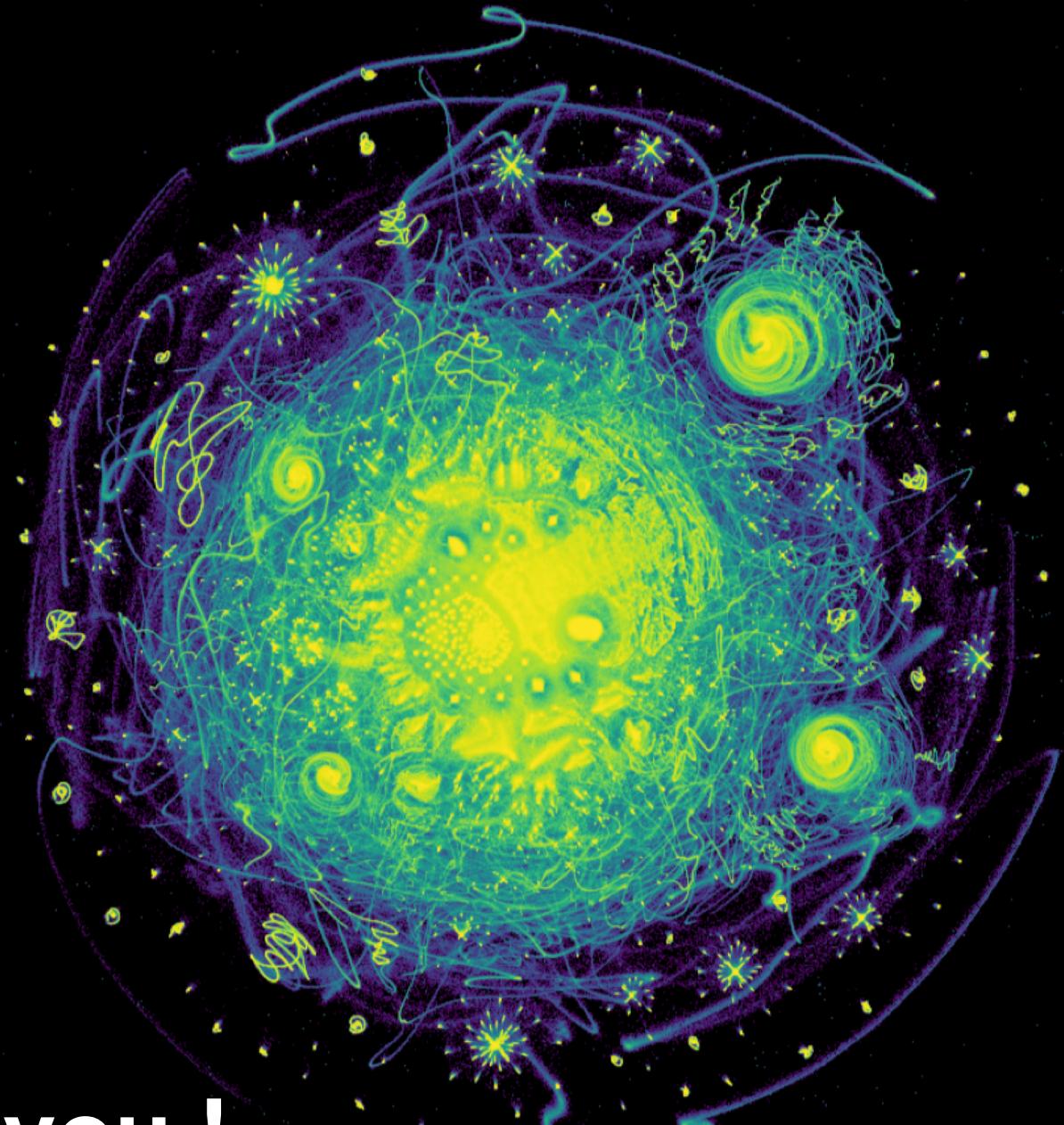
(January 2019)

	Seurat v3	Scater	Pagoda v2	Monocle v3
→	PCA ICA -	PCA - MDS	PCA - -	PCA ICA -
→	tSNE (BH, Flt) UMAP -	tSNE (BH) UMAP -	tSNE (BH) - LargeVis Isomap -	tSNE (BH) UMAP - -
	Diff. Maps -	Diff. Maps -	DDRTree -	
	PHATE	-	-	-

SCVIS and VASC are fairly recent and under development, but maybe soon in the future.

Paper comparing lots of dimensionality reduction techniques:

<https://www.biorxiv.org/content/biorxiv/early/2018/06/28/120378.full.pdf>



Thank you !