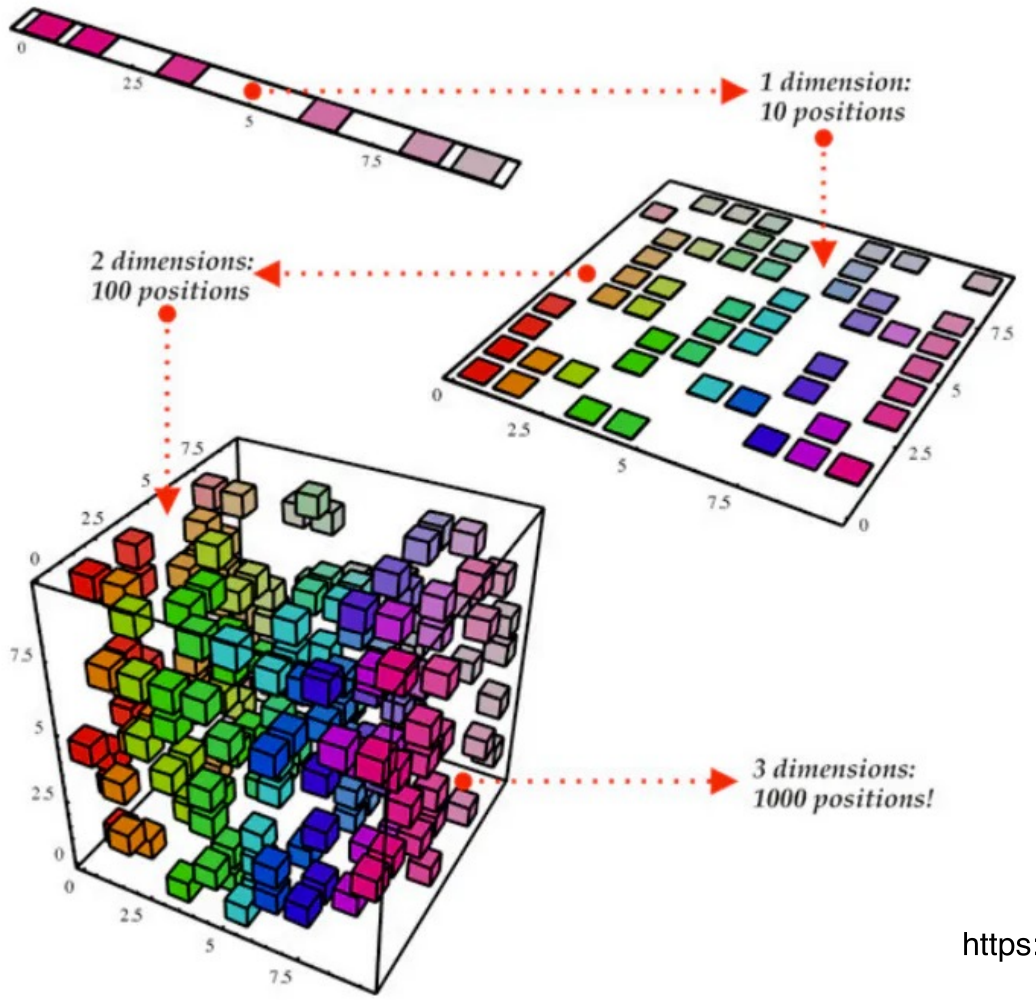
Dimensionality Reduction & Integration

Single Cell Transcriptomics in Python

Alex Lederer

The curse of dimensionality

- More dimensions = exponentially more possible cell “positions” in gene expression space!



https://www.pinecone.io/learn/dimensionality-reduction/

Why do we perform dimensionality reduction?

- To **simplify** our complex data!
- **Reduce** number of features (genes) without losing information
- Identify **signal** and remove **redundancies** (**noise**) in the data
- Speed up **computational time** for downstream steps
- **Facilitate clustering**, since some algorithms struggle with too many dimensions
- Easier data **visualization**

Limits of dimensionality reduction

- Low dimensional representations (two-dimensions) **will not capture the full variability** represented by high dimensional data (20K dimensions)
- In other words: **information will still be lost**
- **Distances** between cells/clusters in low dimensions might not reflect the true high dimensional data.
- Biological conclusions **should not** be drawn from UMAP and tSNE (but PCA is a little better…)

Two types of dimensionality reduction

1. Geometric dimensionality reduction

- The contribution or “weight” of each gene feature for each principal component axis can be directly calculated

PCA: principal component analysis (https://hbctraining.github.io/scRNA-seq/lessons/05_normalization_and_PCA.html)

Two types of dimensionality reduction

1. Geometric dimensionality reduction

- The contribution or “weight” of each gene feature for each principal component axis can be directly calculated

PCA: principal component analysis (https://hbctraining.github.io/scRNA-seq/lessons/05_normalization_and_PCA.html)

2. Non-geometric embedding methods

- The relationship between gene features and the lower dimensional representation is lost; there is a degree of randomness in the embedding computation.

tSNE: T-distributed stochastic neighborhood embedding (https://jmlr.org/papers/v9/vandermaaten08a.html)

UMAP: Uniform manifold approach and projection (https://arxiv.org/abs/1802.03426)

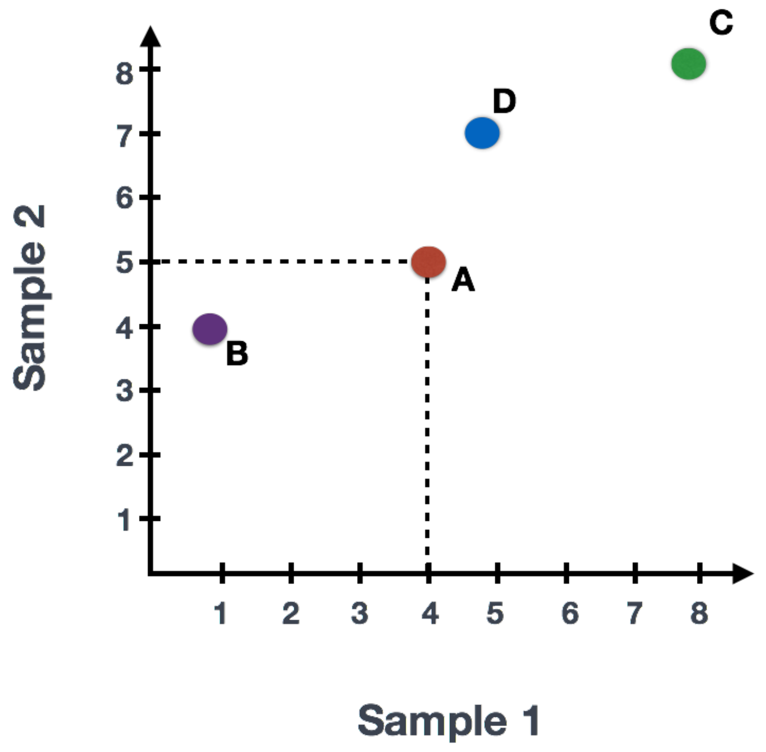
Question

Principal component analysis (PCA)

Principal component analysis (PCA)

Goal: to emphasize **variation** as well as **similarity** in a dataset, among several thousand highly variable genes.

Let’s consider an example of two samples (cells) and four genes:

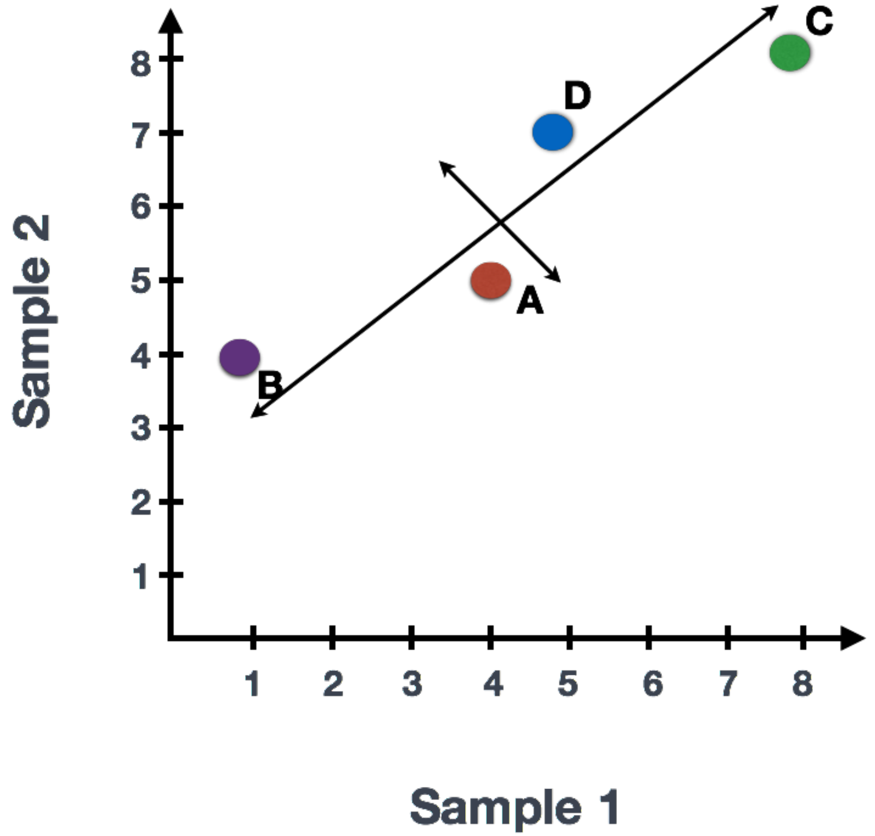


	Sample 1	Sample 2
Gene A	4	5
Gene B	1	4
Gene C	8	8
Gene D	5	7

https://hbctraining.github.io/scRNA-seq/lessons/05_normalization_and_PCA.html

Principal component analysis (PCA)

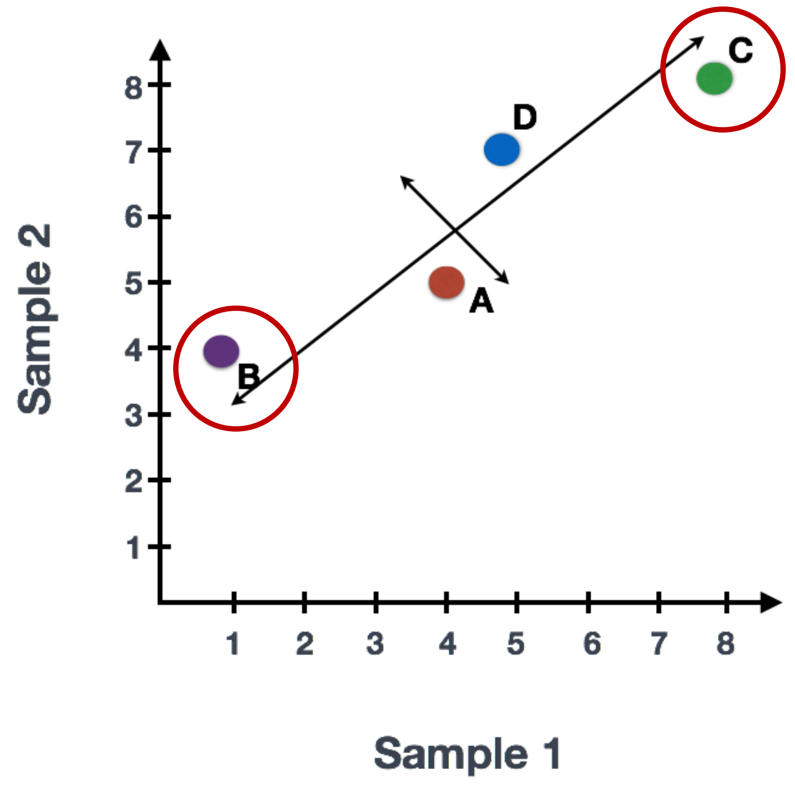
We can draw two lines to represent the largest and second largest **axes of variation** among all four genes.

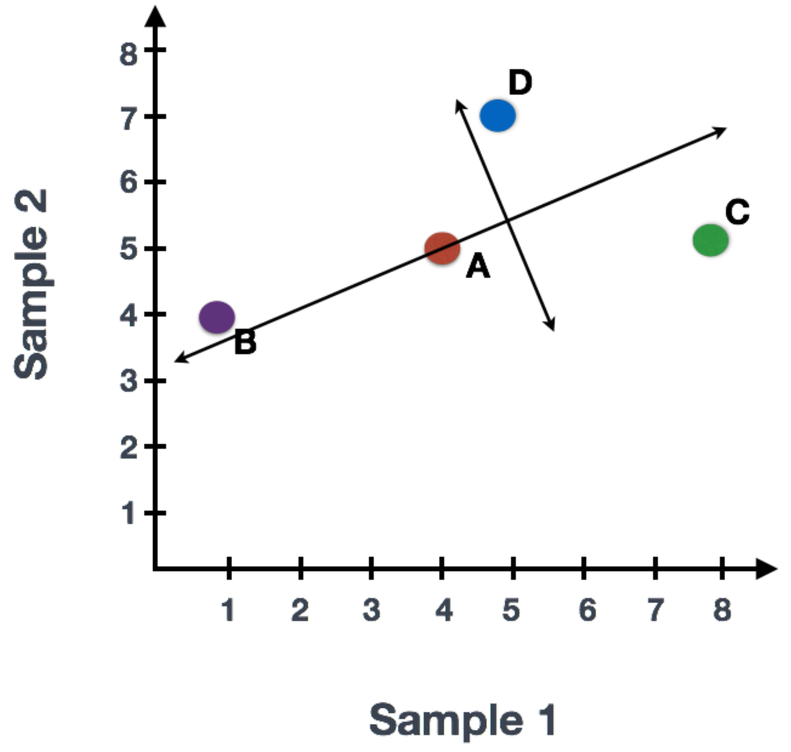


https://hbctraining.github.io/scRNA-seq/lessons/05_normalization_and_PCA.html

Principal component analysis (PCA)

Genes at the extremes of each axis of variation contribute the most to that “component”

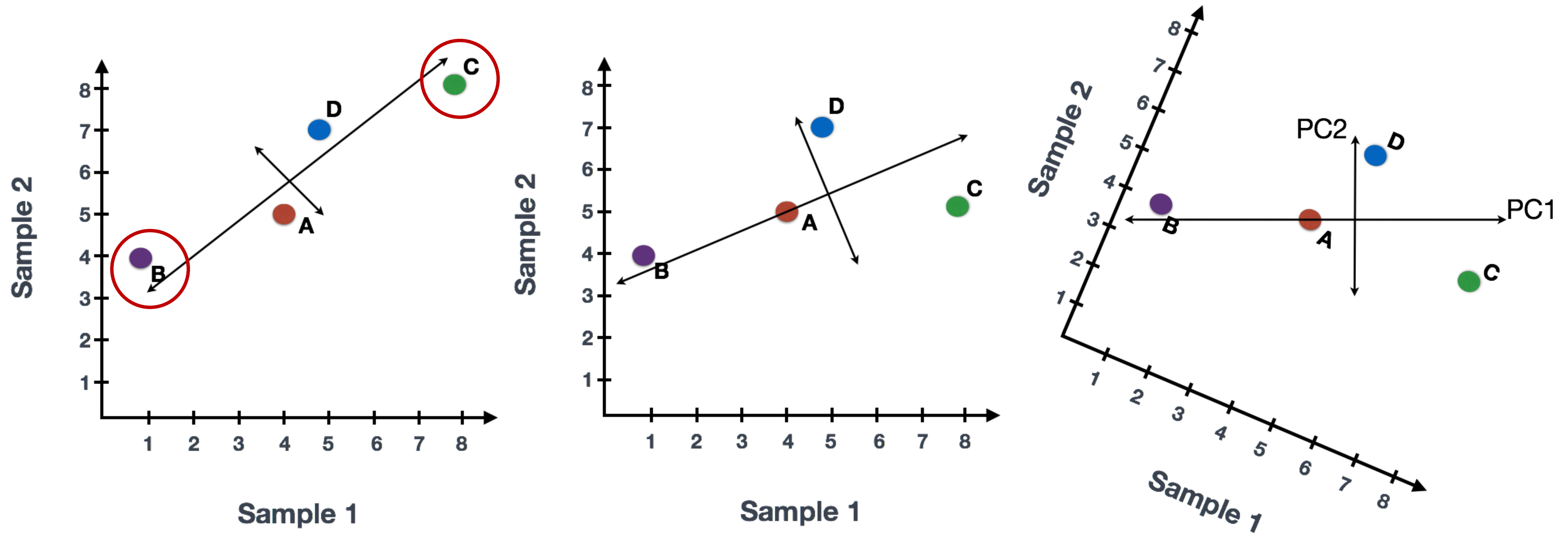




https://hbctraining.github.io/scRNA-seq/lessons/05_normalization_and_PCA.html

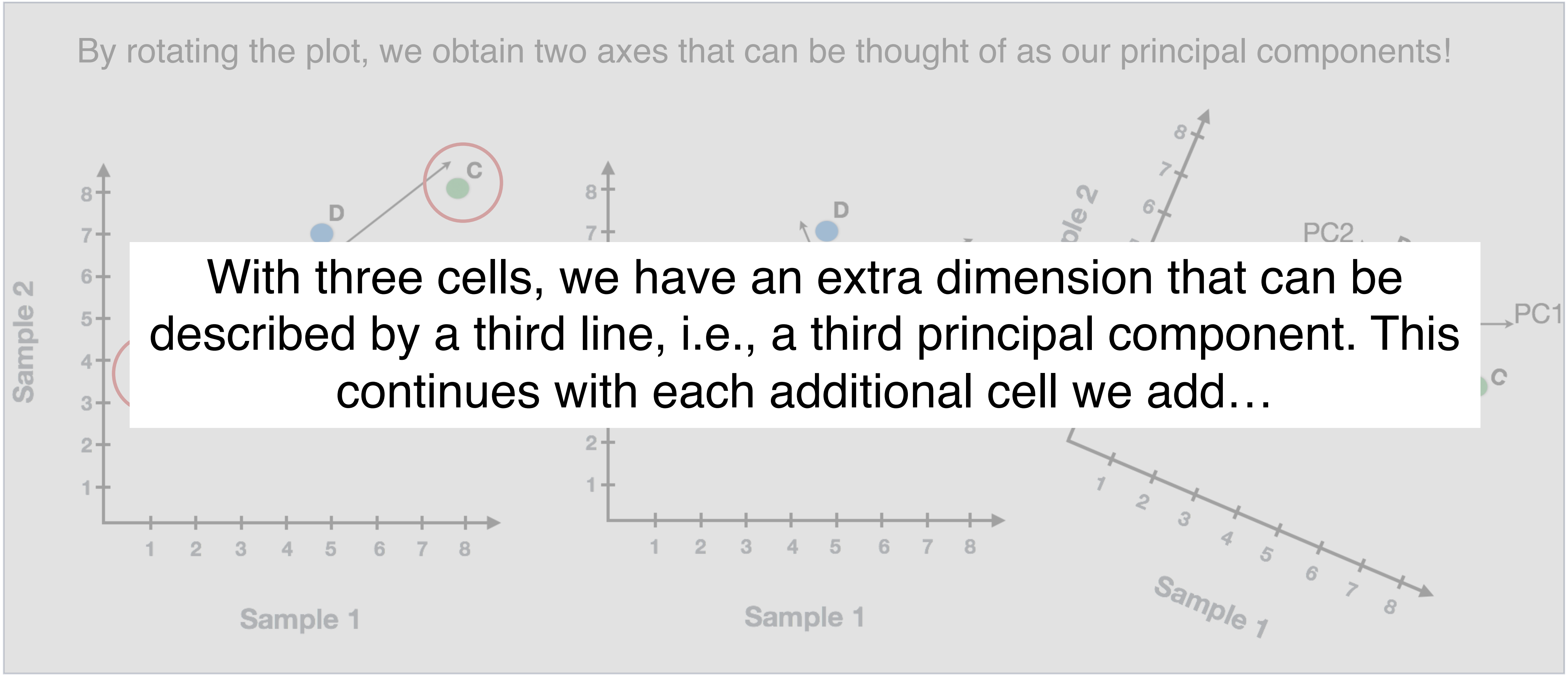
Principal component analysis (PCA)

By rotating the plot, we obtain two axes that can be thought of as our principal components!



https://hbctraining.github.io/scRNA-seq/lessons/05_normalization_and_PCA.html

Principal component analysis (PCA)

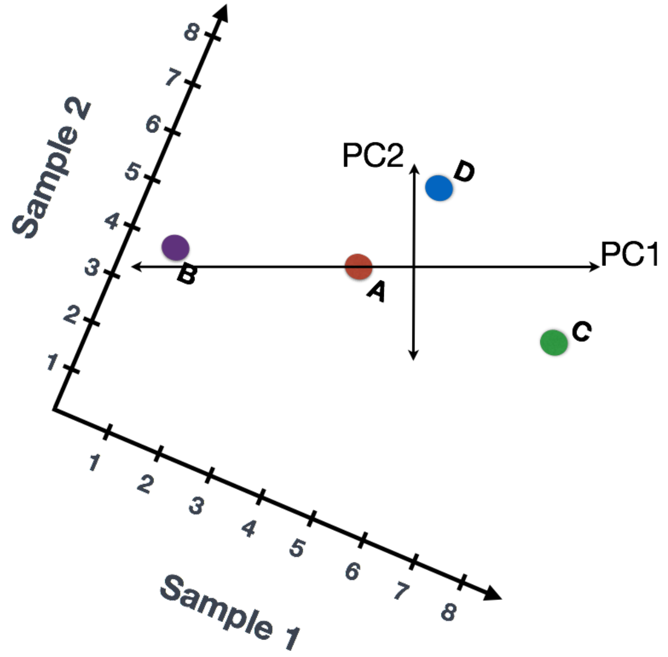

By rotating the plot, we obtain two axes that can be thought of as our principal components!
Sample 2
Sample 1
C
D
PC2
PC1
1
2
3
4
5
6
7
8

With three cells, we have an extra dimension that can be described by a third line, i.e., a third principal component. This continues with each additional cell we add…

https://hbctraining.github.io/scRNA-seq/lessons/05_normalization_and_PCA.html

Principal component analysis (PCA)

Each gene is assigned a score (loading) that weighs its contribution to each principal component.



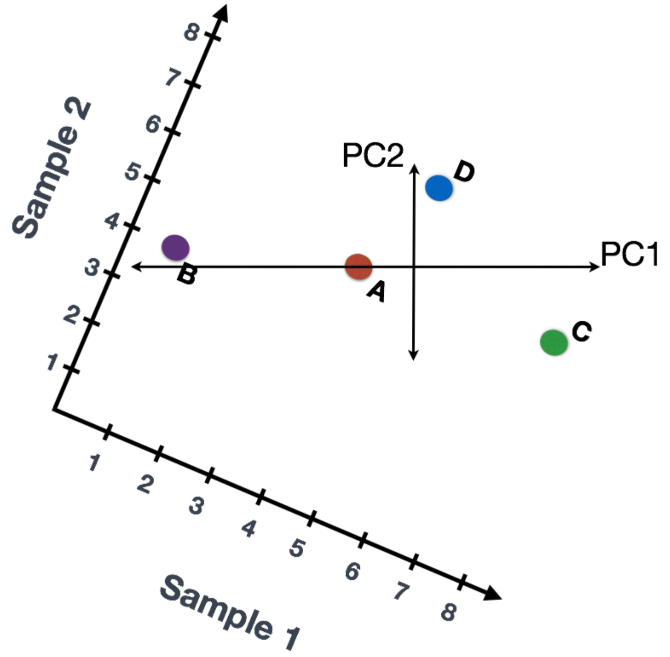
"loadings"

	Sample 1	Sample 2	Influence on PC1	Influence on PC2
Gene A	4	5	-2	0.5
Gene B	1	4	-10	1
Gene C	8	8	8	-5
Gene D	5	7	1	6

https://hbctraining.github.io/scRNA-seq/lessons/05_normalization_and_PCA.html

Principal component analysis (PCA)

Each gene is assigned a score (loading) that weighs its contribution to each principal component.



"loadings"

	Sample 1	Sample 2	Influence on PC1	Influence on PC2
Gene A	4	5	-2	0.5
Gene B	1	4	-10	1
Gene C	8	8	8	-5
Gene D	5	7	1	6

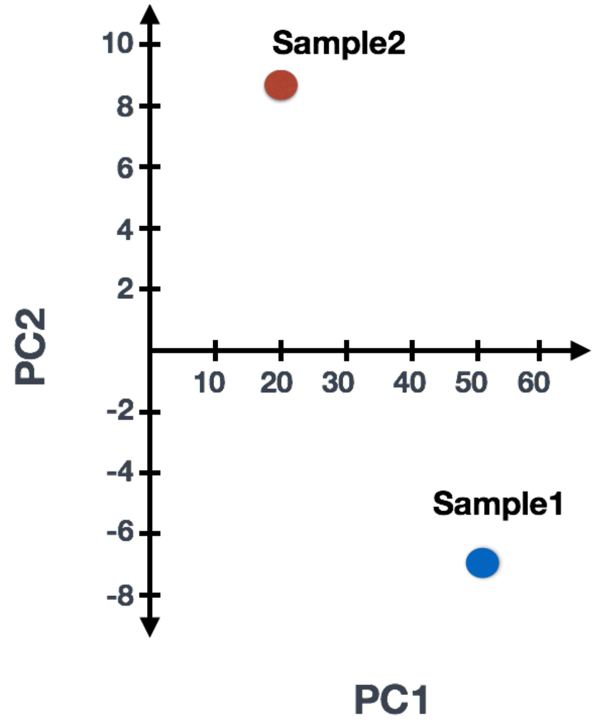
The position of each cell on a PCA plot is then determined by the sum of its gene counts and the loadings of each gene. For **Sample 1 (Cell 1)**:

PC1 score = (4 * -2) + (1 * -10) + (8 * 8) + (5 * 1) = 51

PC2 score = (4 * 0.5) + (1 * 1) + (8 * -5) + (5 * 6) = -7

https://hbctraining.github.io/scRNA-seq/lessons/05_normalization_and_PCA.html

Principal component analysis (PCA)



	PC1	PC2
Sample1	51	-7
Sample2	21	8.5

The position of each cell on a PCA plot is then determined by the sum of its gene counts and the loadings of each gene. For **Sample 1 (Cell 1)**:

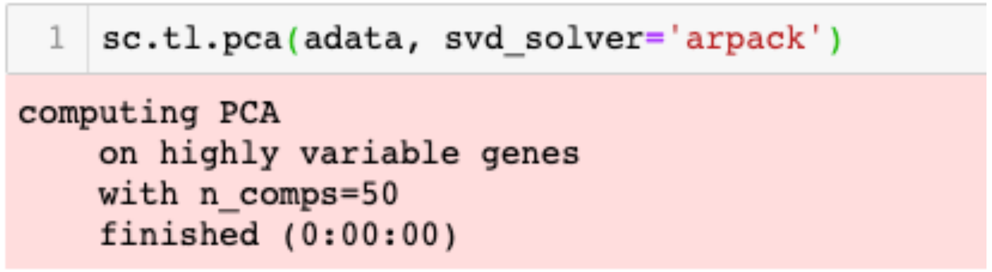
PC1 score = (4 * -2) + (1 * -10) + (8 * 8) + (5 * 1) = 51

PC2 score = (4 * 0.5) + (1 * 1) + (8 * -5) + (5 * 6) = -7

https://towardsdatascience.com/a-one-stop-shop-for-principal-component-analysis-5582fb7e0a9c

https://hbctraining.github.io/scRNA-seq/lessons/05_normalization_and_PCA.html

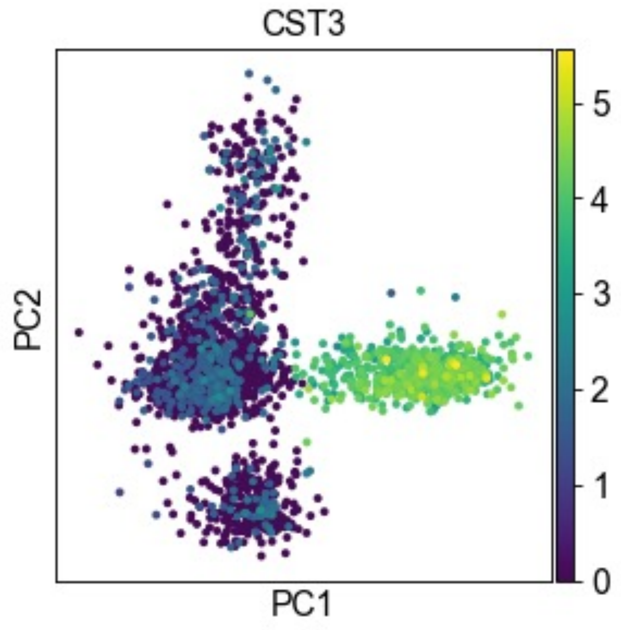
PCA in scanpy



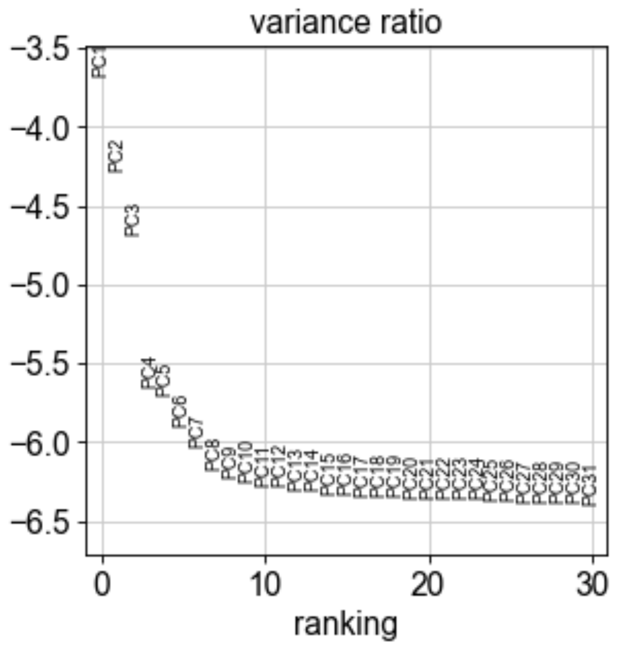
```
sc.tl.pca(adata, svd_solver='arpack')
```

```
computing PCA
    on highly variable genes
    with n_comps=50
    finished (0:00:00)
```

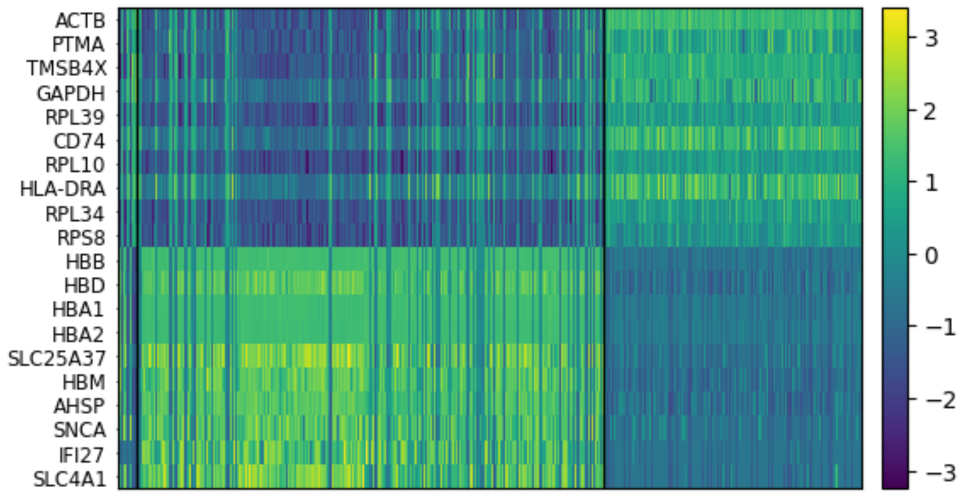
sc.pl.pca



sc.pl.pca_variance_ratio



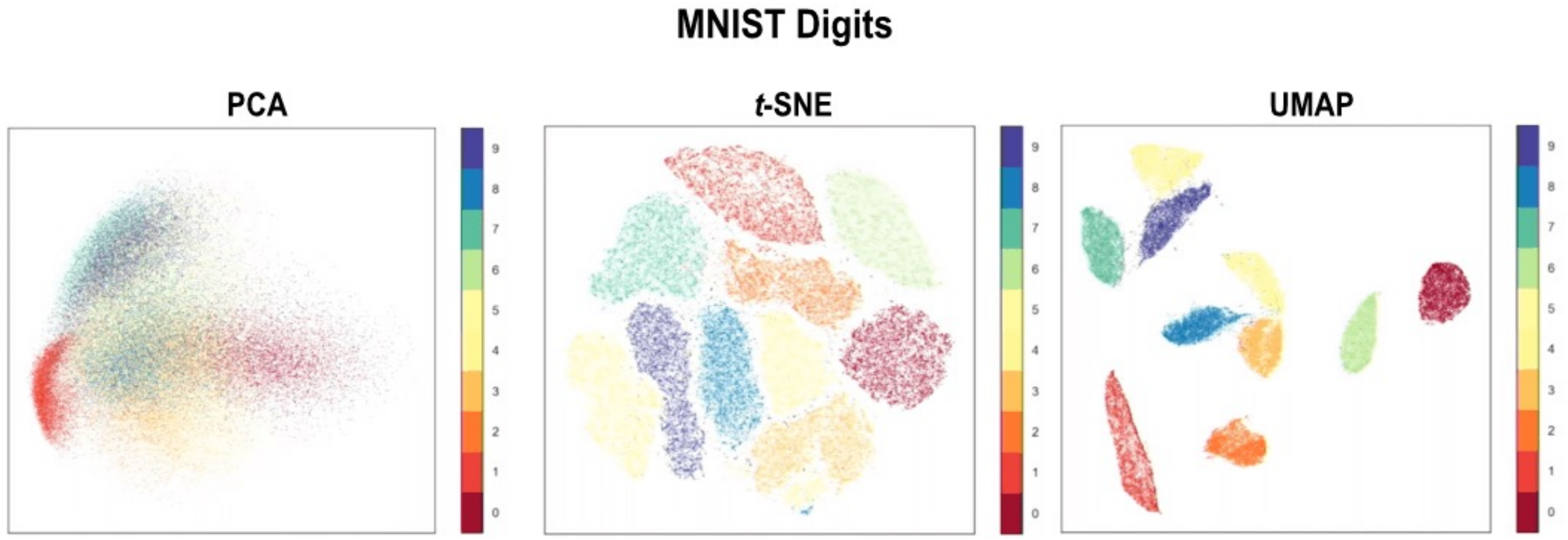
More complex plots
sc.pl.heatmap



Question(s)

UMAP/tSNE embedding methods

Limitations of PCA



The two principal components from PCA **do not always go far enough** to discriminate different data types.

Image source: https://meta.caspershire.net/umap/

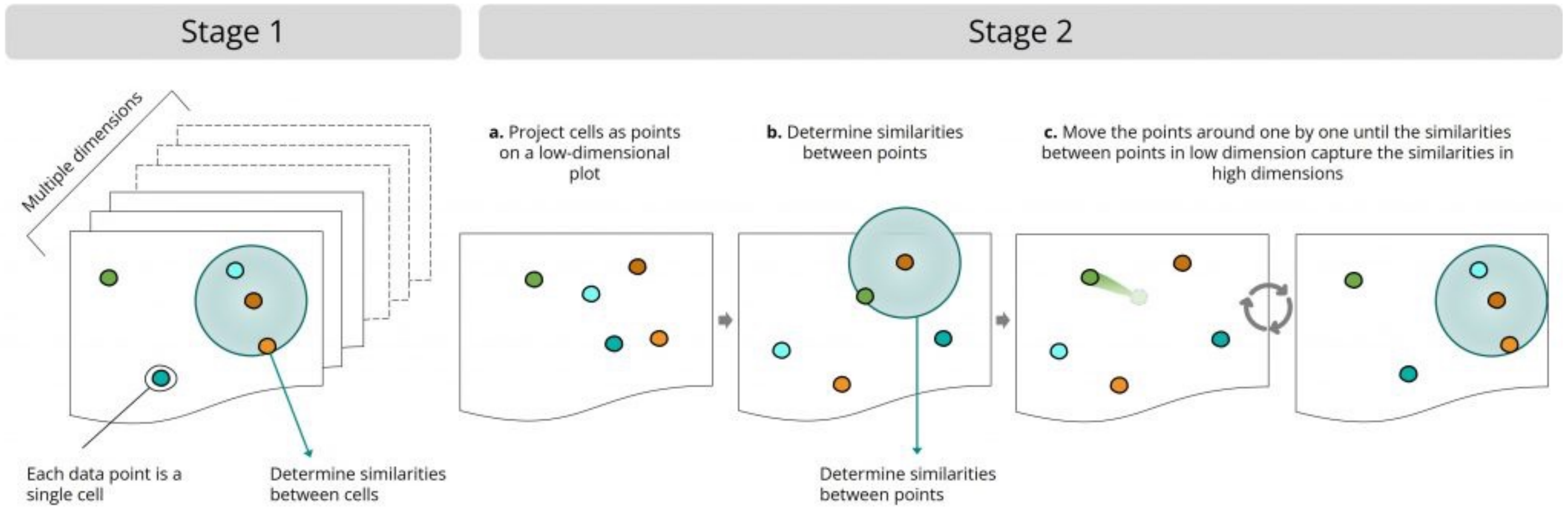
UMAP vs tSNE

- UMAP better preserves global structure compared to tSNE
- UMAP is significantly faster on larger datasets (https://umap-learn.readthedocs.io/en/latest/performance.html)

- More intuitive information on tSNE: https://towardsdatascience.com/t-sne-clearly-explained-d84c537f53a
- Here I will focus more on UMAP

UMAP explained: general steps

1. Compute the similarity between (neighborhood graph with Euclidean distance)
2. Project the cells as points on a low-dimensional (2D) plot
3. Calculate the similarities between points in the 2D space compared to the high-dimensional space.
4. Randomly adjust position of a few points and recompute distances until convergence.

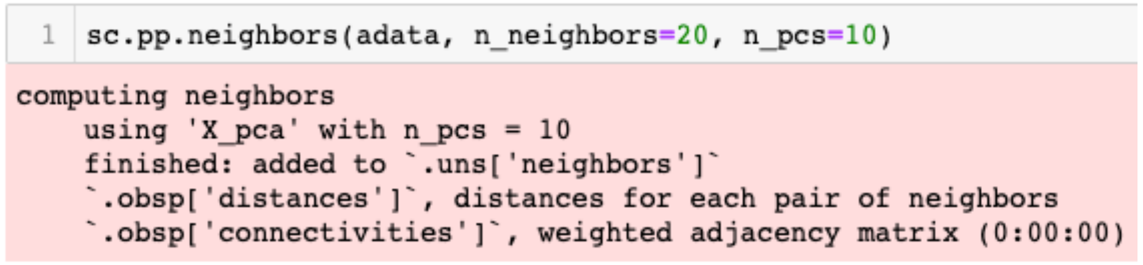

Stage 1
Stage 2
Multiple dimensions
a. Project cells as points on a low-dimensional plot
b. Determine similarities between points
c. Move the points around one by one until the similarities between points in low dimension capture the similarities in high dimensions
Each data point is a single cell
Determine similarities between cells
Determine similarities between points

Interactive UMAP exploration

https://pair-code.github.io/understanding-umap/

Computing and embedding the neighborhood graph

- Compute the neighborhood graph of cells using the PCA representation of the data matrix.

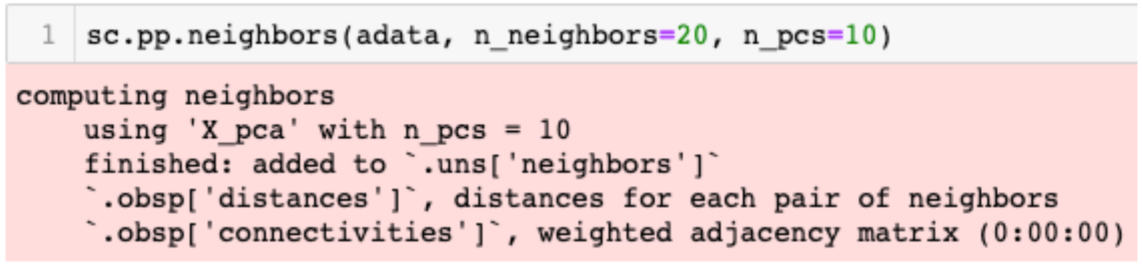


```
sc.pp.neighbors(adata, n_neighbors=20, n_pcs=10)
```

```
computing neighbors
    using 'X_pca' with n_pcs = 10
    finished: added to `.uns['neighbors']`
    `.obsp['distances']`, distances for each pair of neighbors
    `.obsp['connectivities']`, weighted adjacency matrix (0:00:00)
```

Computing and embedding the neighborhood graph

- Compute the neighborhood graph of cells using the PCA representation of the data matrix.



```
sc.pp.neighbors(adata, n_neighbors=20, n_pcs=10)
```

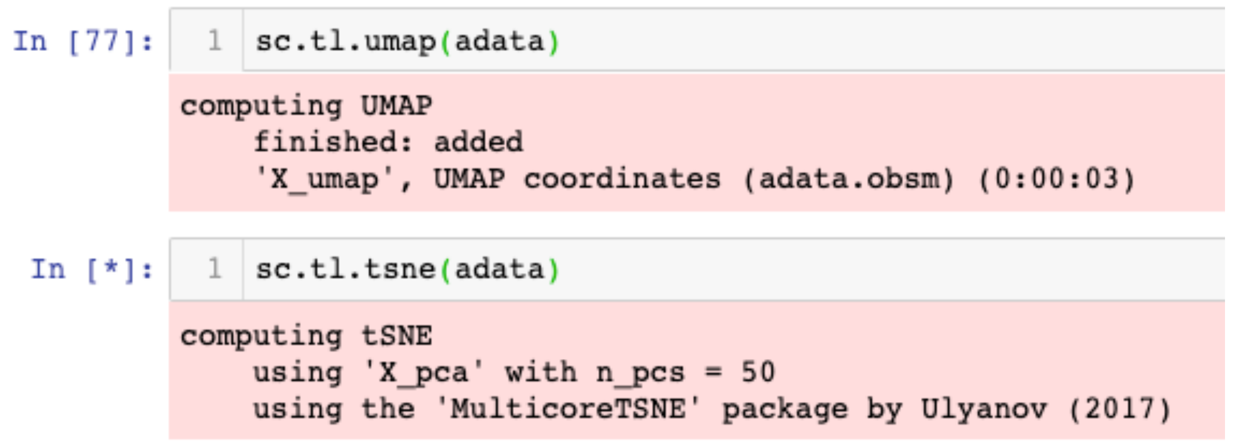
```
computing neighbors
    using 'X_pca' with n_pcs = 10
    finished: added to `.uns['neighbors']`
    `.obsp['distances']`, distances for each pair of neighbors
    `.obsp['connectivities']`, weighted adjacency matrix (0:00:00)
```

Number of nearest neighbors (*n_neighbors)*
- Low values = embedding captures more noise
- High values = smoother embedding (capture less biological variability)

Number of principal components *(n_pcs)*
- Low values = less cell type discrimination (you risk underestimating sample heterogeneity)
- High values = more cell type discrimination (but you risk including noise!)

Computing and embedding the neighborhood graph

- Embedding the graph can be performed using either tSNE or UMAP algorithms



```
In [77]:  sc.tl.umap(adata)
```

```
computing UMAP
    finished: added
    'X_umap', UMAP coordinates (adata.obsm) (0:00:03)
```

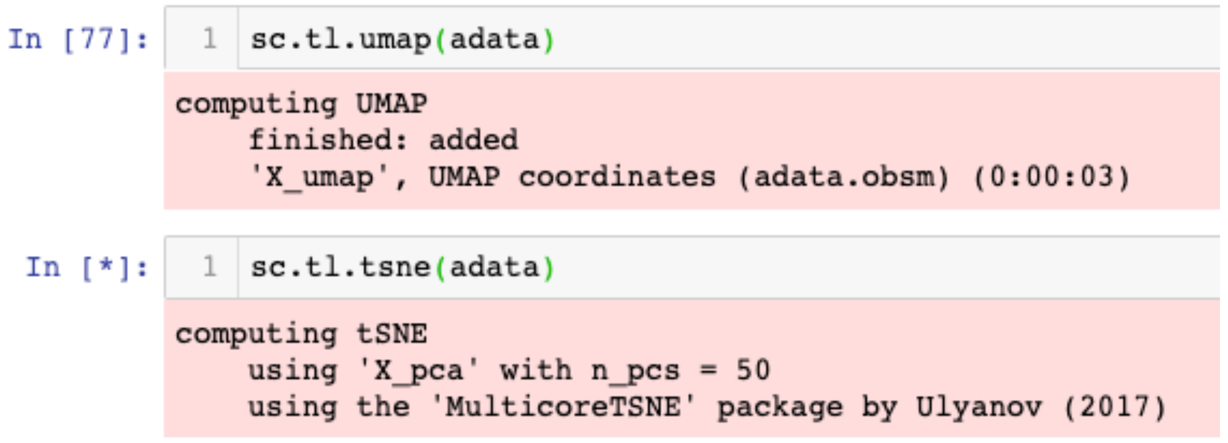
```
In [*]:  sc.tl.tsne(adata)
```

```
computing tSNE
    using 'X_pca' with n_pcs = 50
    using the 'MulticoreTSNE' package by Ulyanov (2017)
```

https://www.pinecone.io/learn/dimensionality-reduction/

Computing and embedding the neighborhood graph

- Embedding the graph can be performed using either tSNE or UMAP algorithms



```
In [77]:  sc.tl.umap(adata)

computing UMAP
    finished: added
    'X_umap', UMAP coordinates (adata.obsm) (0:00:03)

In [*]:   sc.tl.tsne(adata)

computing tSNE
    using 'X_pca' with n_pcs = 50
    using the 'MulticoreTSNE' package by Ulyanov (2017)
```

UMAP hyperparameters:

- **Number of nearest neighbors (n_neighbors)**: controls how UMAP balances local versus global structure. This is adjusted with the sc.pp.neighbors function.
  - Low values = more local structure
  - High values = represent the big-picture structure, but losing fine detail
- **Minimum distance between points in low-dimensional space (min_dist)**: controls how tightly UMAP clumps data points together. This is adjusted with the sc.tl.umap function.
  - Low values = more tightly packed embeddings
  - High values = points packed together more loosely, focusing on the broad structure

https://www.pinecone.io/learn/dimensionality-reduction/

Limitations of UMAP and embedding methods

The specious art of single-cell genomics

Tara Chari, Lior Pachter

Published: August 17, 2023 • https://doi.org/10.1371/journal.pcbi.1011288

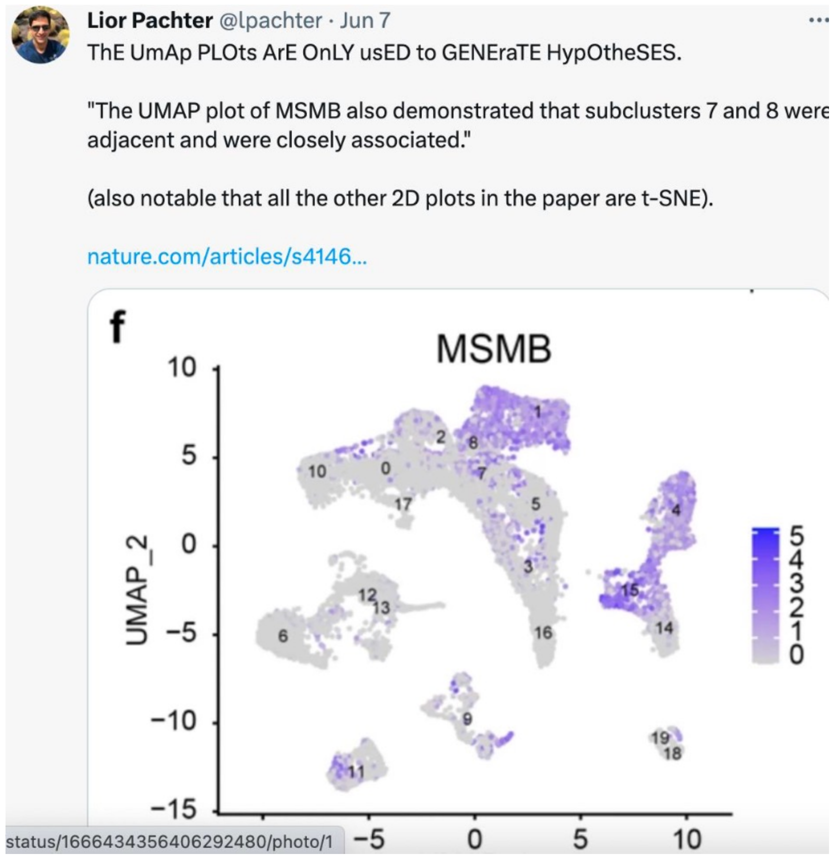
- Dimensionality reduction from tens of thousands to two dimensions introduces distortions into the data

Seeing data as t-SNE and UMAP do

Vivien Marx

Nature Methods (2024) | Cite this article

https://www.nature.com/articles/s41592-024-02301-x



Lior Pachter @lpachter · Jun 7

ThE UmAp PLOts ArE OnLY usED to GENEraTE HypOtheSES.

"The UMAP plot of MSMB also demonstrated that subclusters 7 and 8 were adjacent and were closely associated."

(also notable that all the other 2D plots in the paper are t-SNE).

nature.com/articles/s4146...

status/1666434356406292480/photo/1

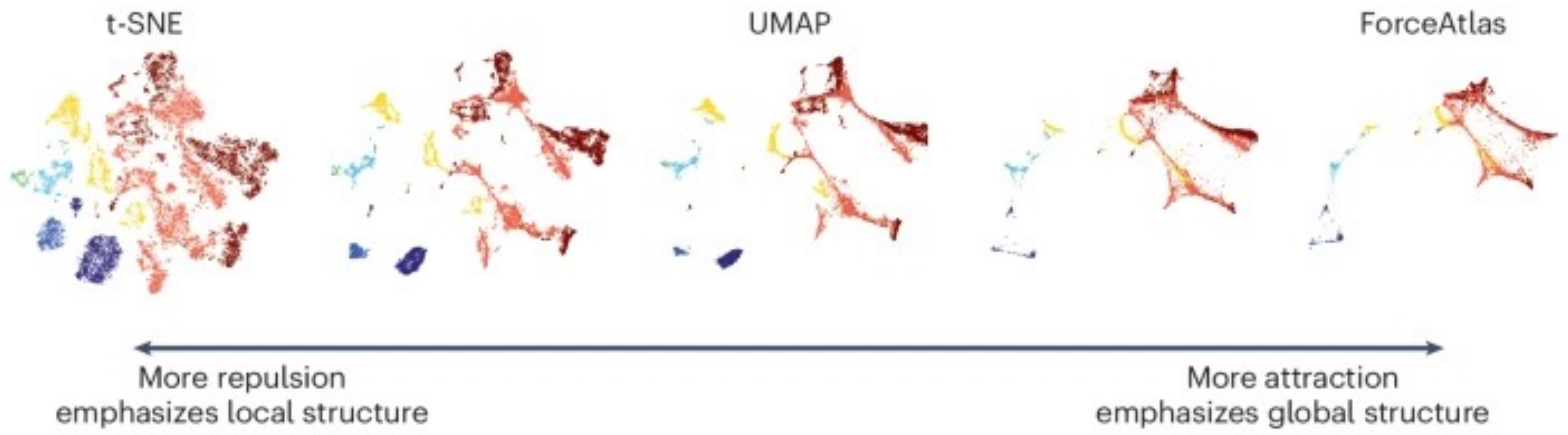
Limitations of UMAP and embedding methods

Seeing data as t-SNE and UMAP do

Vivien Marx

Nature Methods (2024) | Cite this article

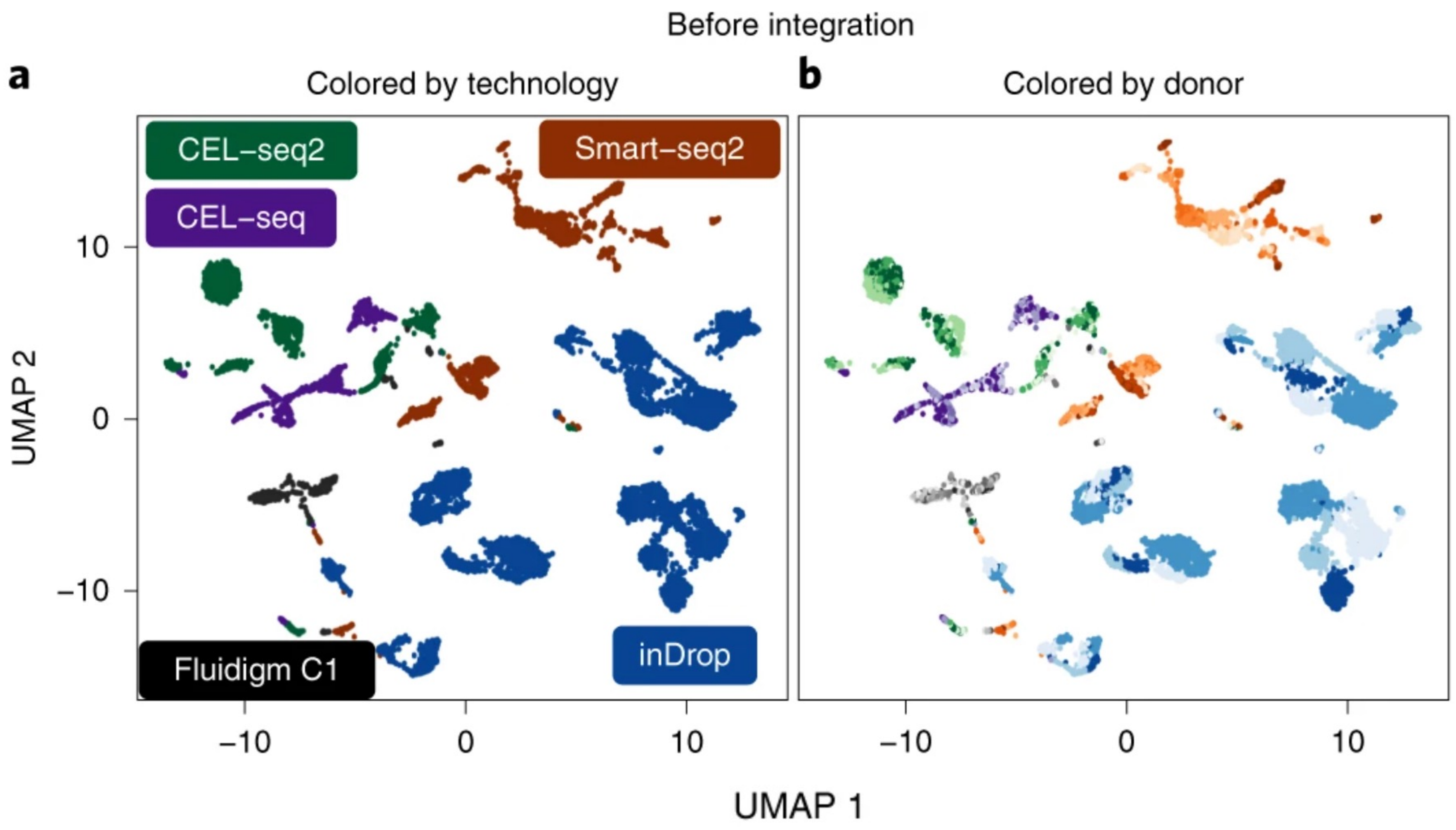
https://www.nature.com/articles/s41592-024-02301-x



Data integration

Integration analysis

Why do we integrate single-cell data?



From: https://doi.org/10.1038/s41592-019-0619-0

Common steps of integration algorithms

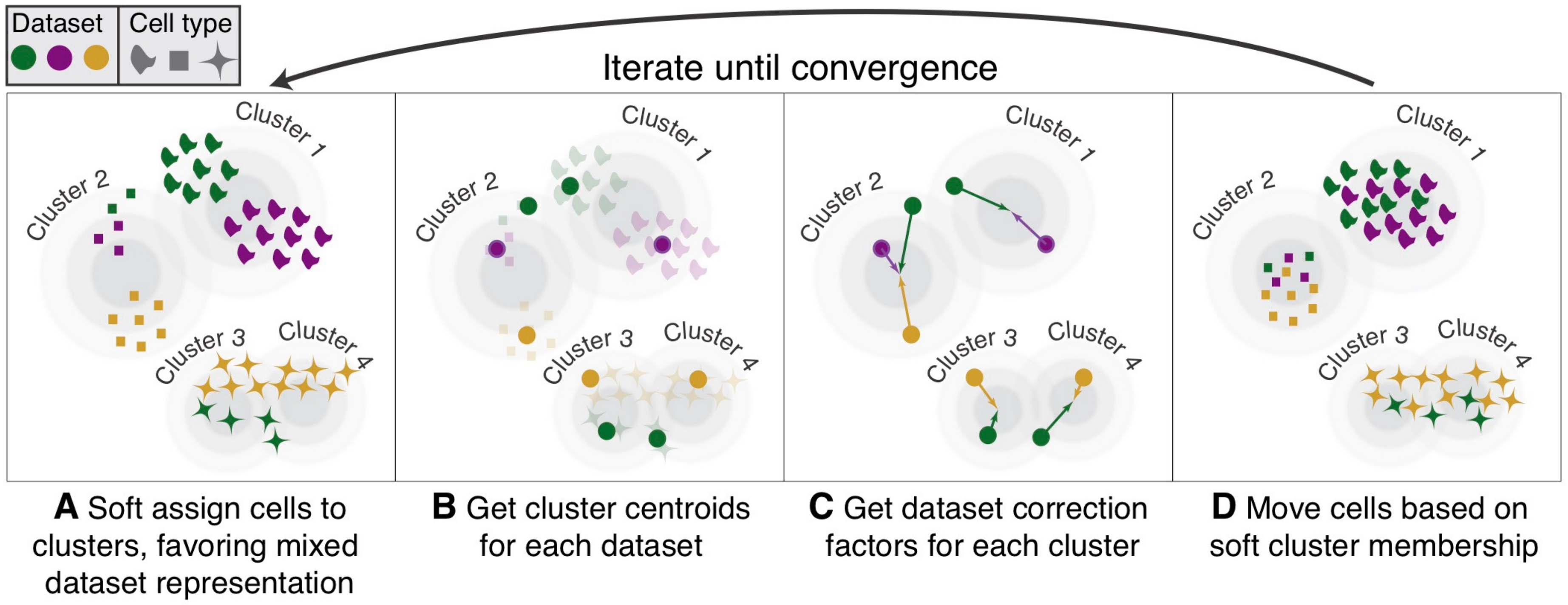
1. Find similar cells across batches by computing a distance between cells in a certain space (i.e., PCA, gene feature space).

2. Compute a data adjustment based on correspondences between cells from different batches

3. Apply the adjustment and repeat until certain criteria (i.e., mean distance between cells in different batches) are reached

Data integration with Harmony

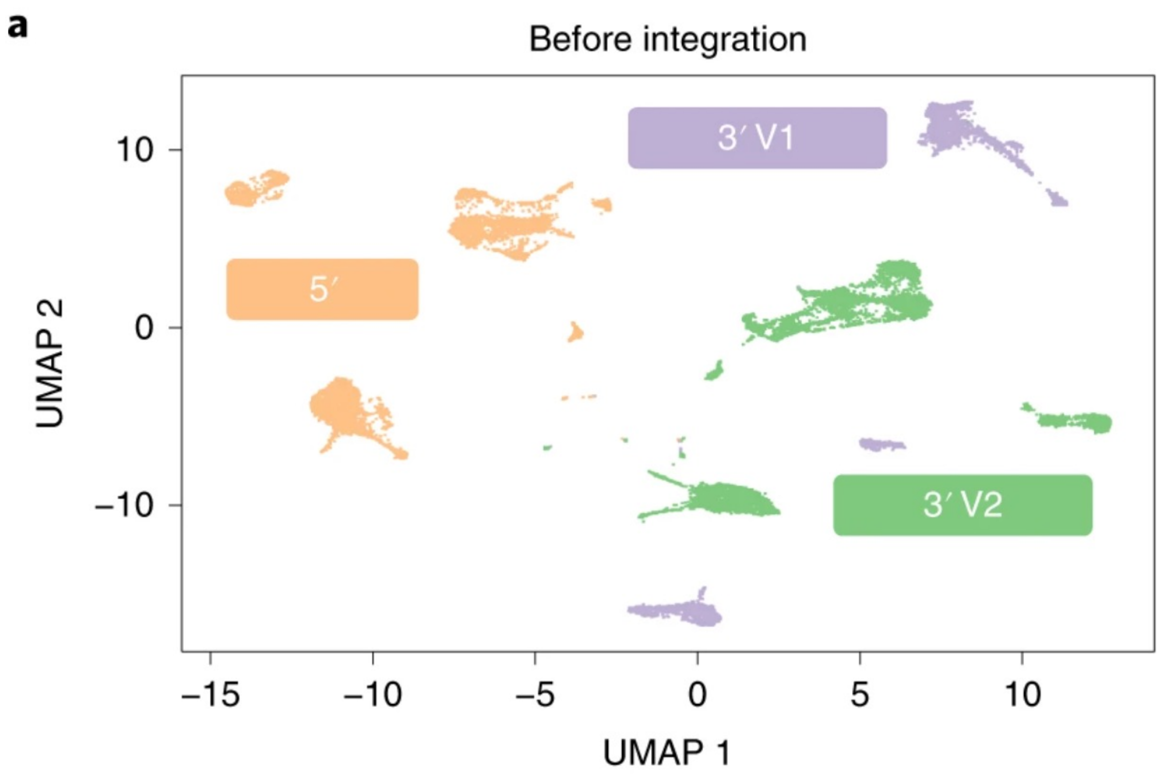
- An iterative algorithm to adjust principal components and reduce batch effect between samples
- Modifies the PCs but not the count data (unlike CCA in Seurat)

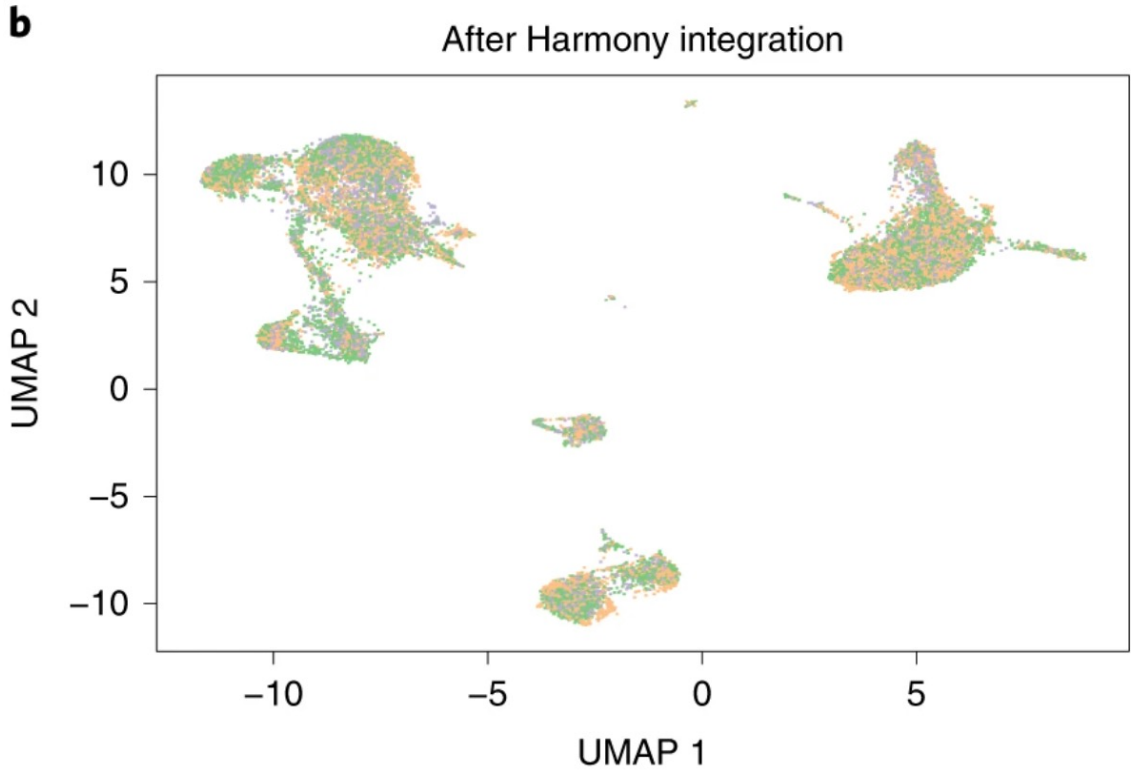

Dataset
Cell type
Iterate until convergence
Cluster 1
Cluster 2
Cluster 3
Cluster 4
A Soft assign cells to clusters, favoring mixed dataset representation
B Get cluster centroids for each dataset
C Get dataset correction factors for each cluster
D Move cells based on soft cluster membership

Fast, sensitive and accurate integration of single-cell data with Harmony (Nature Methods 2019)
https://doi.org/10.1038/s41592-019-0619-0

Data integration with Harmony

Integration of three PBMC datasets from different 10X technologies





From: https://doi.org/10.1038/s41592-019-0619-0

Many integration methods are only available in R

- Harmony (https://doi.org/10.1101/461954)
- MNNcorrect (https://doi.org/10.1038/nbt.4091)
- RPCA + anchors (Seurat v3)(https://doi.org/10.1101/460147)
- CCA + anchors (Seurat v3) (https://doi.org/10.1101/460147)
- CCA + dynamic time warping (Seurat v2; https://doi.org/10.1038/nbt.4096)
- LIGER (https://doi.org/10.1101/459891)
- Conos (https://doi.org/10.1101/460246)
- Scanorama (https://doi.org/10.1101/371179)
- scMerge (https://doi.org/10.1073/pnas.1820006116)
- STACAS (https://doi.org/10.1093/bioinformatics/btaa755)

Benchmarking study of 68 different methods and preprocessing choices on 1.2 million single cell: *"Benchmarking atlas-level data integration in single-cell genomics"* (https://www.nature.com/articles/s41592-021-01336-8)

Package list compiled by Rachel Marcone

Limitation: technical variability versus biological variability

Too “strong” of an integration can remove all technical variability, but also biological variability (i.e., differences between a diseased and control sample.

Potential solution: **cell type label transfer** approaches (`sc.tl.ingest`)

- Uses PCs and neighborhood from **reference dataset** to infer label information for a new **unlabeled dataset**.
- Leaves the data matrix **invariant**
- Solves the label **mapping** problem
- Maintains a **sample-specific embedding** that might have desired properties like specific clusters or trajectories

https://scanpy-tutorials.readthedocs.io/en/latest/integrating-data-using-ingest.html

Question