

Statistical Methods for single cell data analysis 2

Yongjin Park, UBC Path + Stat, BC Cancer

13 March 2024

Unsupervised Learning in single-cell analysis

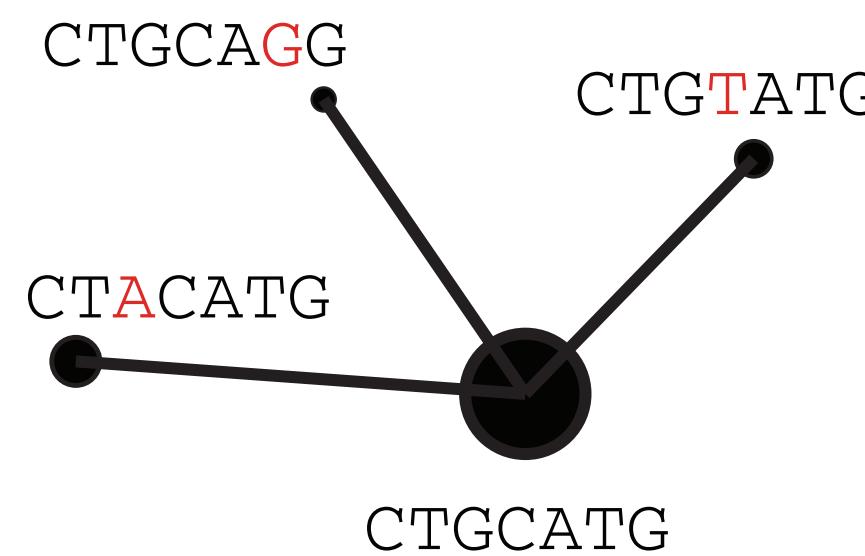
- **Review of PCA in single-cell data analysis**
 - What have we learned?
 - How can independent factors decompose variability?
- **Non-negative matrix factorization**
 - fastTopics & rliger for scalable inference
 - Post-hoc enrichment to relate factors to cell types
- **Can we identify cell type-specific disease genes?**
 - Pseudo-bulk-based DE approach

Source code available:

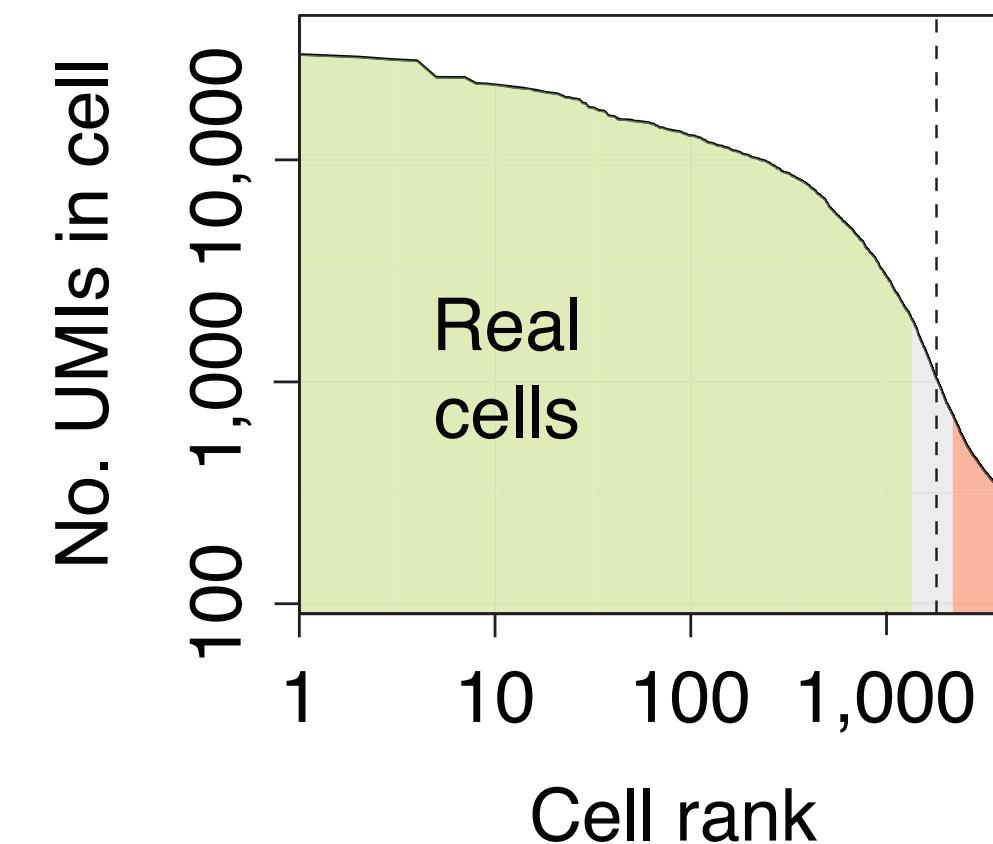
<https://github.com/stat540-UBC/lectures>

Overview of single-cell data analysis

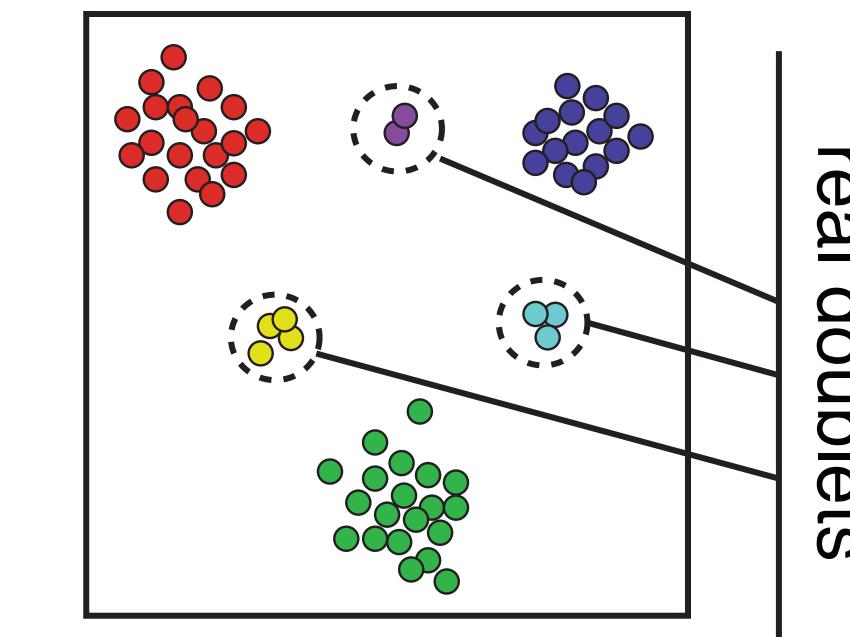
Alignment and molecular counting



Cell filtering and quality control



Doublet scoring

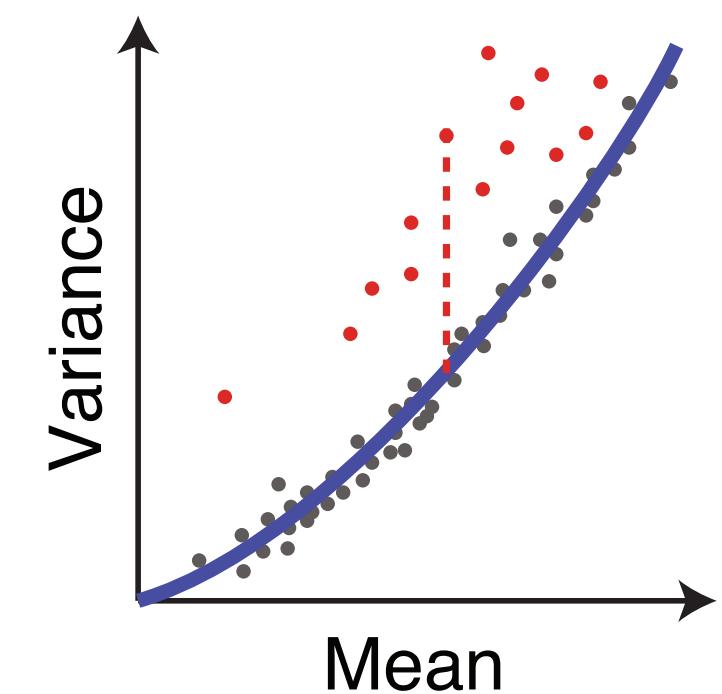


Cell size estimation

Cells	c_1	c_2	c_3
Gene ₁	2	4	20
Gene ₂	1	2	10
Gene ₃	3	6	30

Cell depth: 6 12 60

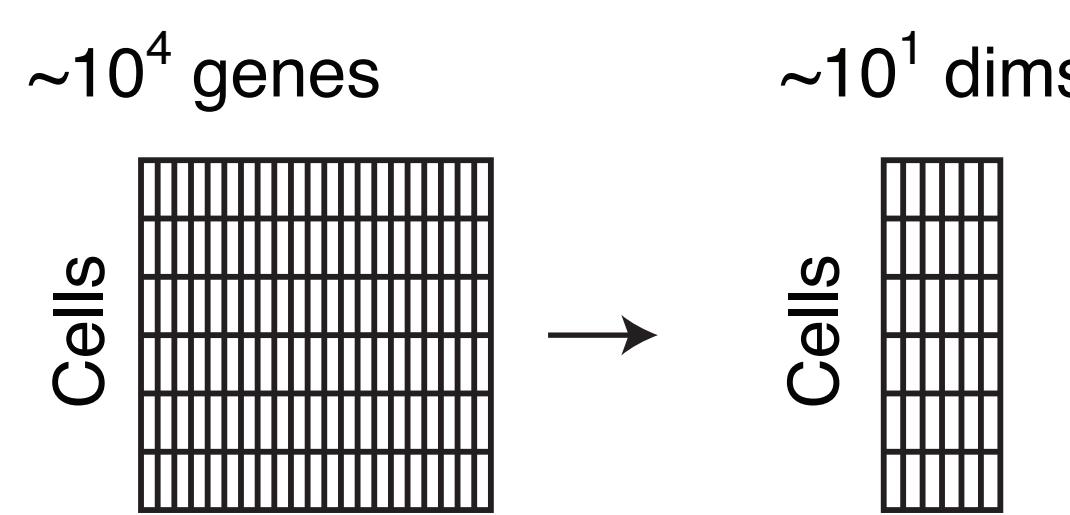
Gene variance analysis



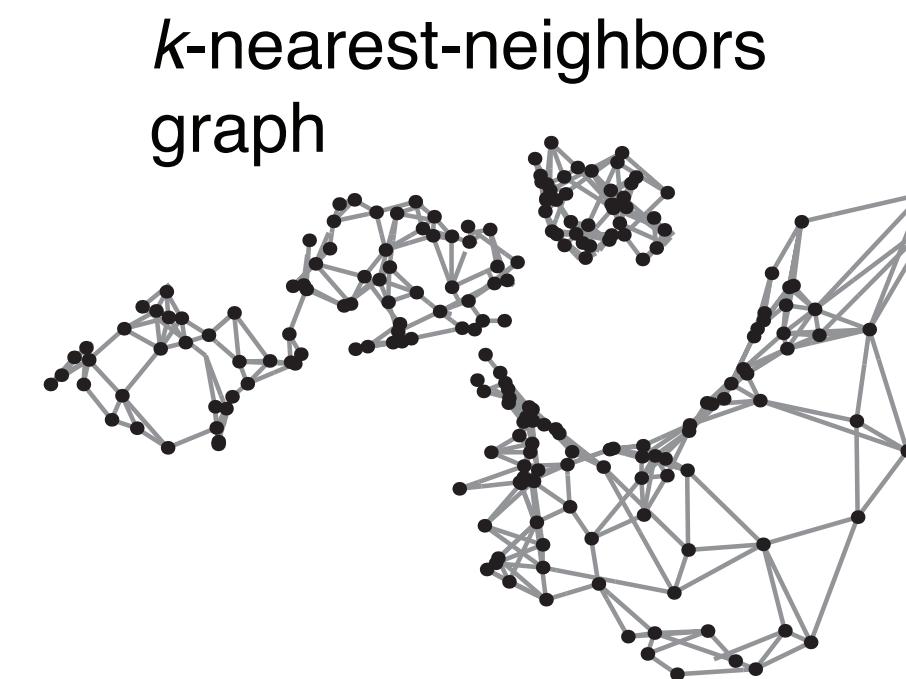
Karchenko, *Nature Methods* (2021)

Overview of single-cell data analysis cont'd

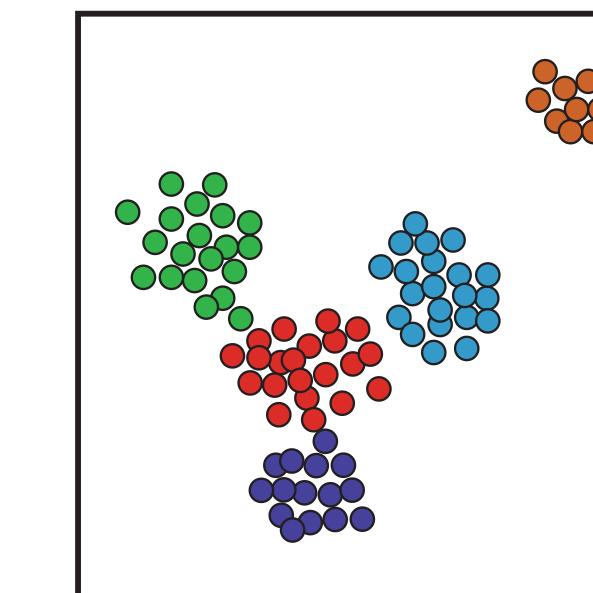
Reduction to a medium-dimensional space



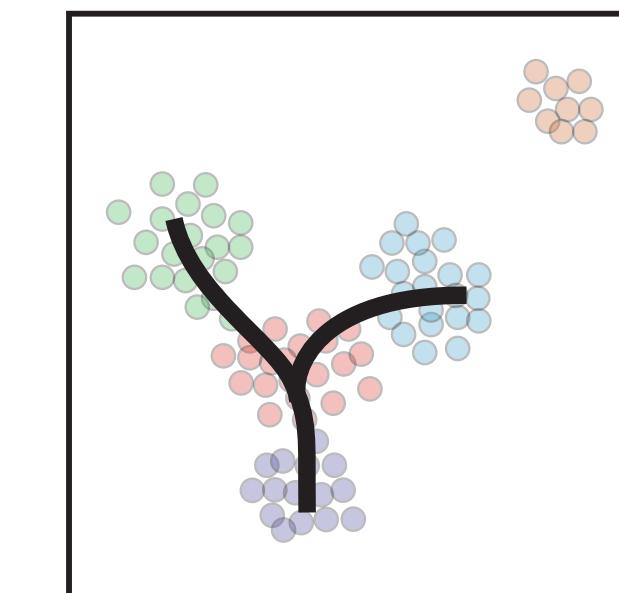
Manifold representation



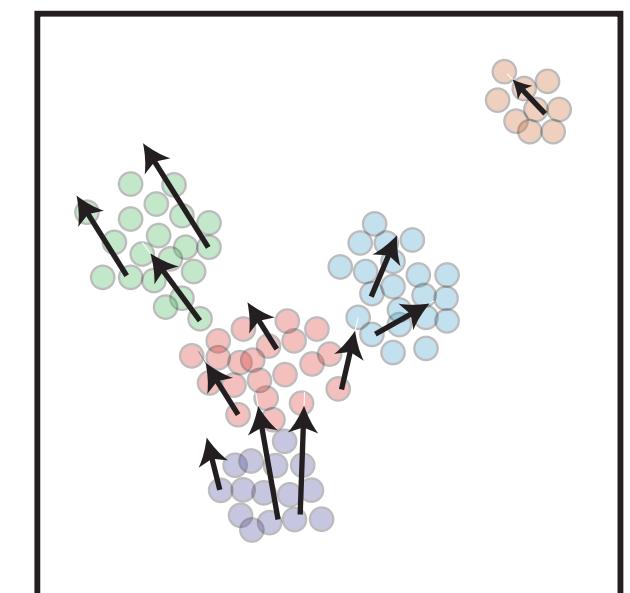
Clustering and differential expression



Trajectories



Velocity estimation



Karchenko, *Nature Methods* (2021)

Today's lecture

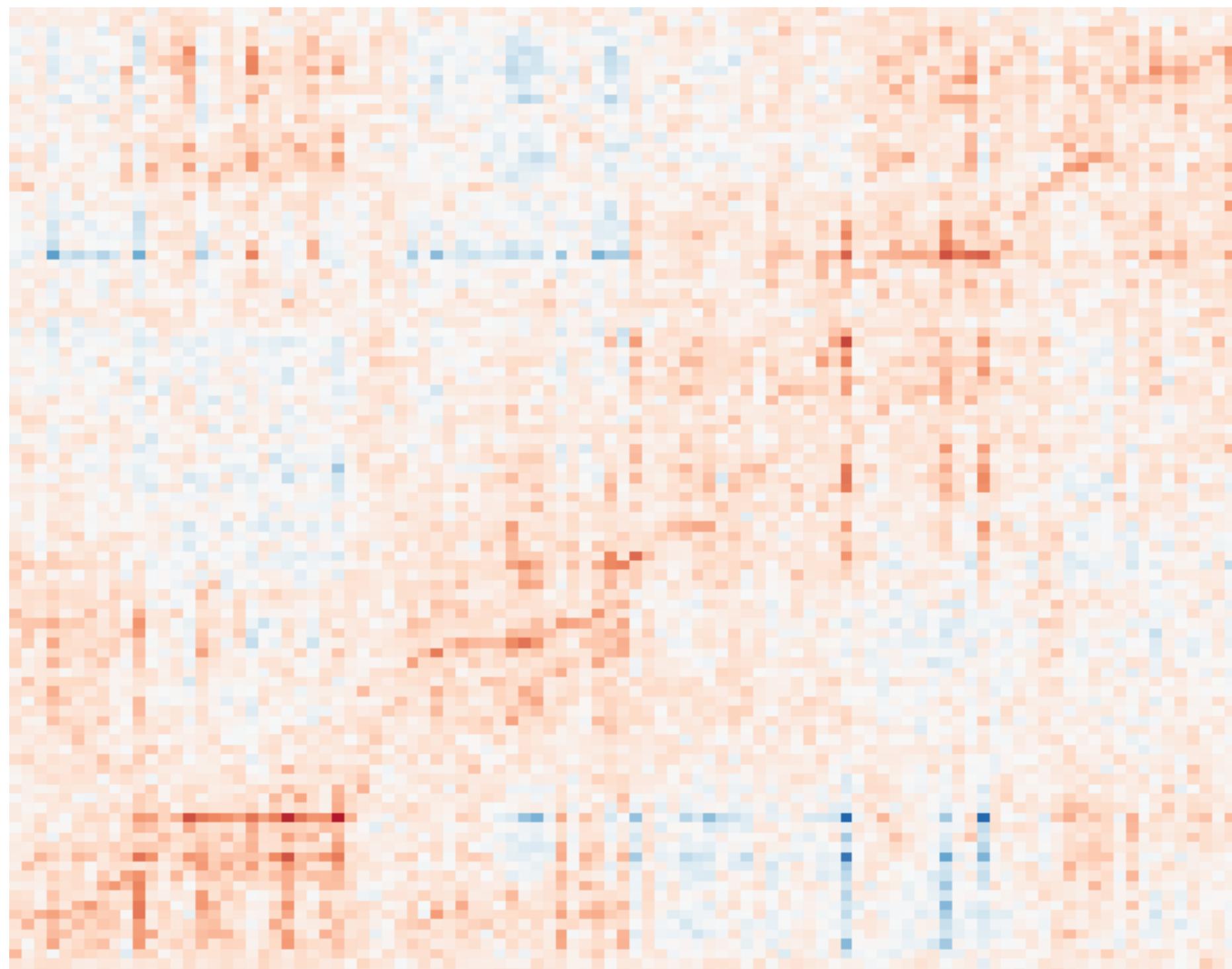
- 1 (review) Principal Component Analysis
- 2 Non-negative matrix factorization
- 3 Differential gene expression analysis

The goal is to find hidden patterns

A data matrix was generated by the following:

$$\mathbf{x}_i = \mathbf{u}_i V^\top + \boldsymbol{\epsilon}_i$$

for all i with $\boldsymbol{\epsilon}_i \sim \mathcal{N}(\mathbf{0}, I)$



Can we reverse engineer the U and V matrices from the data X ?

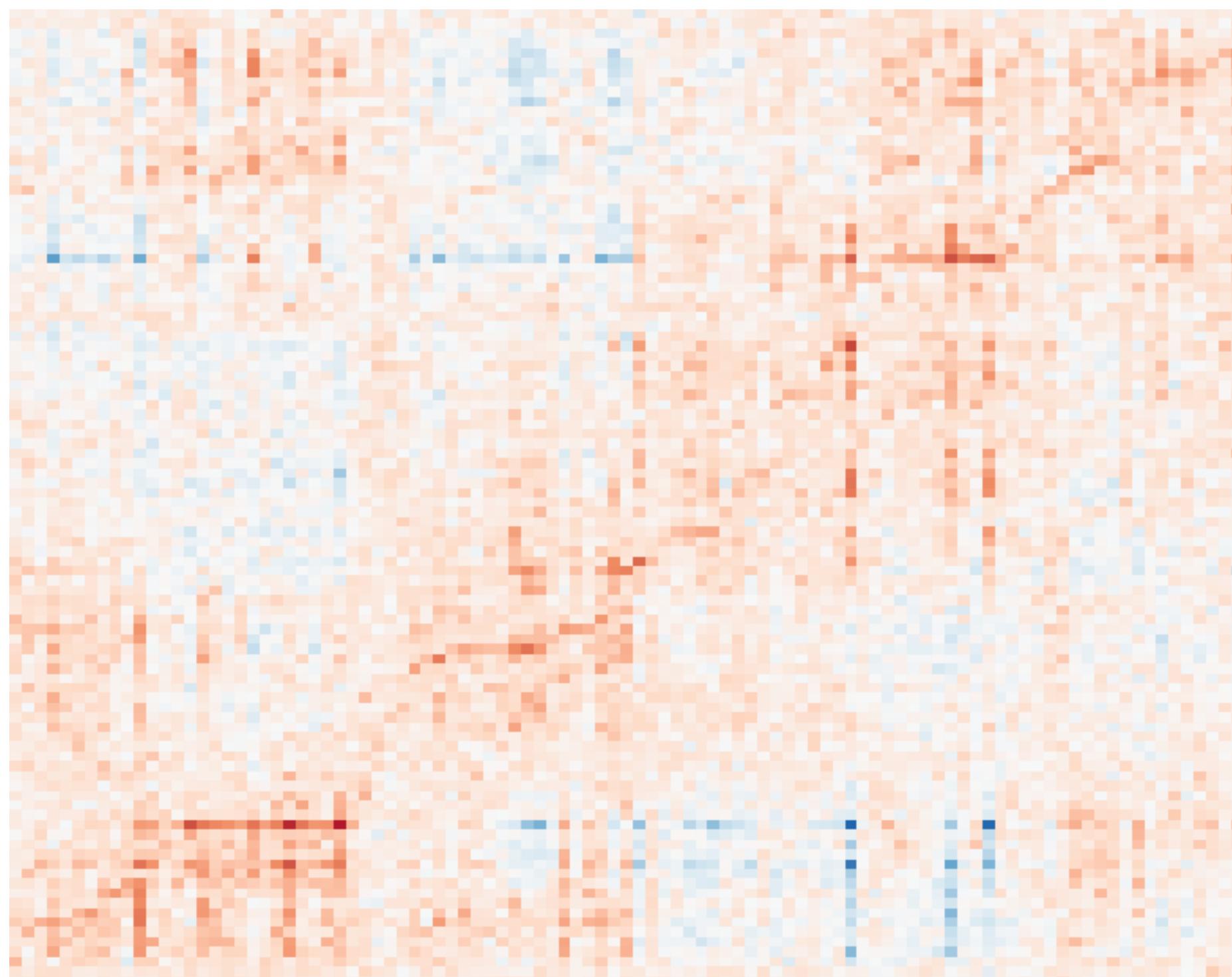
$$X \rightarrow UV^\top$$

The goal is to find hidden patterns

A data matrix was generated by the following:

$$\mathbf{x}_i = \mathbf{u}_i V^\top + \epsilon_i$$

for all i with $\epsilon_i \sim \mathcal{N}(\mathbf{0}, I)$



Can we reverse engineer the U and V matrices from the data X ?

$$X \rightarrow UV^\top$$

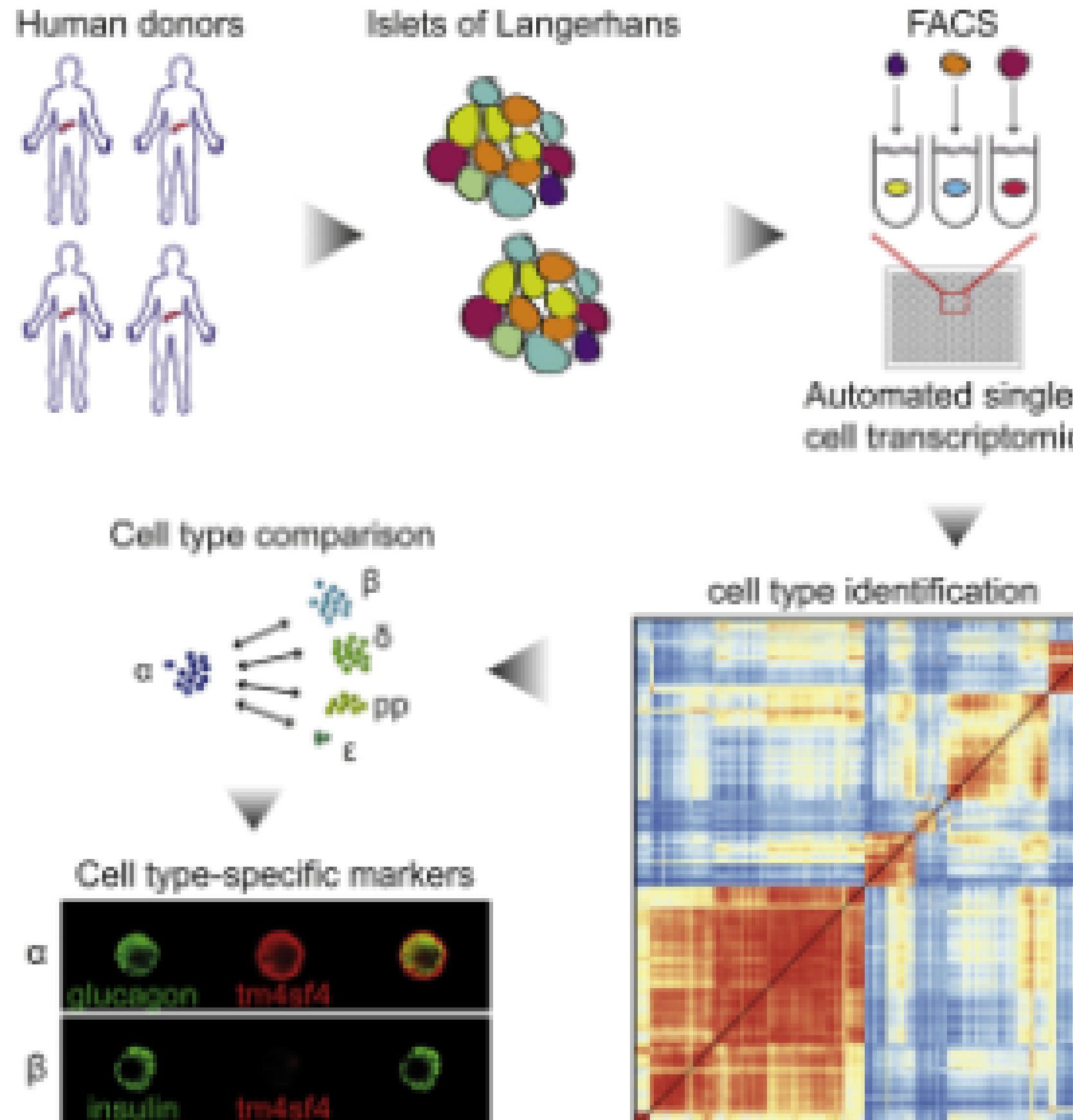
FEATURES



FEATURE LOADING

X

Working Example: Pancreatic cells (Muraro et al. 2016)



It's a sparse matrix...

```
x[1:10, 1:5]
```

```
## 10 x 5 sparse Matrix of class "dgCMatrix"
##          D28-1_1 D28-1_2 D28-1_3 D28-1_4 D28-1_5
## A1BG-AS1      .      .      .      .
## A1BG          .      .      1      1      .
## A1CF          6      .      2      6      .
## A2M-AS1        .      .      .
## A2ML1          .      .      .
## A2M           .      5      .
## A4GALT         .      1      .
## A4GNT          .      .
## AAAS           1      .
## AACSP1          .      .      .      .
```

- rows: 19,049 genes
- columns: 3,072 cells

Why do we need to reduce dimensionality?

Why do we need to reduce dimensionality?

- ① High dimensionality, robust downstream analysis

Why do we need to reduce dimensionality?

- ① High dimensionality, robust downstream analysis
- ② Pattern recognition: meta genes, meta cells, etc.

Why do we need to reduce dimensionality?

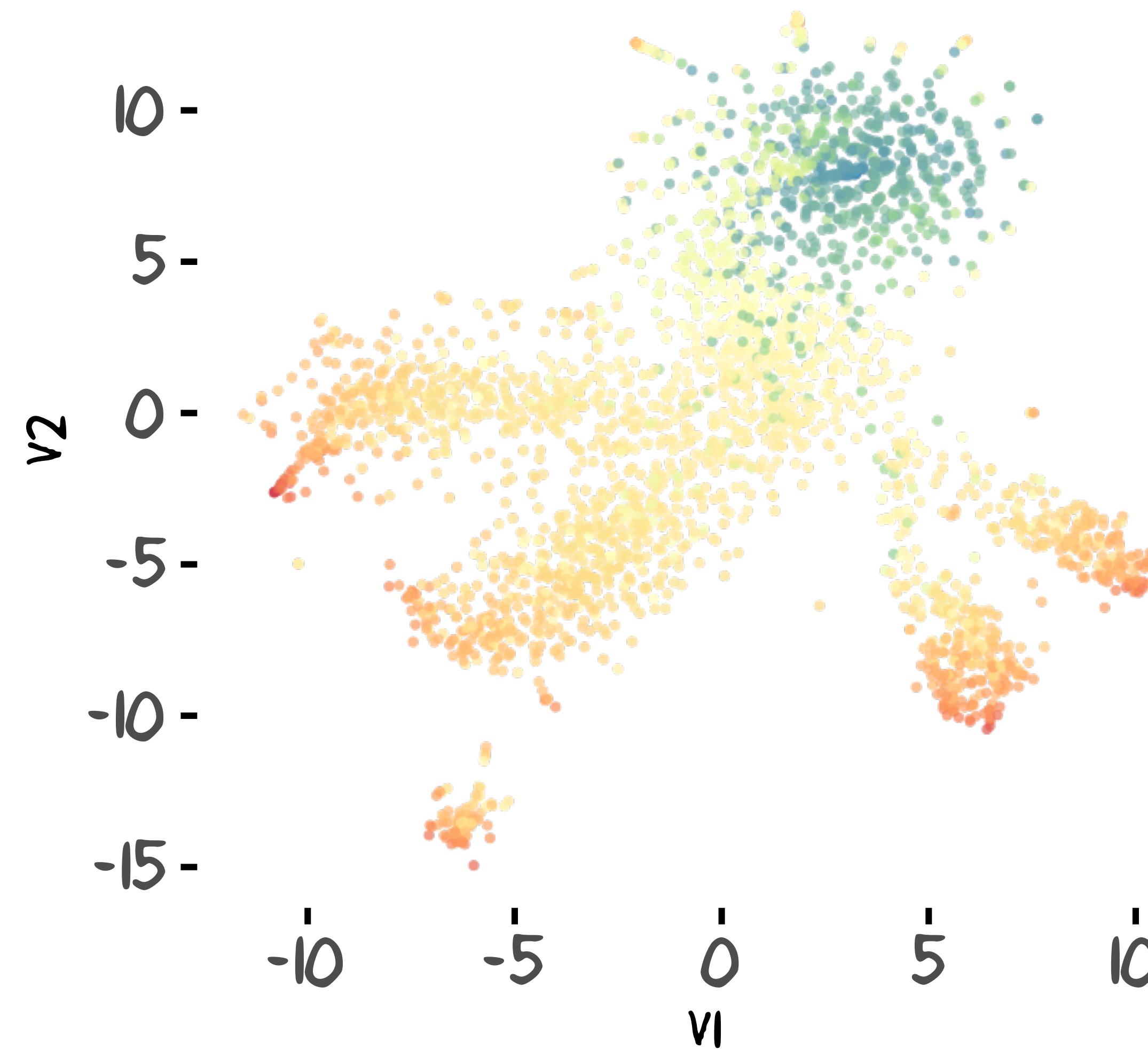
- ① High dimensionality, robust downstream analysis
- ② Pattern recognition: meta genes, meta cells, etc.
- ③ Reduce unwanted variation of outliers

Why do we need to reduce dimensionality?

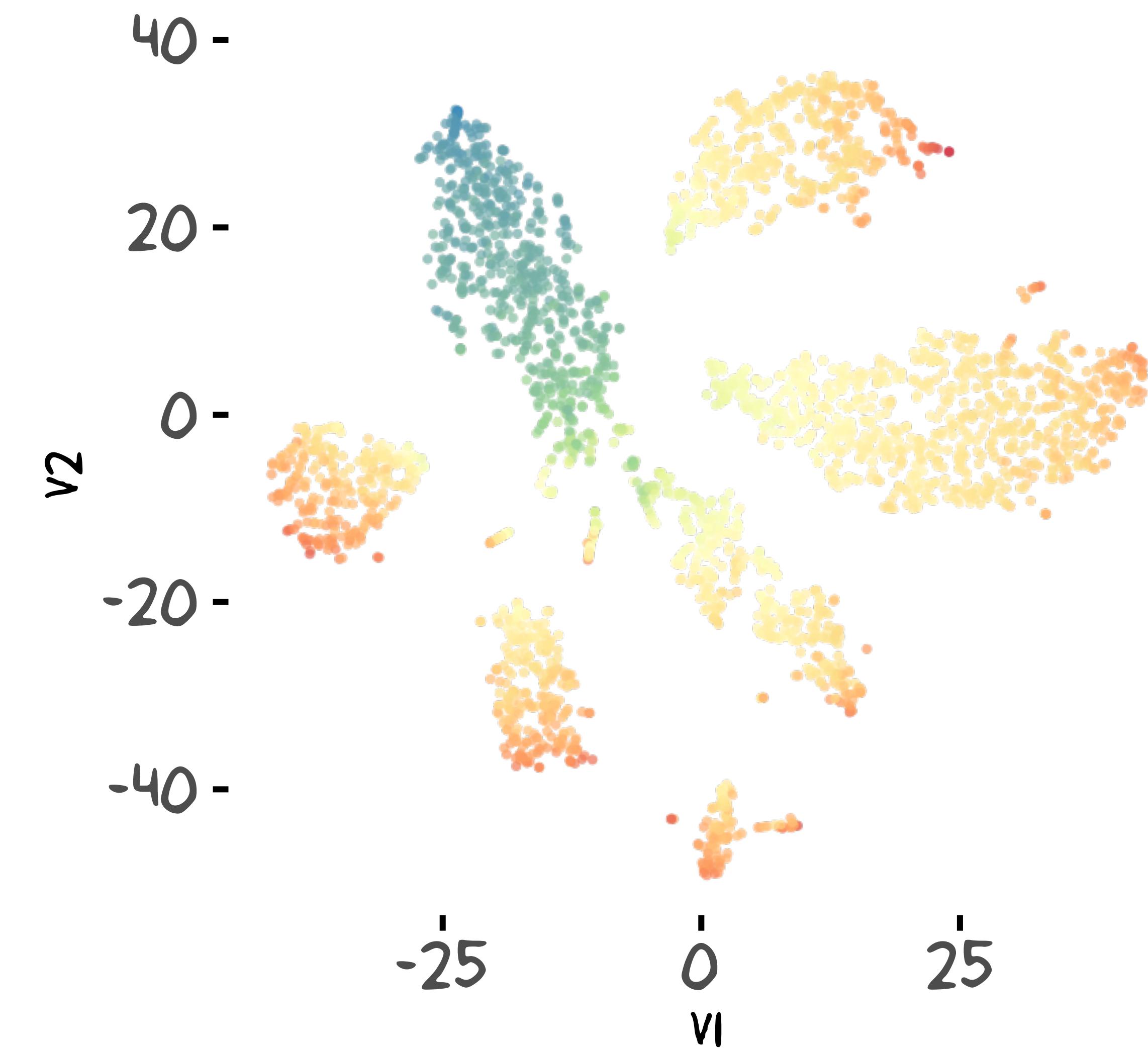
- ① High dimensionality, robust downstream analysis
- ② Pattern recognition: meta genes, meta cells, etc.
- ③ Reduce unwanted variation of outliers
- ④ Aggregate weak contributions of many features

Visualization with or without dimensionality reduction

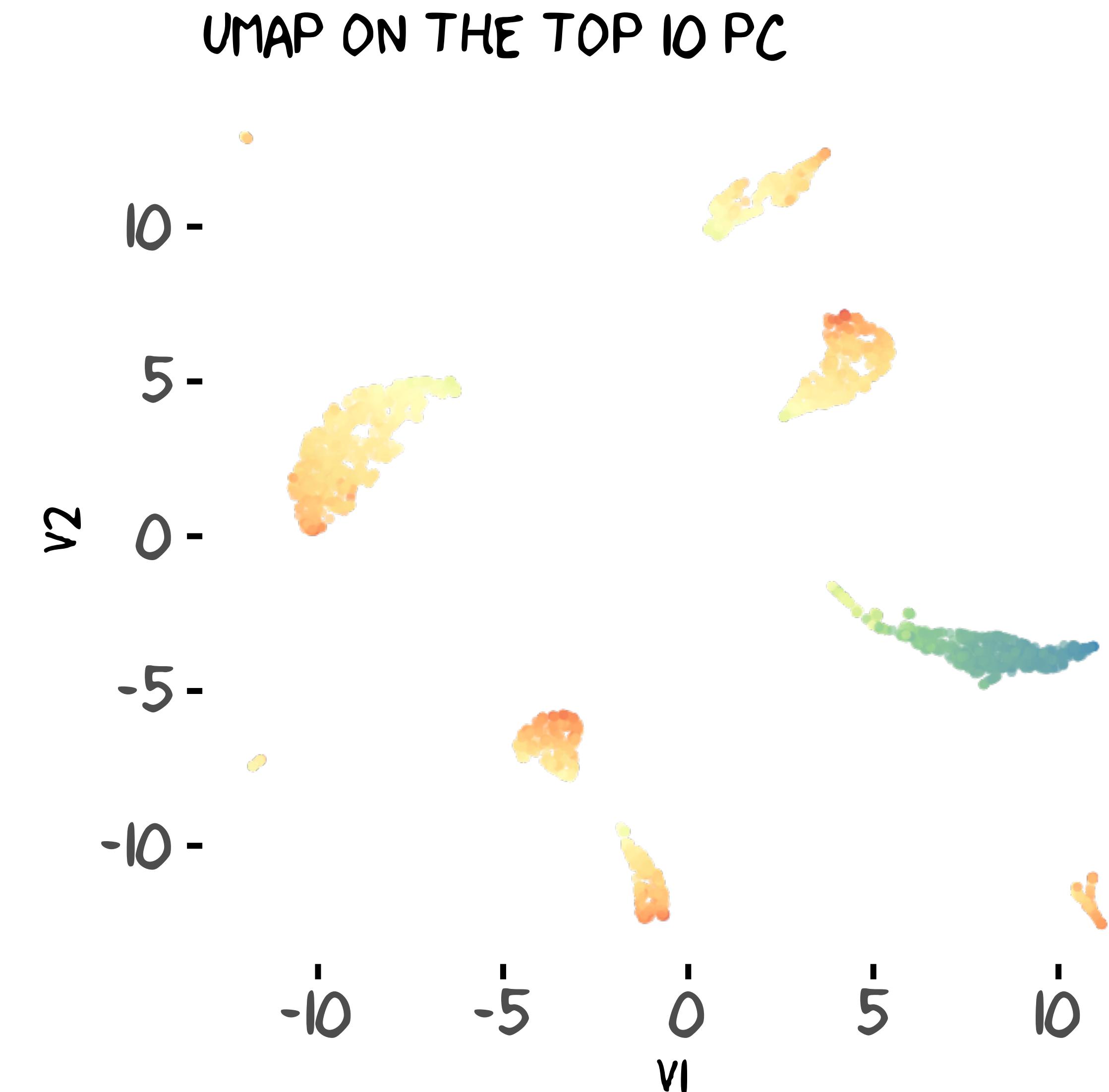
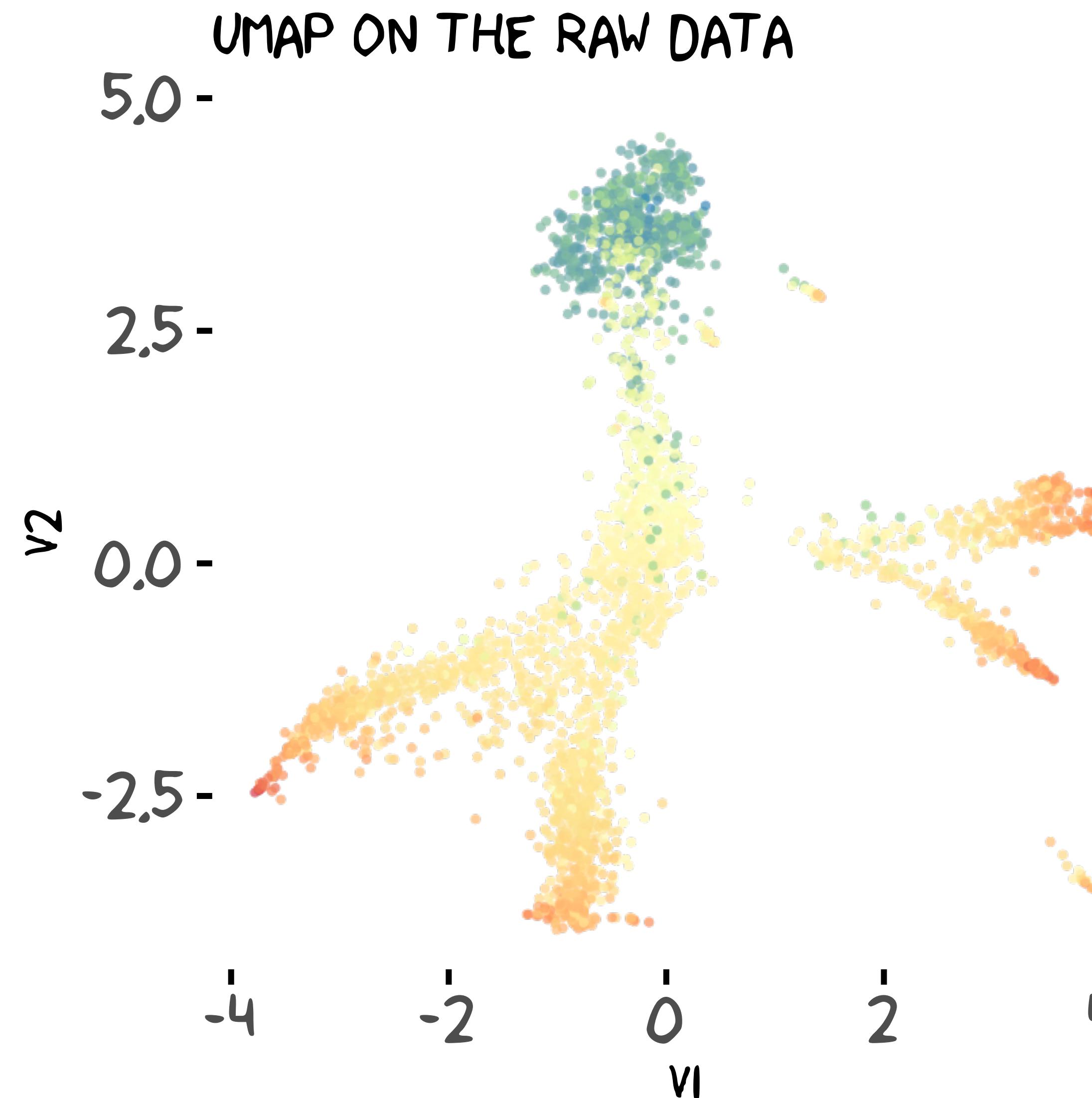
TSNE ON THE RAW DATA



TSNE ON THE TOP 10 PC

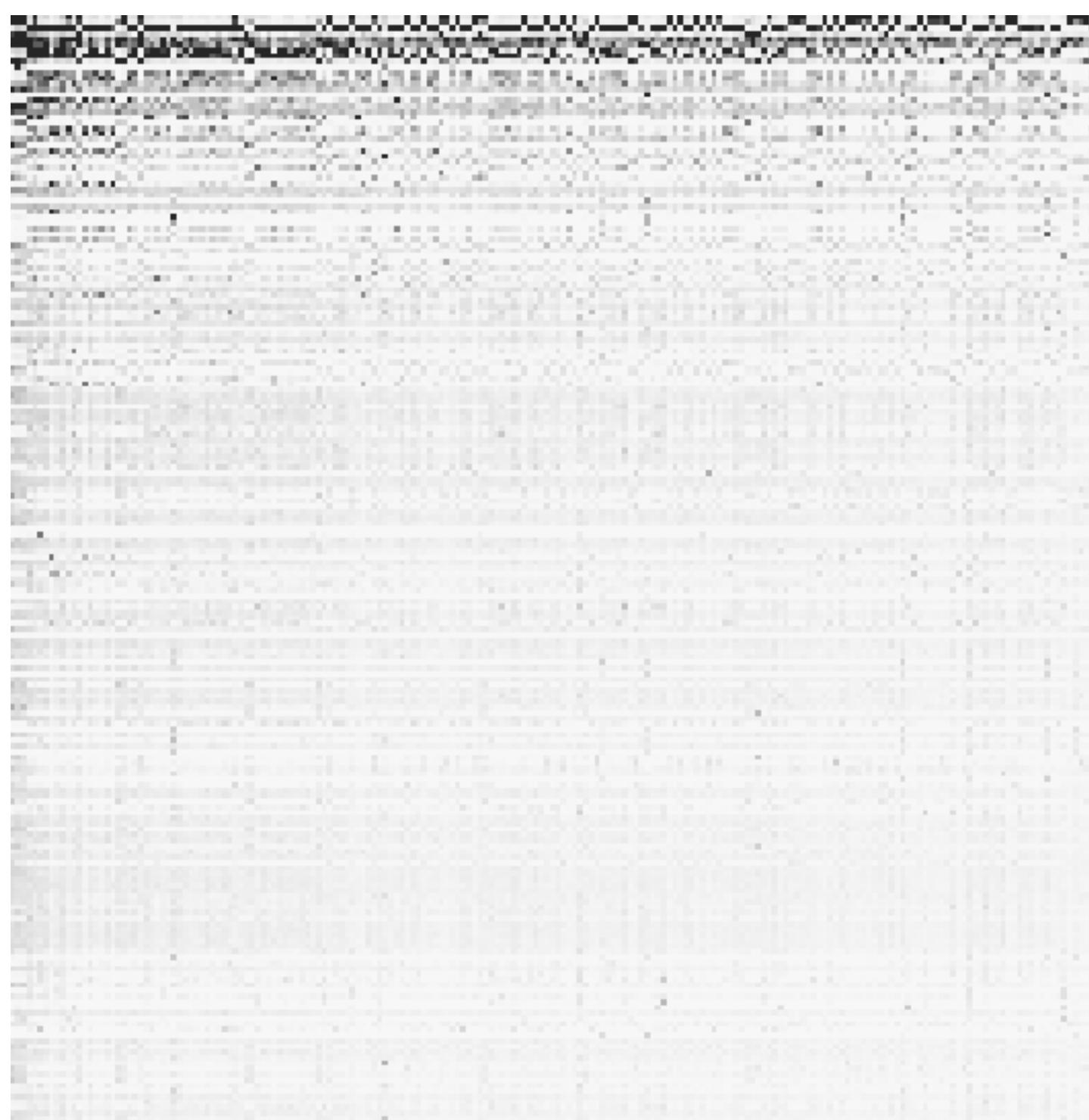


Visualization with or without dimensionality reduction



Let's just use top 200 most variable genes and cells

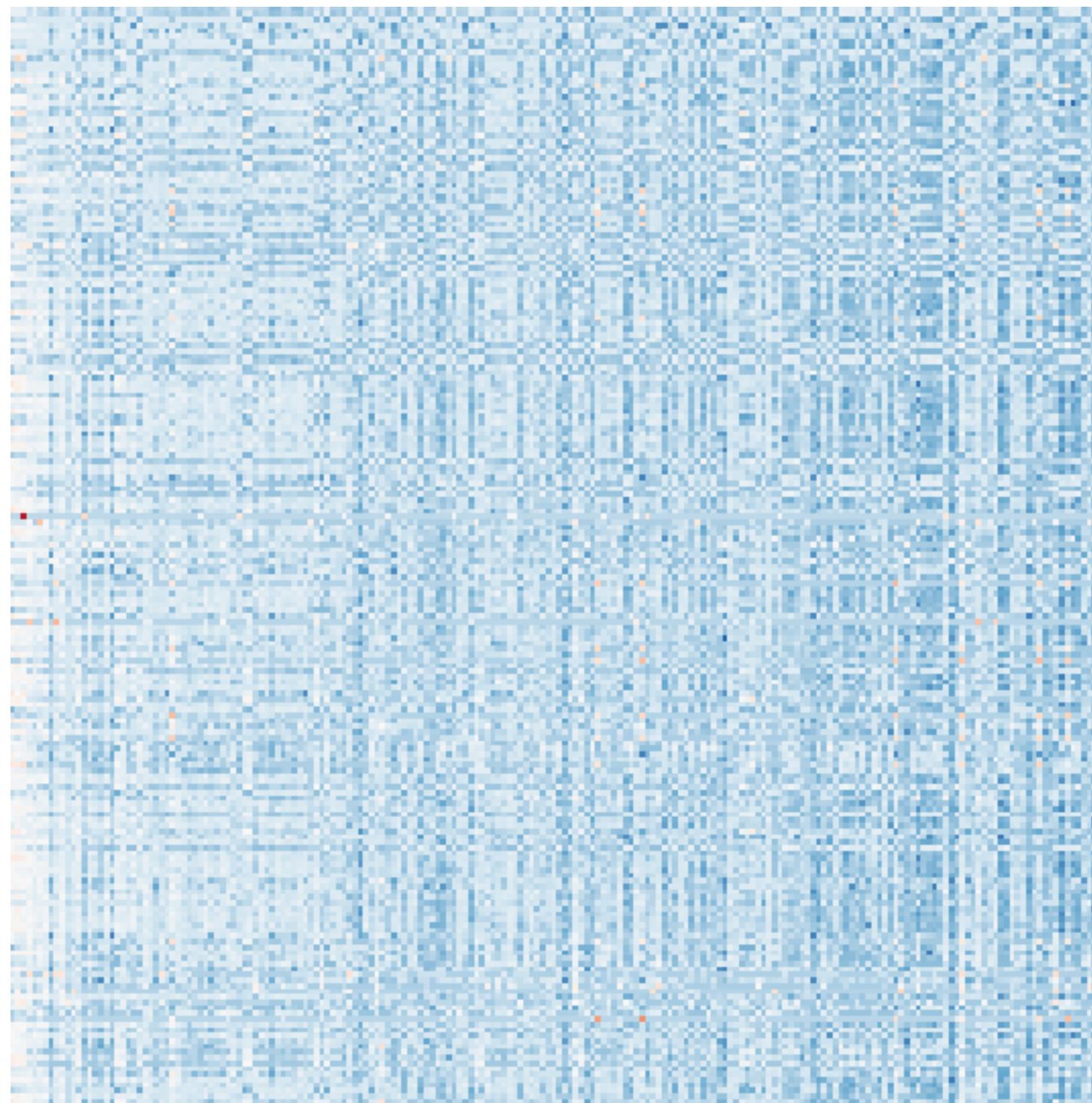
Top 200 genes and cells



- We will call such a high-dimensional matrix X ($m \times n$)
- X has $m=19,049$ rows (transcripts/genes/features)
- X has $n=3,072$ columns (cells/#data points)
- The rows were log-transformed and scaled by `scale()` for visualization
- Each cell is a 19,049-dimensional vector!
- Each gene is a 3,072-dimensional vector...

Let's just use top 200 most variable genes and cells

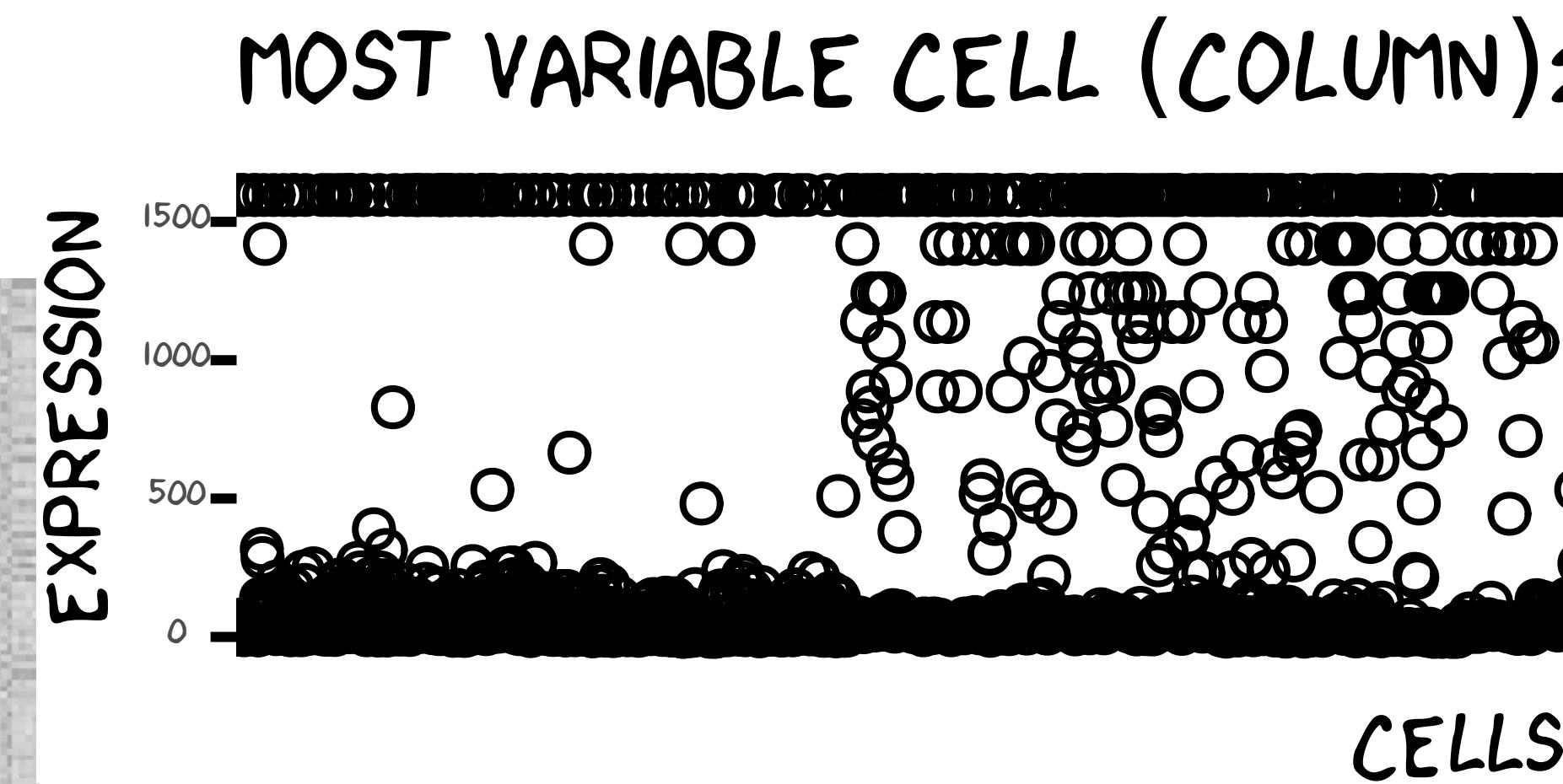
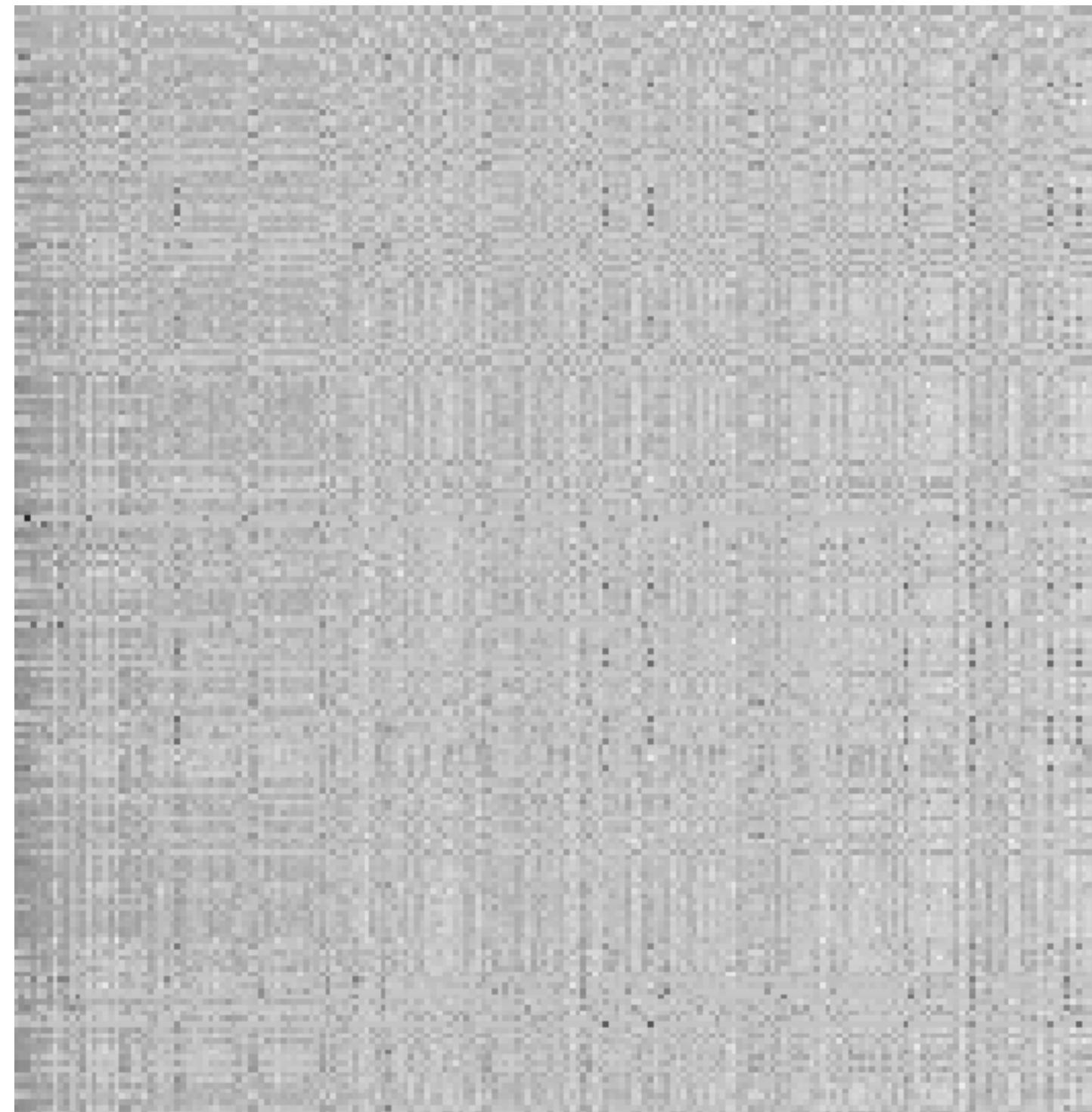
Top 200 genes and cells



- We will call such a high-dimensional matrix X ($m \times n$)
- X has $m=19,049$ rows (transcripts/genes/features)
- X has $n=3,072$ columns (cells/#data points)
- The rows were log-transformed and scaled by `scale()` for visualization
- Each cell is a 19,049-dimensional vector!
- Each gene is a 3,072-dimensional vector...

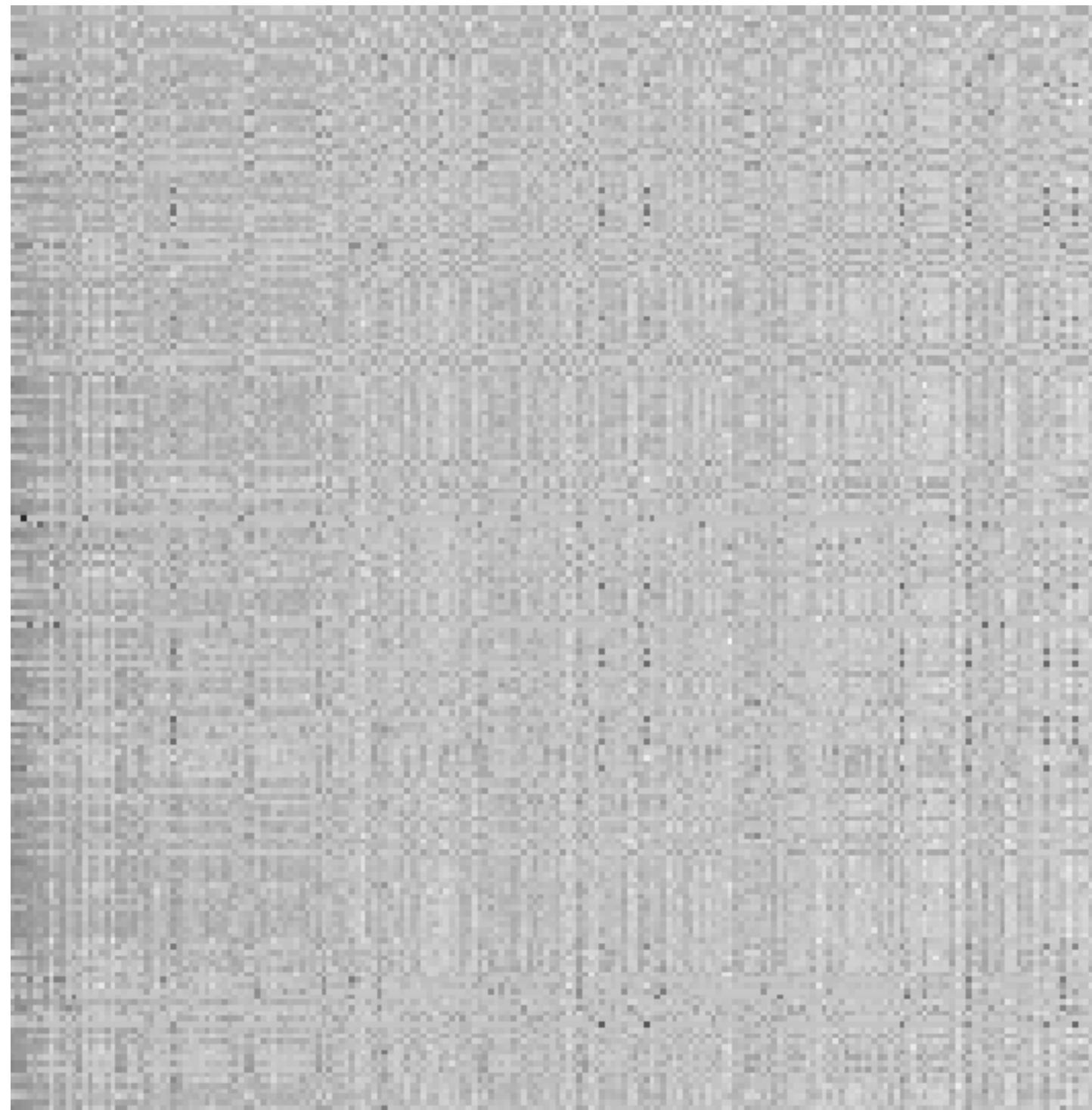
1-dimensional representations/summary of data matrix

Top 200 genes and cells:

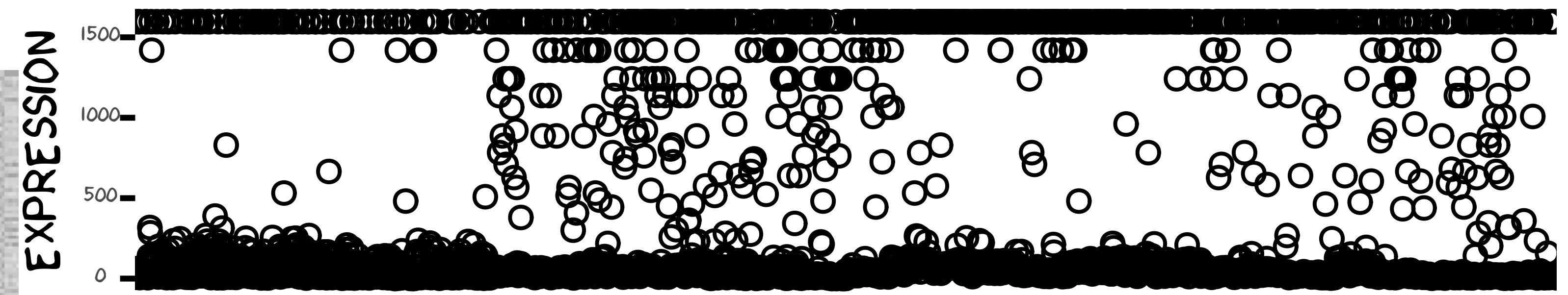


1-dimensional representations/summary of data matrix

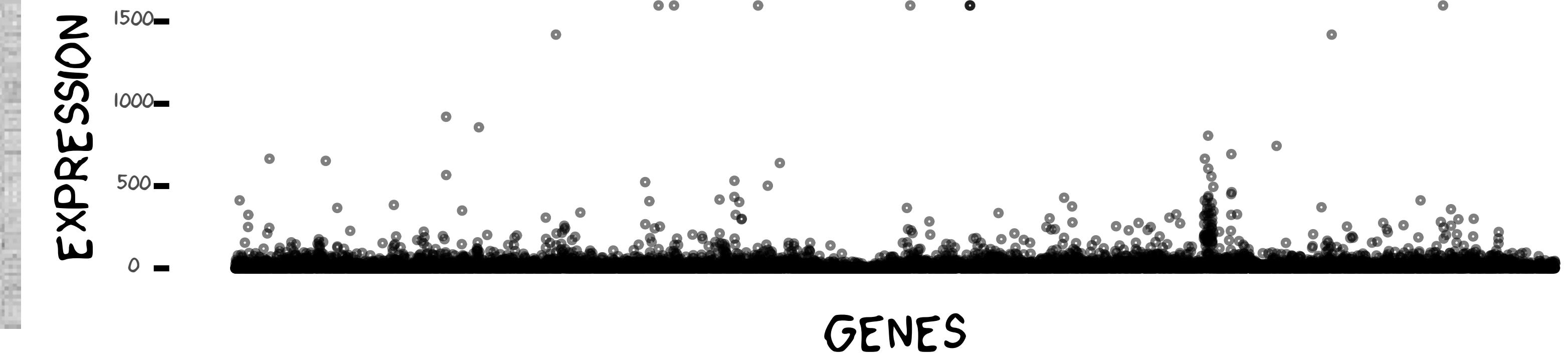
Top 200 genes and cells:



MOST VARIABLE CELL (COLUMN):

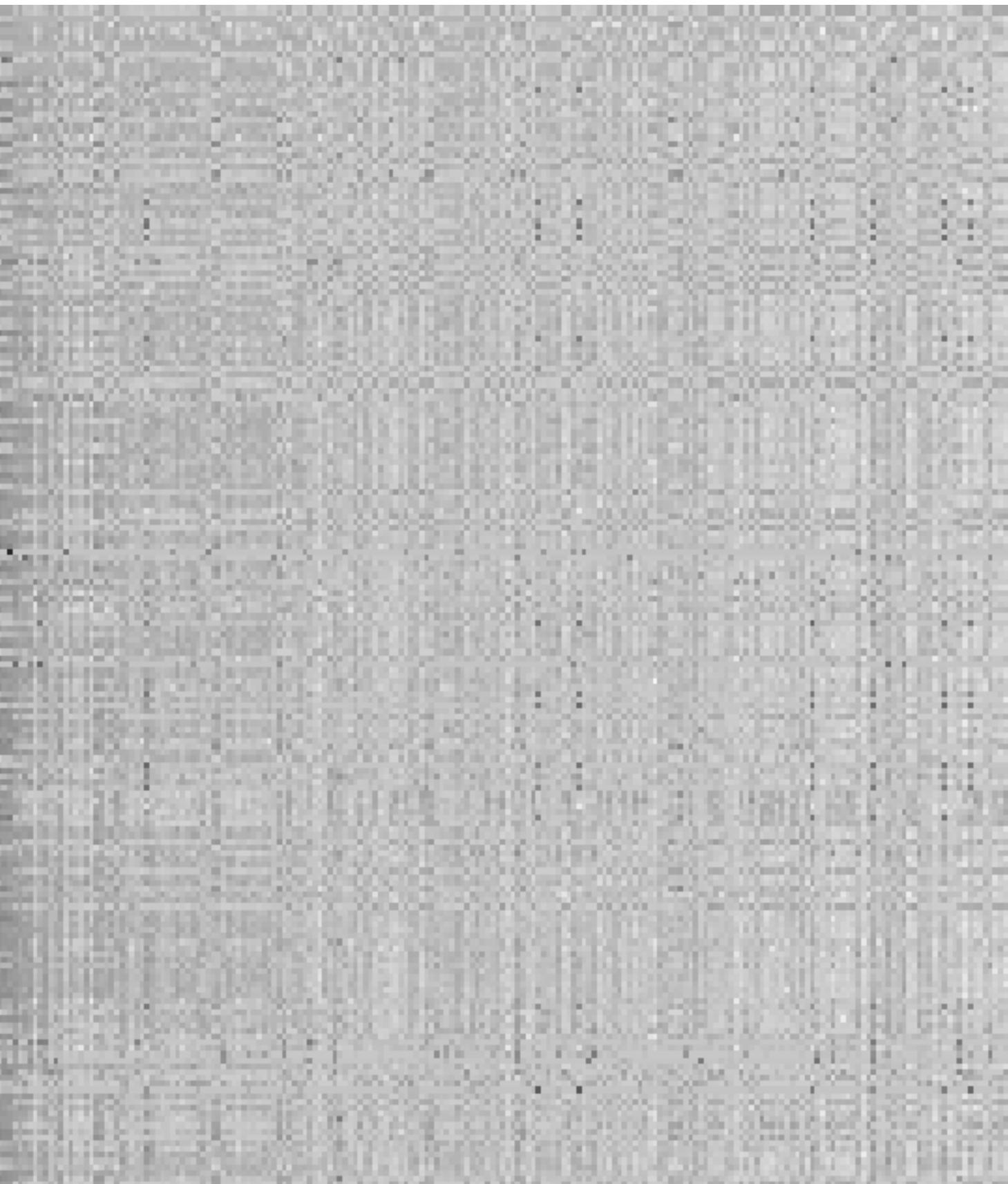


CELLS
MOST VARIABLE GENE (ROW):



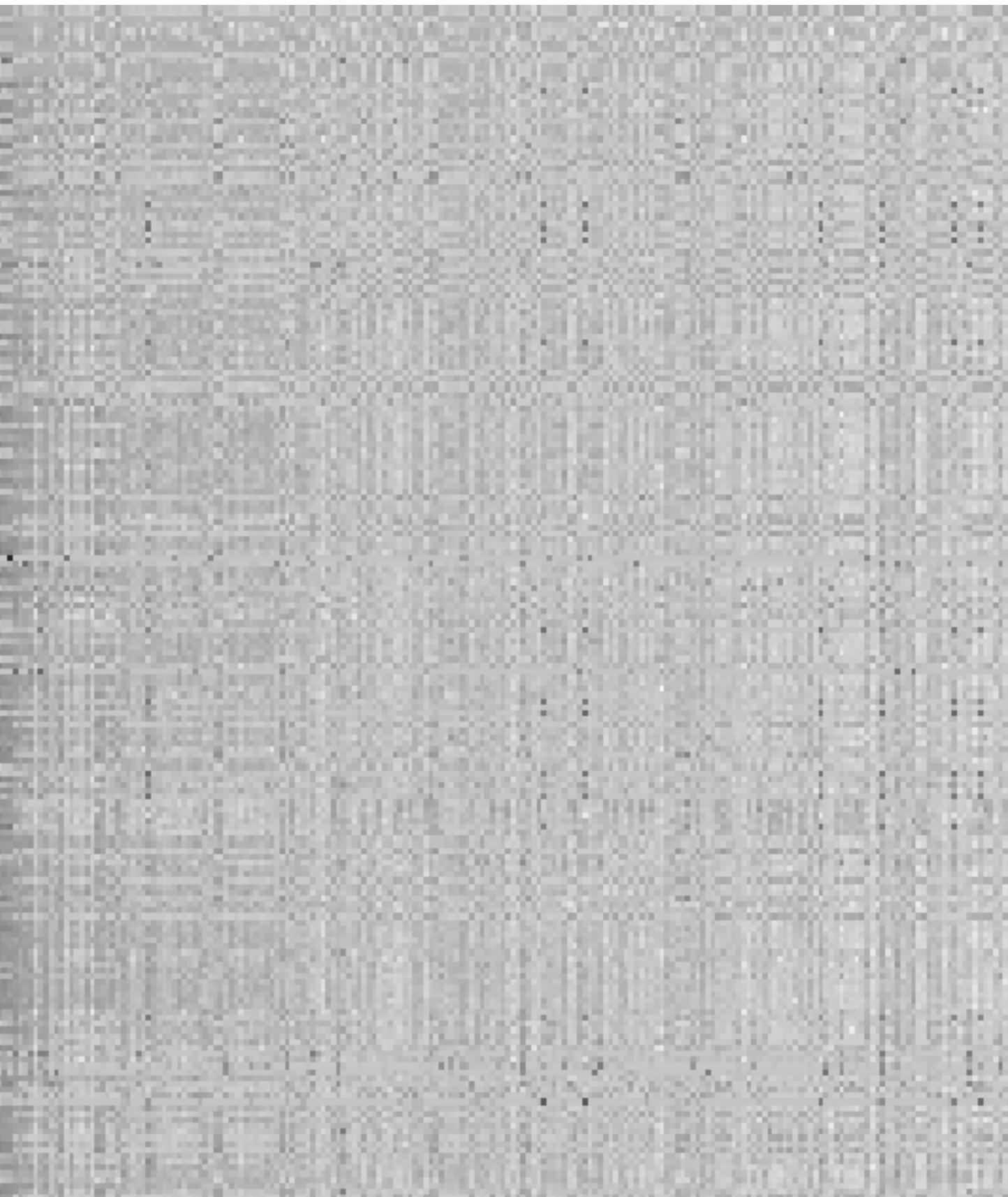
Can you see some common patterns?

200 GENES AND CELLS



Can you see some common patterns?

200 GENES AND CELLS



REARRANGE THEM...



PCA: How do we recover “common” patterns from data?

Principal Component Analysis

(Pearson 1901)

- Projection [of the original data] that minimizes the projection cost between the original and projected
- The cost = mean squared distance

(Hotelling 1933)

- Orthogonal projection of data into a lower-dimensional **[principal]** sub-space,
- such that the total **variation of the projected** is maximized

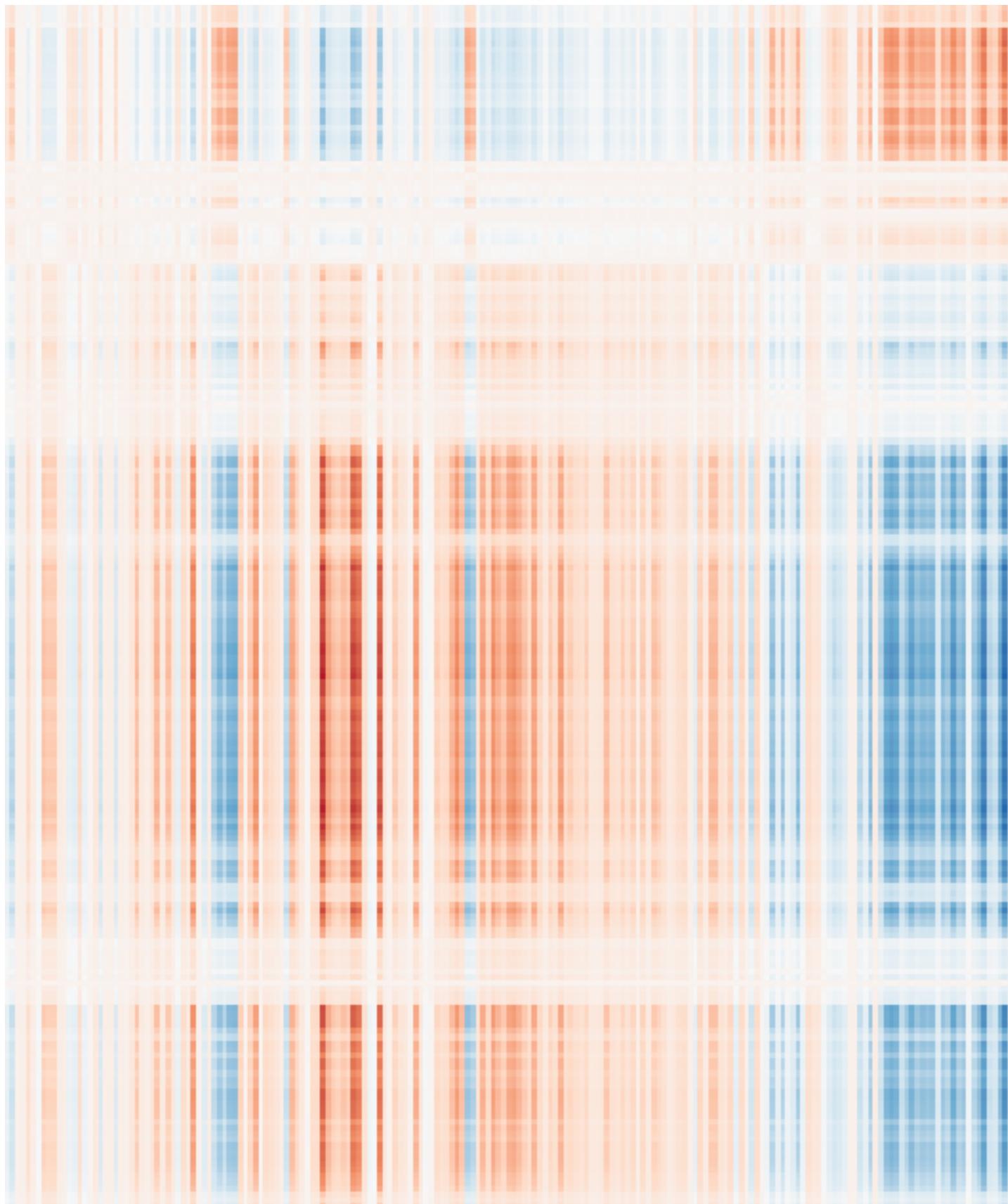
Let's see how much the first eigenvector can explain

TOP 200 GENES AND CELLS



Let's see how much the first eigenvector can explain

EIGEN PROJ.



E-VEC LOADING

=

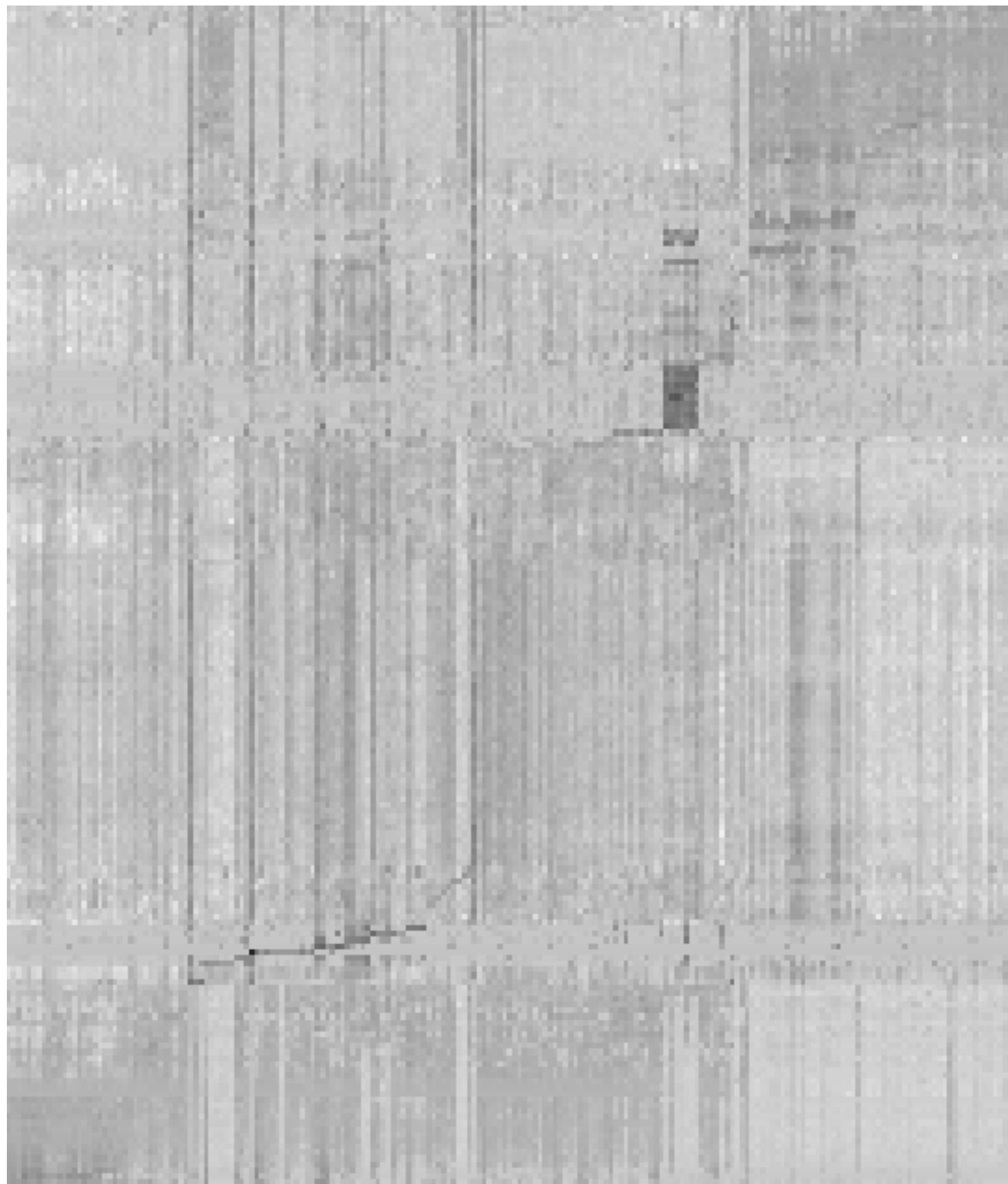


X

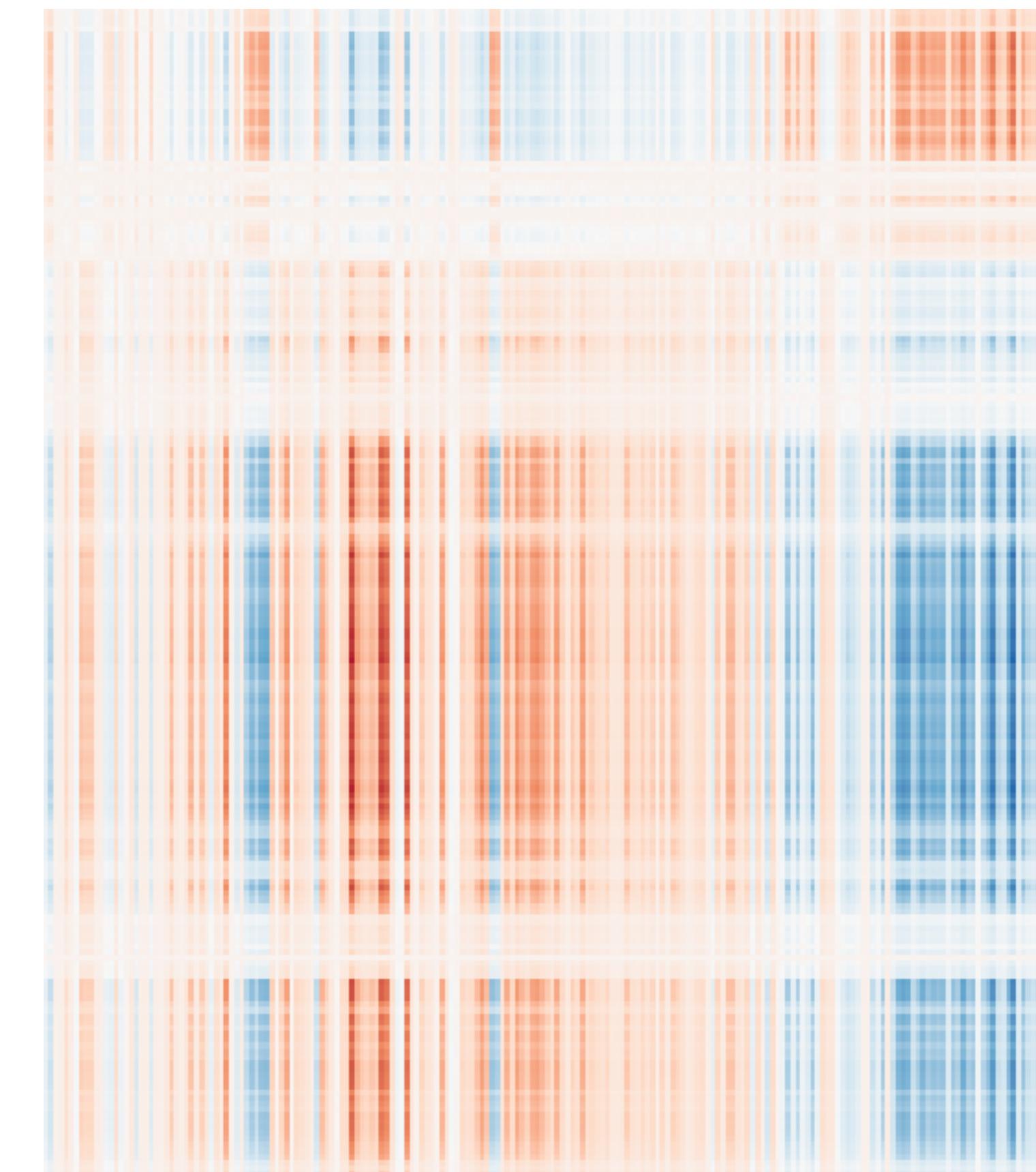


Let's see how much the first eigenvector can explain

TOP 200 GENES AND 200 CELL

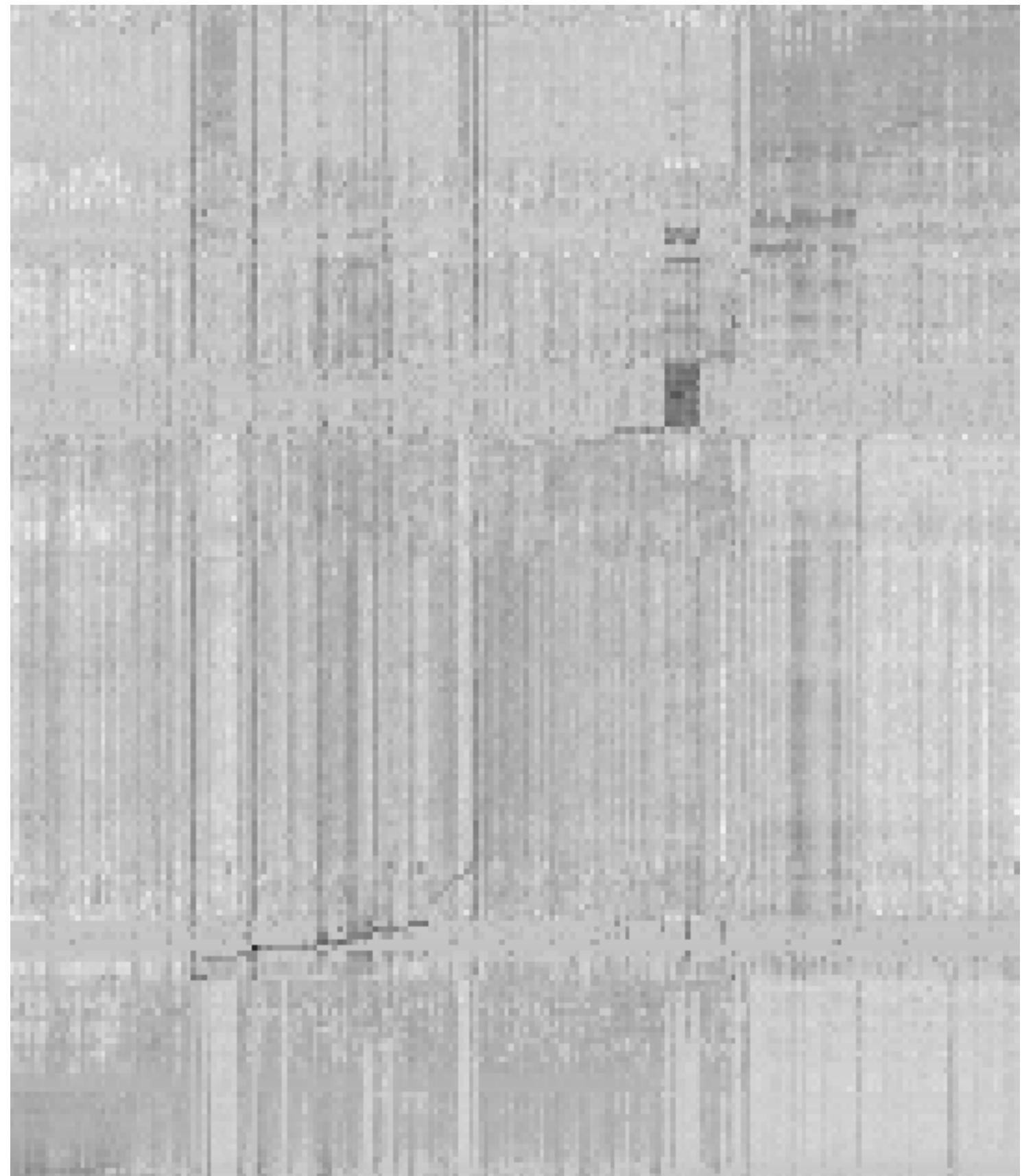


29%



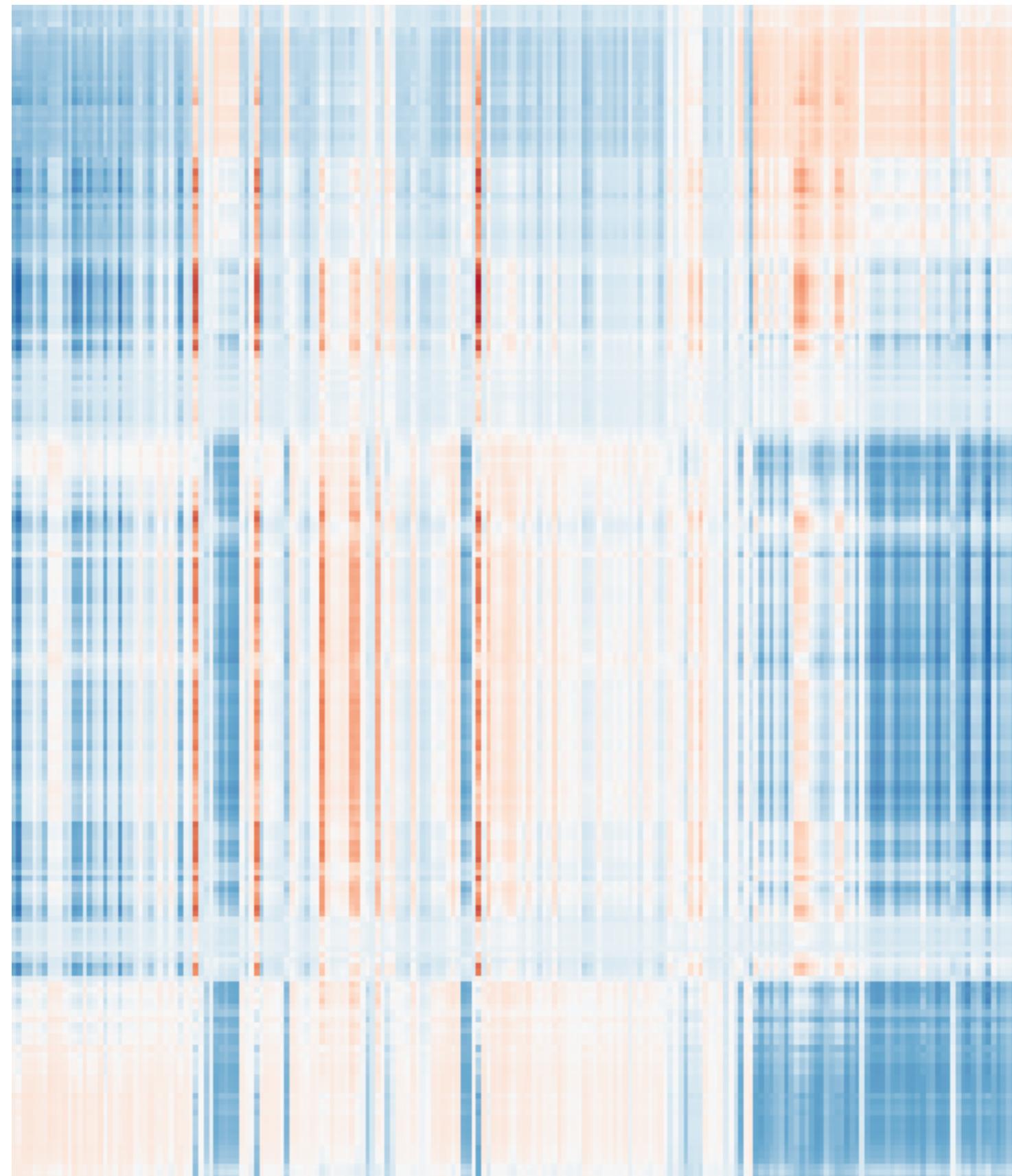
Top two eigen-vectors

TOP 200 GENES AND 200 CELL



Top two eigen-vectors

EIGEN PROJ.



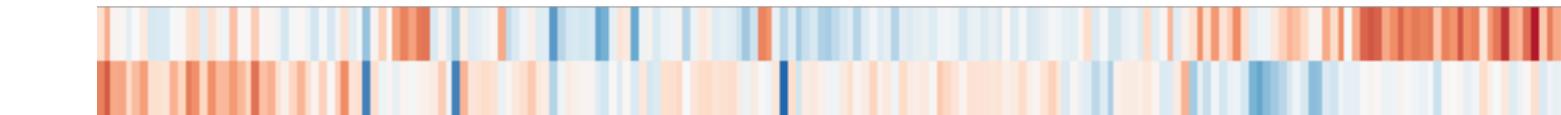
=

E-VEC

X

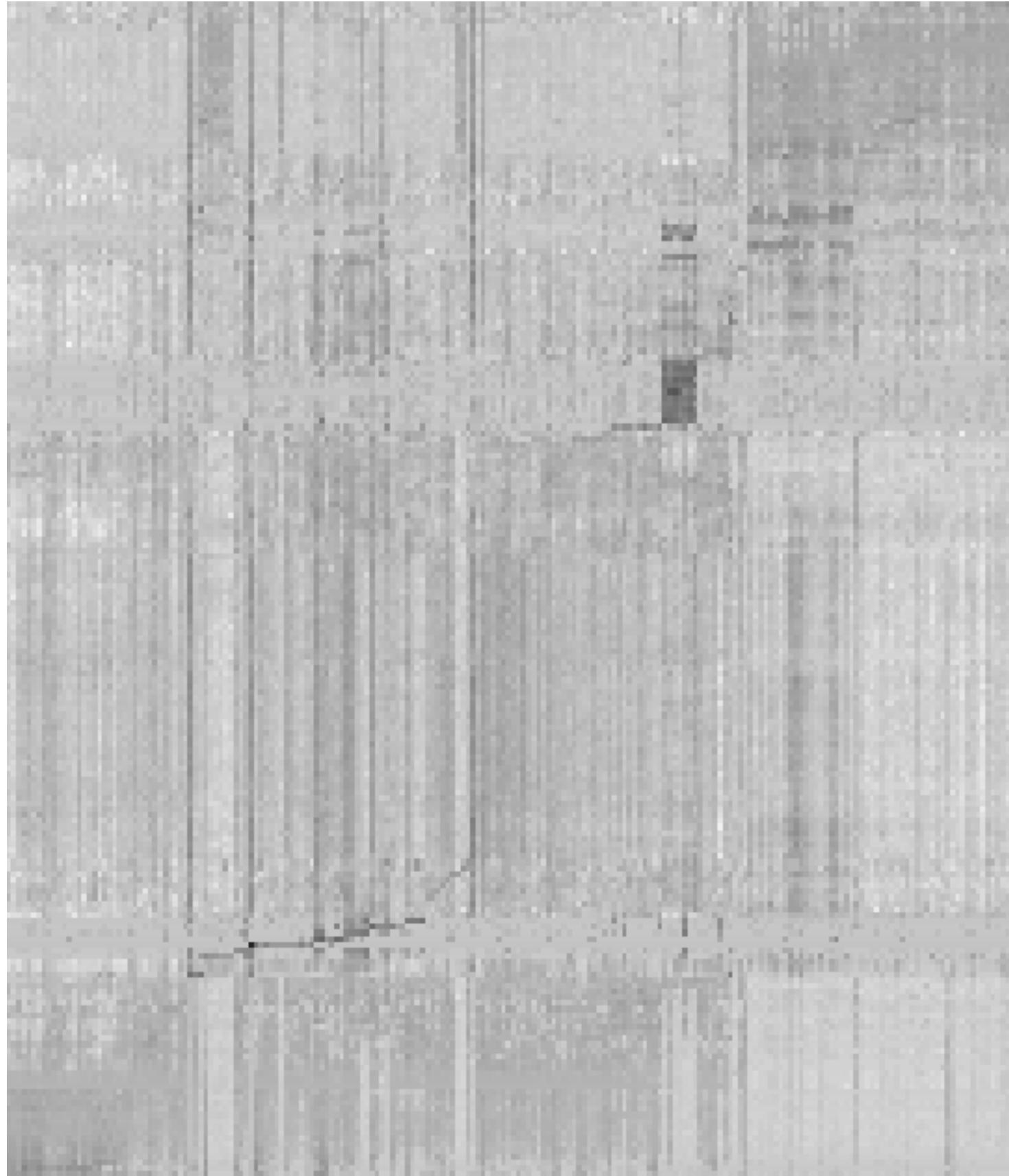


LOADING

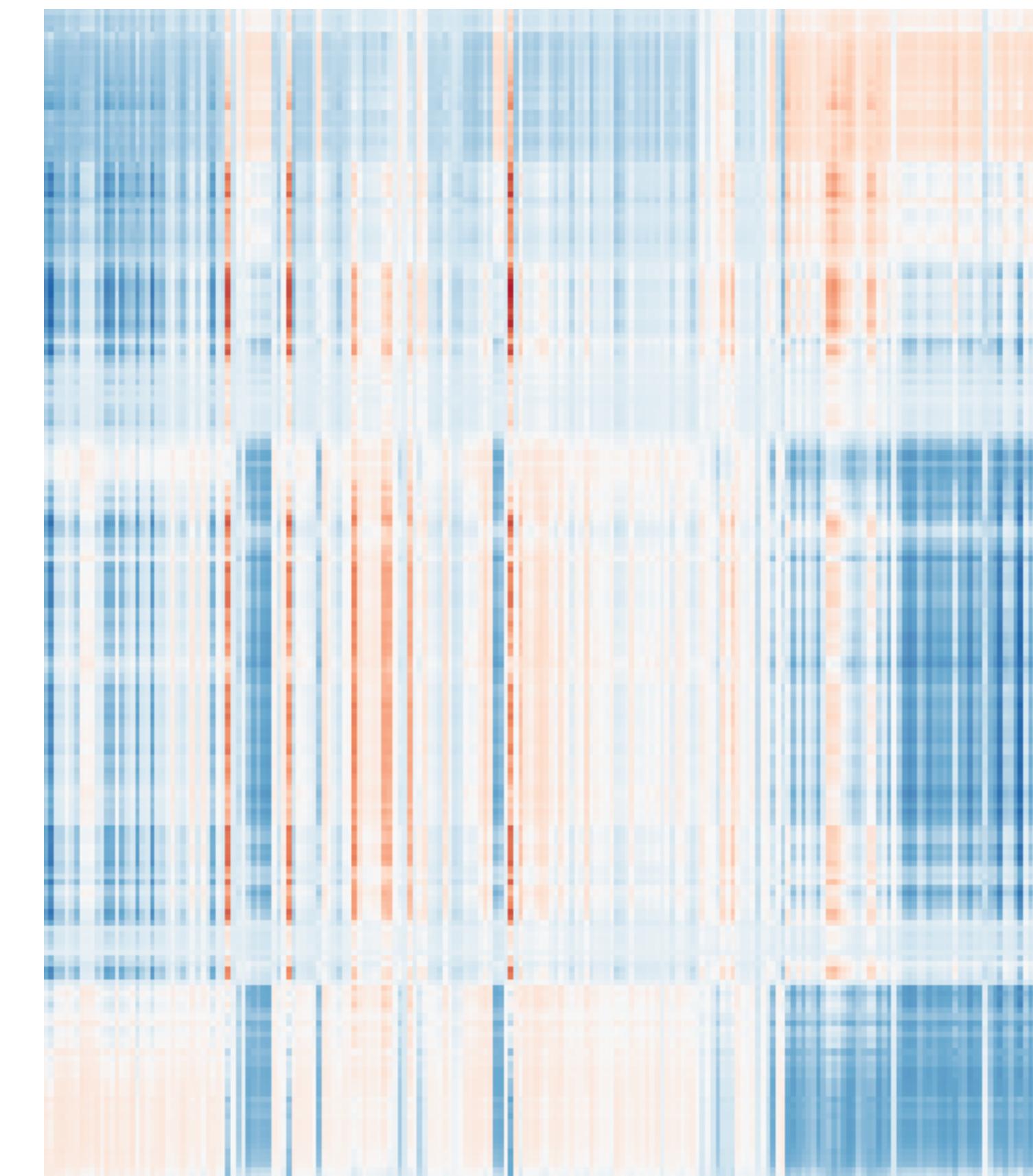


Top two eigen-vectors

TOP 200 GENES AND 200 CELL

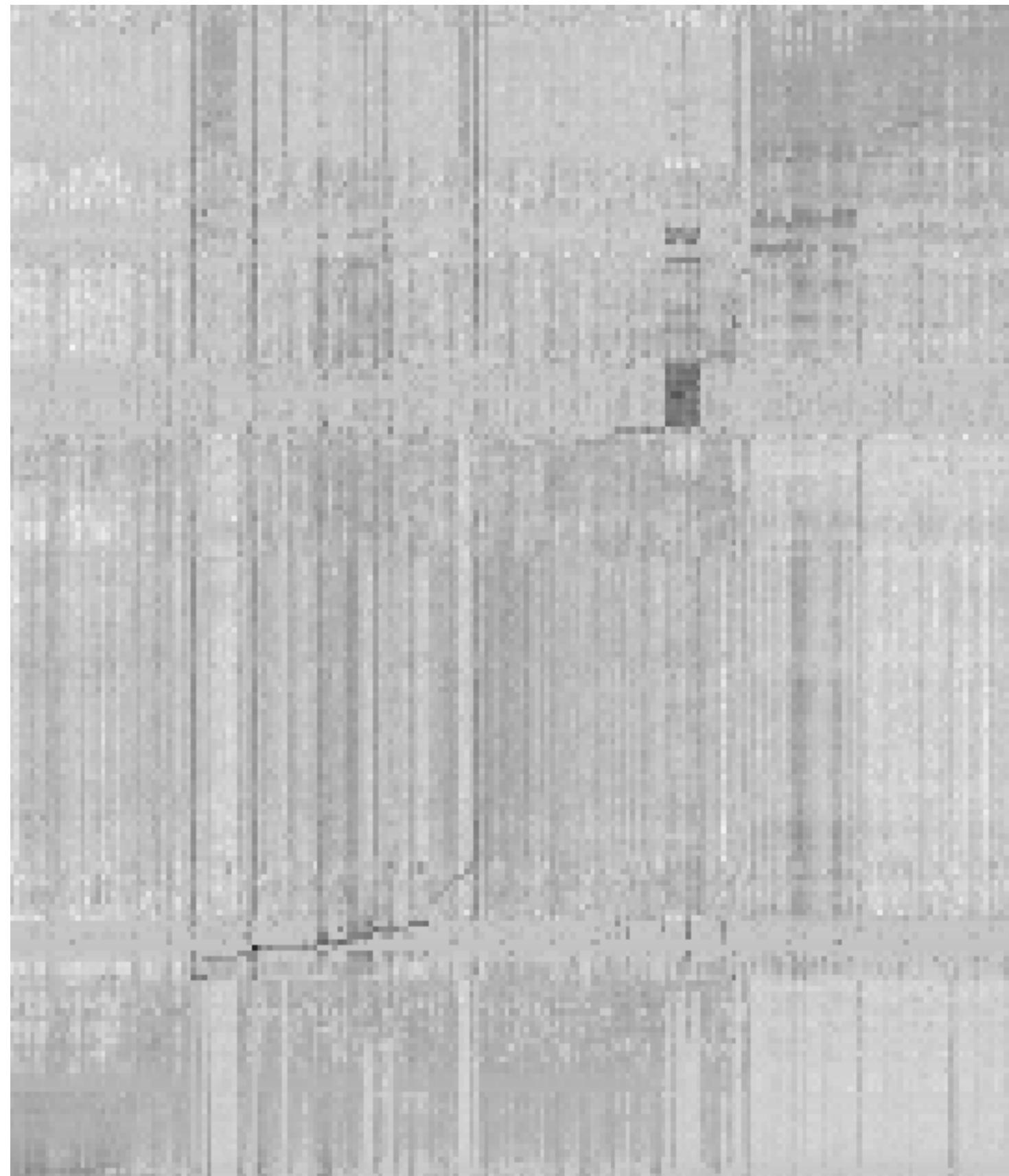


51%



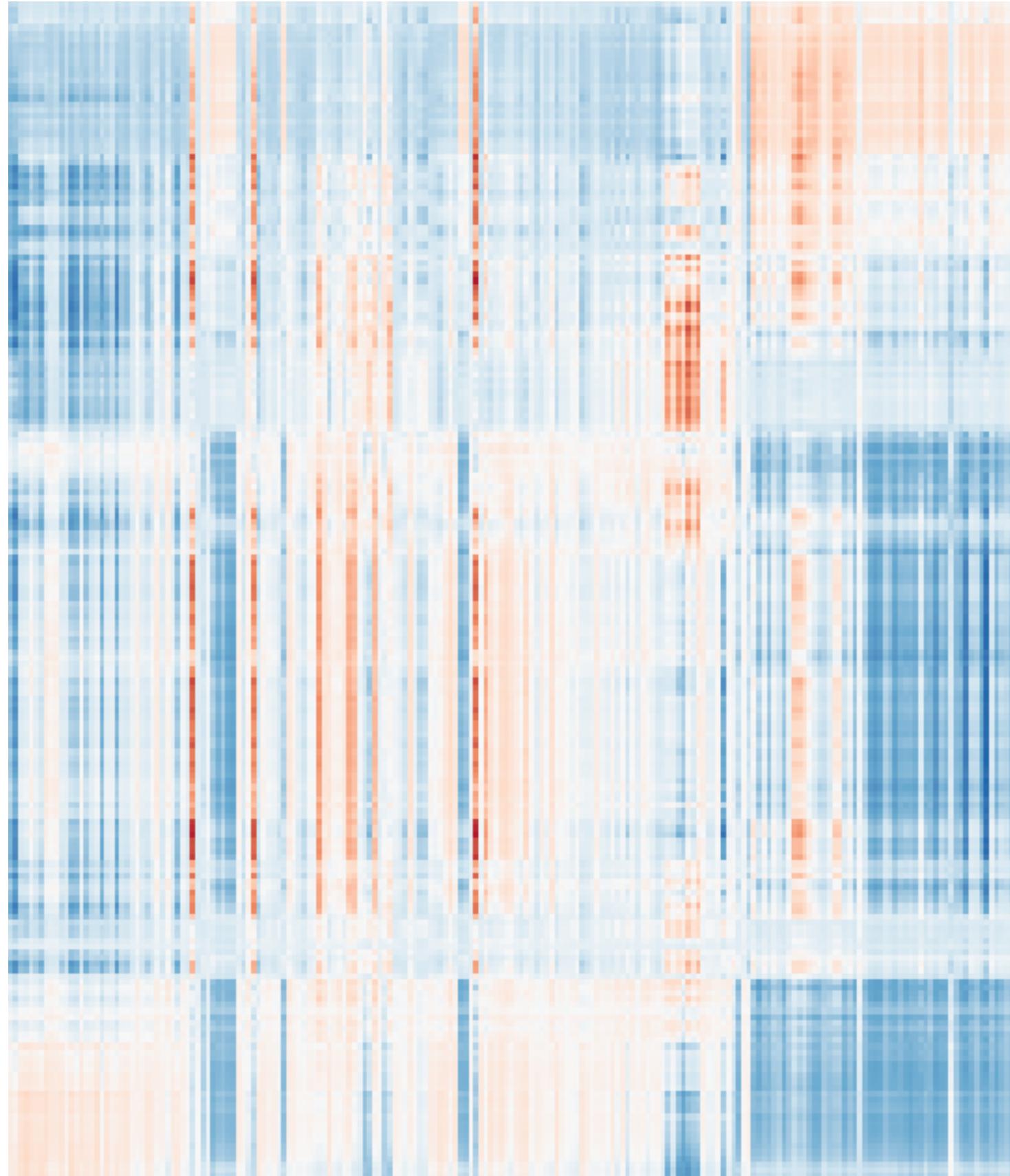
Top three eigen-vectors

TOP 200 GENES AND 200 CELL



Top three eigen-vectors

EIGEN PROJ.

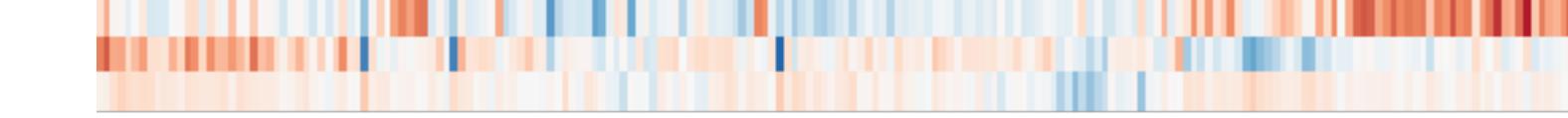


=

E-VEC



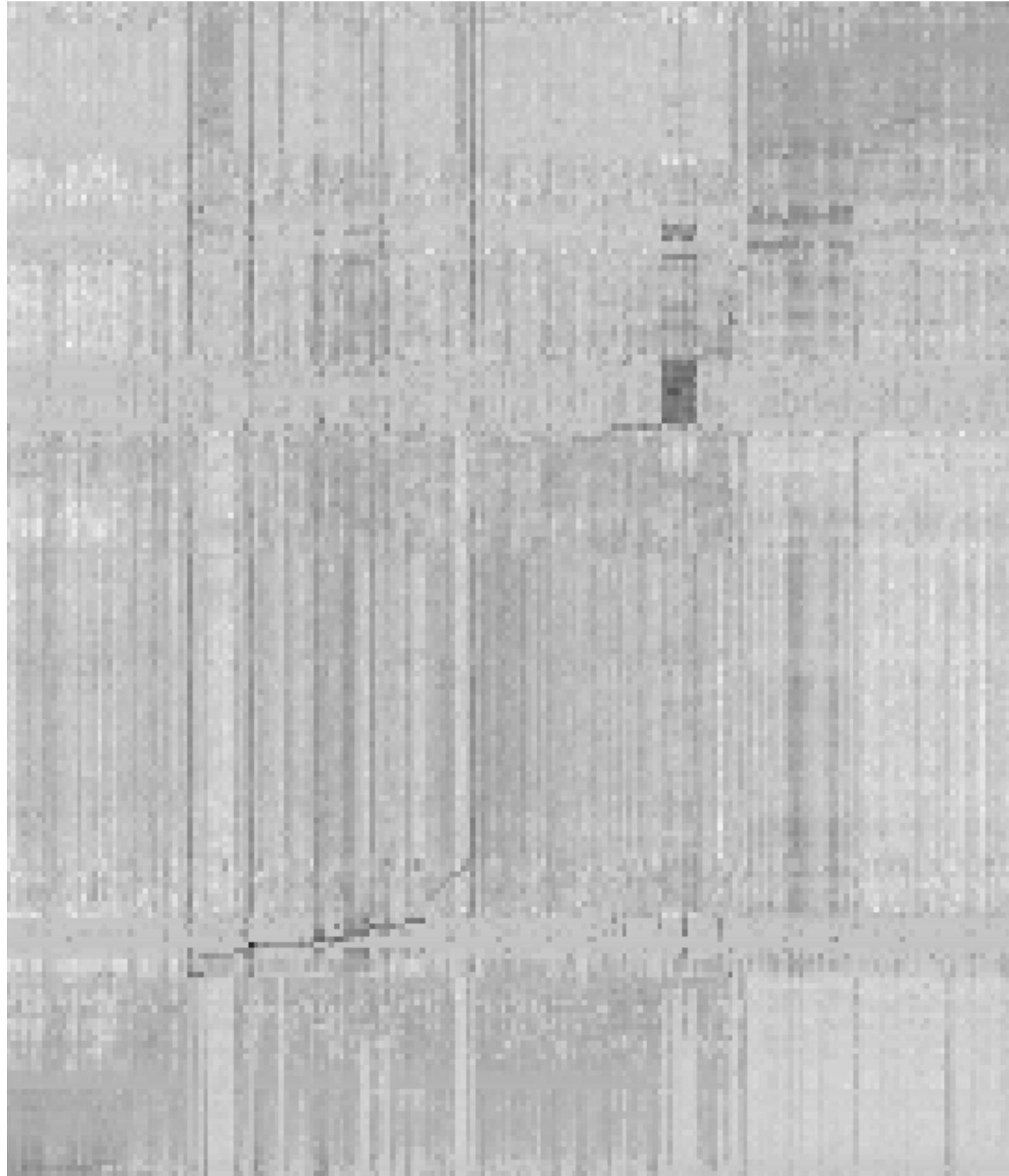
LOADING



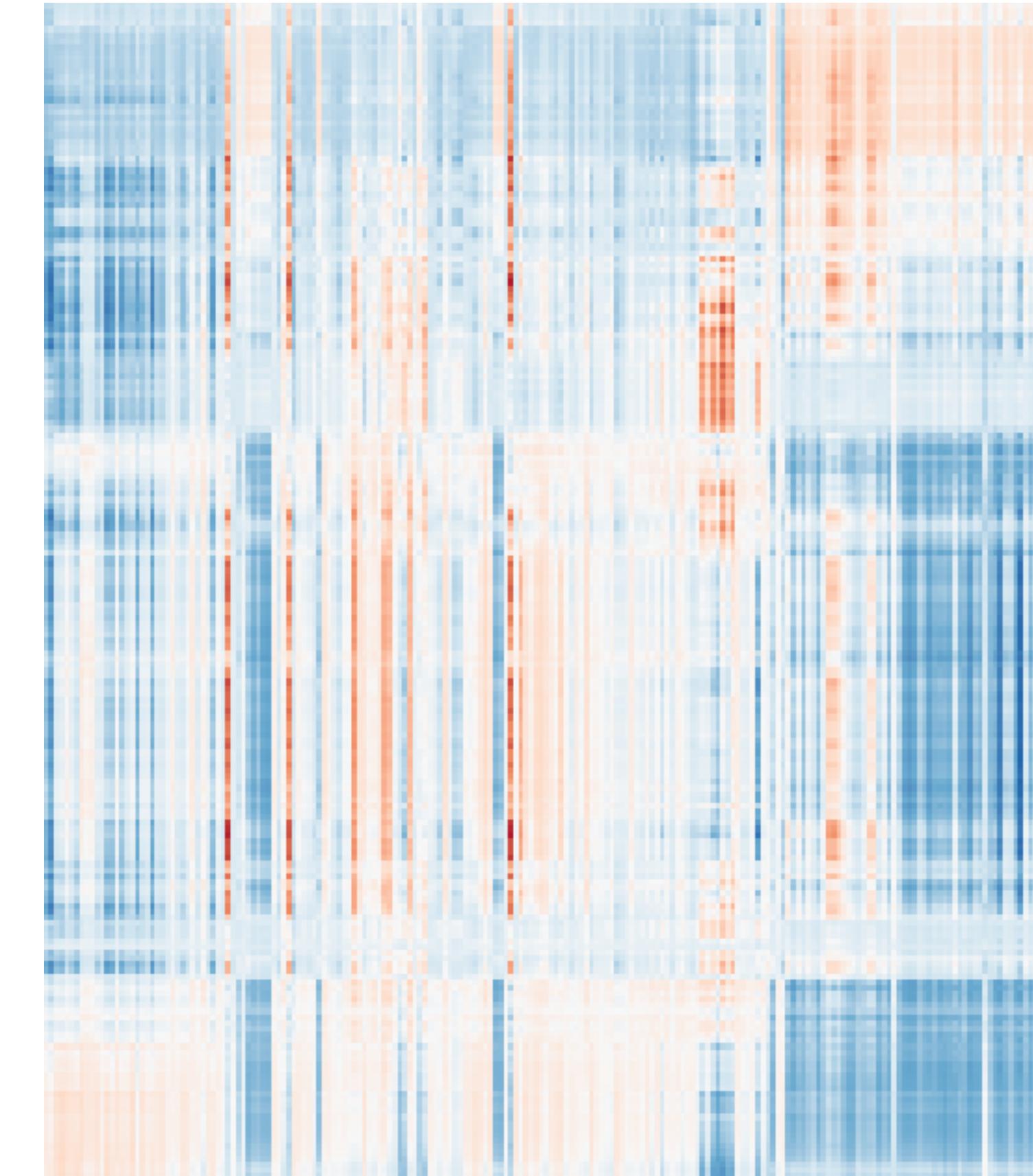
X

Top three eigen-vectors

TOP 200 GENES AND 200 CELL



58%



Singular Value Decomposition can find a PCA solution

Singular Value Decomposition

SVD identifies three matrices of X :

$$X = UDV^\top$$

where both U and V vectors are orthonormal, namely,

- $U^\top U = I, \mathbf{u}_k^\top \mathbf{u}_k = 1$ for all k ,
- $V^\top V = I, \mathbf{v}_k^\top \mathbf{v}_k = 1$ for all k .

Run SVD to find principal components

```
svd.out <- rsvd::rsvd(x.sorted, k = 7)  
U <- svd.out$u  
D <- diag(svd.out$d)  
V <- svd.out$v
```

DATA MATRIX



$$\text{DATA MATRIX} = U \times D \times \text{TRANSPOSE}(V)$$

The diagram illustrates the decomposition of a data matrix into three components: U , D , and $\text{TRANSPOSE}(V)$. The U matrix is shown as a vertical column of colored pixels, the D matrix is a small blue square representing a diagonal matrix, and the $\text{TRANSPOSE}(V)$ matrix is shown as a horizontal row of colored pixels. The multiplication of U and D is indicated by an equals sign followed by a multiplication symbol (\times).

SVD decomposes covariance calculation

Singular Value Decomposition

SVD identifies three matrices of X :

$$X = UDV^\top$$

where both U and V vectors are orthonormal, namely,

- $U^\top U = I$, $\mathbf{u}_k^\top \mathbf{u}_k = 1$ for all k ,
- $V^\top V = I$, $\mathbf{v}_k^\top \mathbf{v}_k = 1$ for all k .

Covariance by SVD

Covariance across the columns (samples)

$$X^\top X / m = V D^2 V^\top / m$$

Covariance across the rows (genes)

$$X X^\top / n = U D^2 U^\top / n$$

Remark: standardized matrix

Review: SVD finds eigenvectors of covariance

$$\underbrace{\left(\frac{1}{m} X^\top X \right)}_{\text{sample covariance}} \mathbf{v}_1 = \frac{1}{m} (\mathbf{v}_1, \mathbf{v}_2, \dots, \mathbf{v}_k) \begin{pmatrix} D_1^2 & 0 & \cdots & \cdots \\ 0 & D_2^2 & 0 & \cdots \\ 0 & \cdots & \ddots & 0 \\ 0 & \cdots & 0 & D_k^2 \end{pmatrix} \begin{pmatrix} \mathbf{v}_1^\top \\ \mathbf{v}_2^\top \\ \vdots \\ \mathbf{v}_k^\top \end{pmatrix} \mathbf{v}_1$$

Review: SVD finds eigenvectors of covariance

$$\underbrace{\left(\frac{1}{m} X^\top X \right)}_{\text{sample covariance}} \mathbf{v}_1 = \frac{1}{m} (\mathbf{v}_1, \mathbf{v}_2, \dots, \mathbf{v}_k) \begin{pmatrix} D_1^2 & 0 & \cdots & \cdots \\ 0 & D_2^2 & 0 & \cdots \\ 0 & \cdots & \ddots & 0 \\ 0 & \cdots & 0 & D_k^2 \end{pmatrix} \begin{pmatrix} \mathbf{v}_1^\top \\ \mathbf{v}_2^\top \\ \vdots \\ \mathbf{v}_k^\top \end{pmatrix} \mathbf{v}_1$$
$$= \frac{1}{m} (\mathbf{v}_1, \mathbf{v}_2, \dots, \mathbf{v}_k) \begin{pmatrix} D_1^2 & 0 & \cdots & \cdots \\ 0 & D_2^2 & 0 & \cdots \\ 0 & \cdots & \ddots & 0 \\ 0 & \cdots & 0 & D_k^2 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix}$$

Review: SVD finds eigenvectors of covariance

$$\begin{aligned} \underbrace{\left(\frac{1}{m} X^\top X \right)}_{\text{sample covariance}} \mathbf{v}_1 &= \frac{1}{m} (\mathbf{v}_1, \mathbf{v}_2, \dots, \mathbf{v}_k) \begin{pmatrix} D_1^2 & 0 & \cdots & \cdots \\ 0 & D_2^2 & 0 & \cdots \\ 0 & \cdots & \ddots & 0 \\ 0 & \cdots & 0 & D_k^2 \end{pmatrix} \begin{pmatrix} \mathbf{v}_1^\top \\ \mathbf{v}_2^\top \\ \vdots \\ \mathbf{v}_k^\top \end{pmatrix} \mathbf{v}_1 \\ &= \frac{1}{m} (\mathbf{v}_1, \mathbf{v}_2, \dots, \mathbf{v}_k) \begin{pmatrix} D_1^2 & 0 & \cdots & \cdots \\ 0 & D_2^2 & 0 & \cdots \\ 0 & \cdots & \ddots & 0 \\ 0 & \cdots & 0 & D_k^2 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix} \\ &= \frac{1}{m} (\mathbf{v}_1, \mathbf{v}_2, \dots, \mathbf{v}_k) \begin{pmatrix} D_1^2 \\ 0 \\ 0 \\ 0 \end{pmatrix} \end{aligned}$$

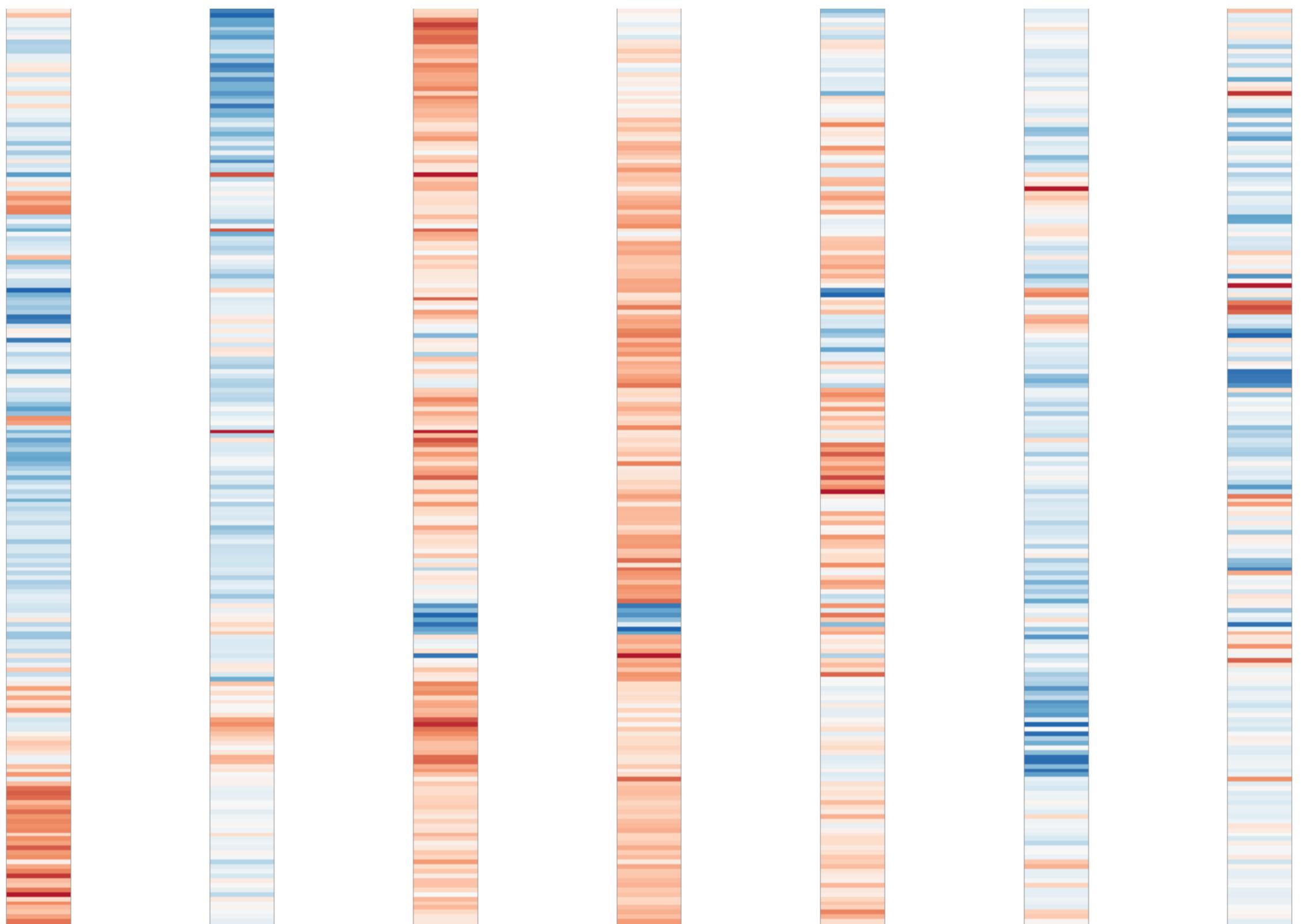
Review: SVD finds eigenvectors of covariance

$$\begin{aligned} \underbrace{\left(\frac{1}{m} X^\top X \right)}_{\text{sample covariance}} \mathbf{v}_1 &= \frac{1}{m} (\mathbf{v}_1, \mathbf{v}_2, \dots, \mathbf{v}_k) \begin{pmatrix} D_1^2 & 0 & \cdots & \cdots \\ 0 & D_2^2 & 0 & \cdots \\ 0 & \cdots & \ddots & 0 \\ 0 & \cdots & 0 & D_k^2 \end{pmatrix} \begin{pmatrix} \mathbf{v}_1^\top \\ \mathbf{v}_2^\top \\ \vdots \\ \mathbf{v}_k^\top \end{pmatrix} \mathbf{v}_1 \\ &= \frac{1}{m} (\mathbf{v}_1, \mathbf{v}_2, \dots, \mathbf{v}_k) \begin{pmatrix} D_1^2 & 0 & \cdots & \cdots \\ 0 & D_2^2 & 0 & \cdots \\ 0 & \cdots & \ddots & 0 \\ 0 & \cdots & 0 & D_k^2 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix} \\ \hat{\Sigma} \mathbf{v}_1 &= \frac{1}{m} (\mathbf{v}_1, \mathbf{v}_2, \dots, \mathbf{v}_k) \begin{pmatrix} D_1^2 \\ 0 \\ 0 \\ 0 \end{pmatrix} = \underbrace{\frac{D_1^2}{m}}_{\text{eigenvalue}} \underbrace{\mathbf{v}_1}_{\text{eigenvector}} \end{aligned}$$

Eigen values λ 's quantify the proportion of variance

Given this SVD, $X = UDV^\top$, where $V = [\mathbf{v}_1, \dots, \mathbf{v}_k]$,

$\mathbf{v}_{[,1]} \quad \mathbf{v}_{[,2]} \quad \mathbf{v}_{[,3]} \quad \mathbf{v}_{[,4]} \quad \mathbf{v}_{[,5]} \quad \mathbf{v}_{[,6]} \quad \mathbf{v}_{[,7]}$



V is orthonormal (orthogonal and normalized):

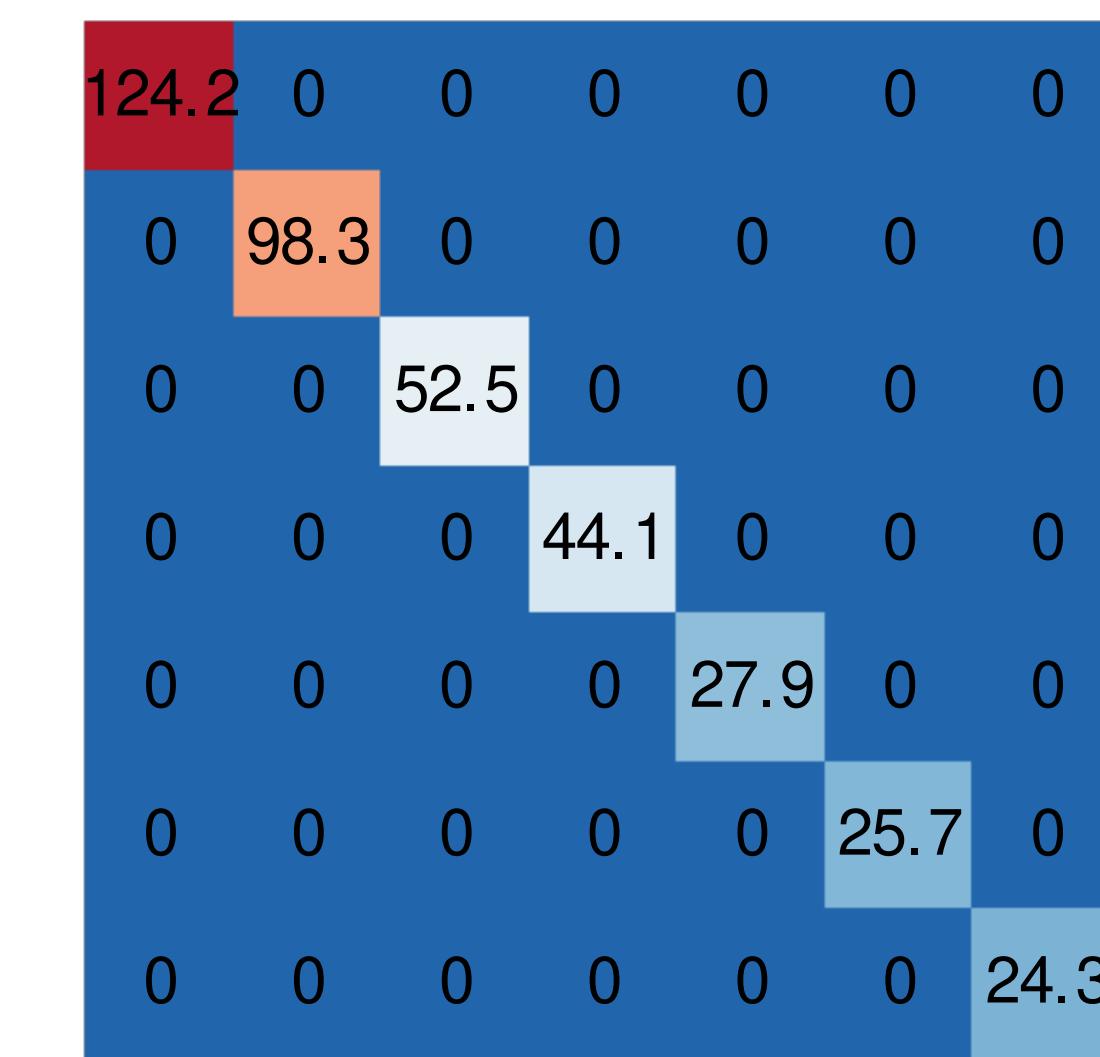
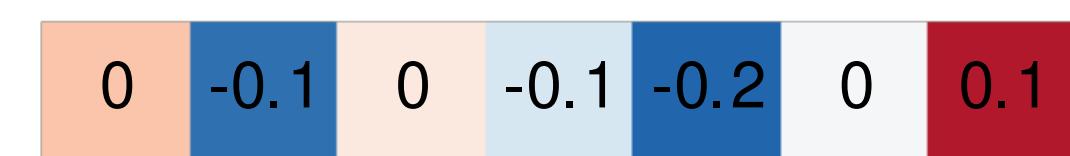
$\text{TRANSPOSE}(\mathbf{v}) * \mathbf{v}$

1	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0
0	0	1	0	0	0	0	0
0	0	0	1	0	0	0	0
0	0	0	0	1	0	0	0
0	0	0	0	0	1	0	0
0	0	0	0	0	0	1	0
0	0	0	0	0	0	0	1

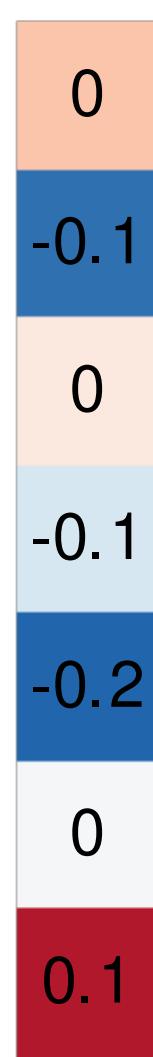
Eigen values λ 's quantify the proportion of variance

Let's consider an element (i, j) of the covariance

$$\frac{1}{m} (X^\top X)_{ij} = \frac{1}{m} (V_{i1}, V_{i2}, \dots, V_{ik}) \begin{pmatrix} D_1^2 & 0 & \dots & \dots \\ 0 & D_2^2 & 0 & \dots \\ 0 & \dots & \ddots & 0 \\ 0 & \dots & 0 & D_k^2 \end{pmatrix} \begin{pmatrix} V_{j1} \\ V_{j2} \\ \vdots \\ V_{jk} \end{pmatrix}$$



\times



\times

Eigen values λ 's quantify the proportion of variance

Let's consider an element (i, j) of the covariance

$$\frac{1}{m} (X^\top X)_{ij} = \frac{1}{m} (V_{i1}, V_{i2}, \dots, V_{ik}) \begin{pmatrix} D_1^2 & 0 & \dots & \dots \\ 0 & D_2^2 & 0 & \dots \\ 0 & \dots & \ddots & 0 \\ 0 & \dots & 0 & D_k^2 \end{pmatrix} \begin{pmatrix} V_{j1} \\ V_{j2} \\ \vdots \\ V_{jk} \end{pmatrix}$$

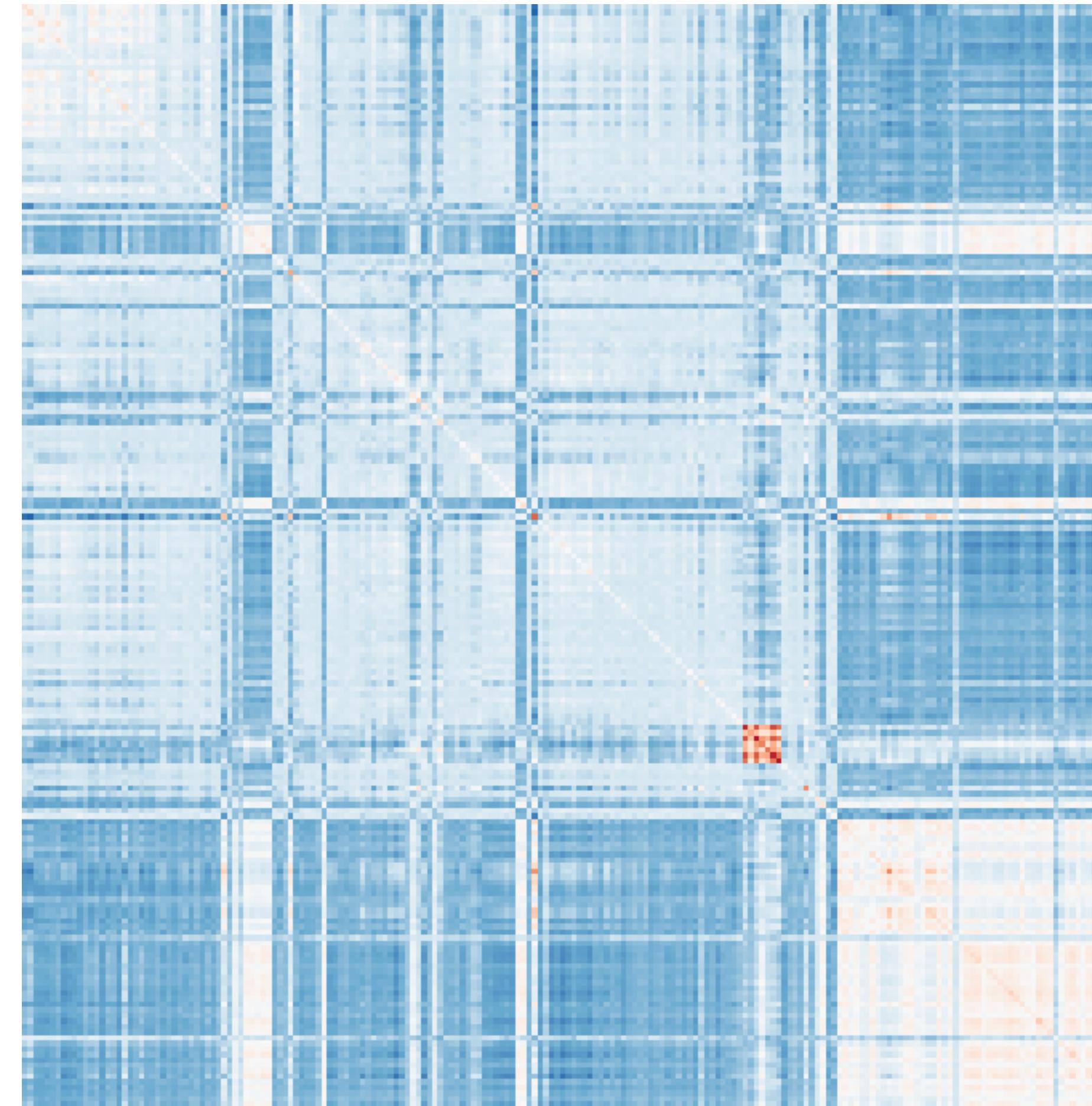
Therefore,

$$\hat{\Sigma}_{ij} = \frac{D_1^2}{m} V_{i1} V_{j1} + \frac{D_2^2}{m} V_{i2} V_{j2} + \dots + \frac{D_k^2}{m} V_{ik} V_{jk}$$

We have k terms for the covariance between i and j .

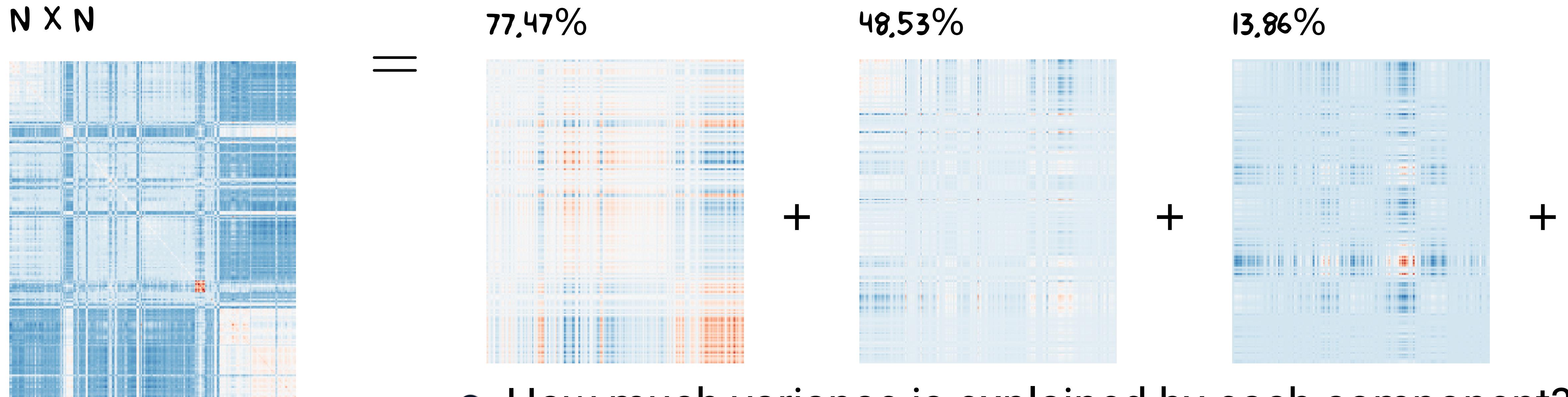
Sample covariance matrix

$$\hat{\Sigma} = X^\top X / m$$



PCs (SVs) decompose the covariance matrix

$$\hat{\Sigma} = X^\top X / m$$

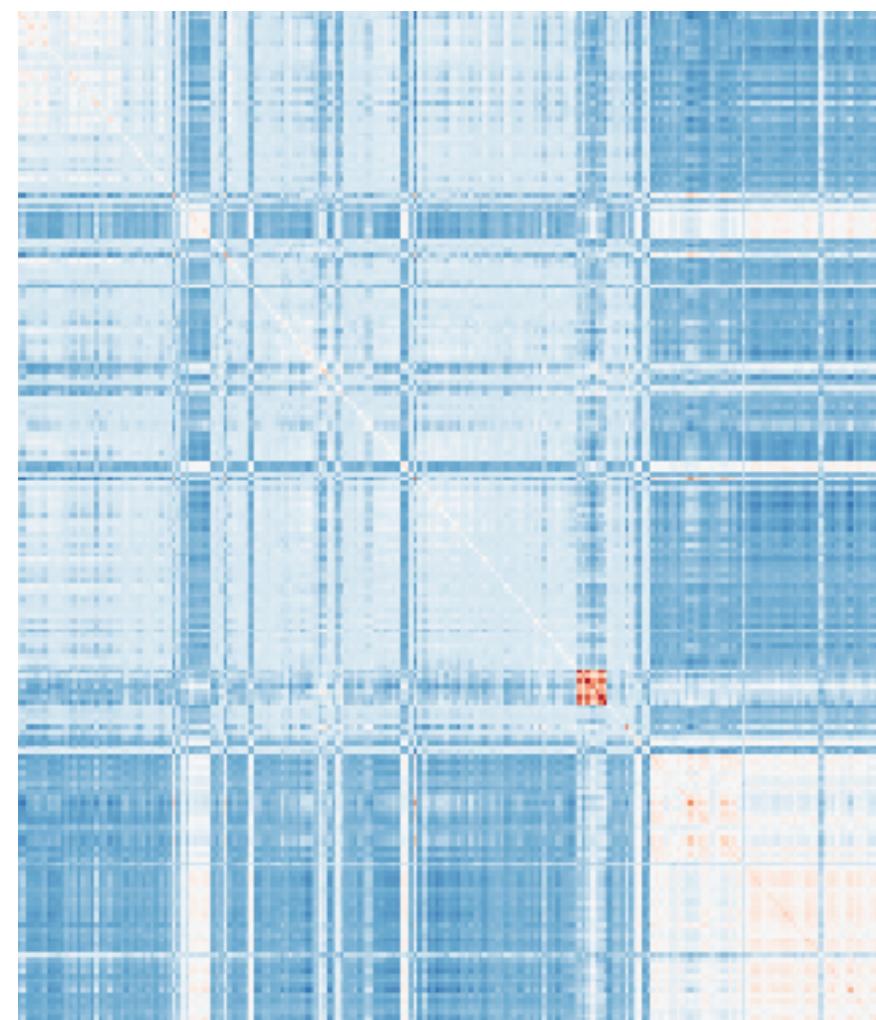


- How much variance is explained by each component?

PCs (SVs) decompose the covariance matrix

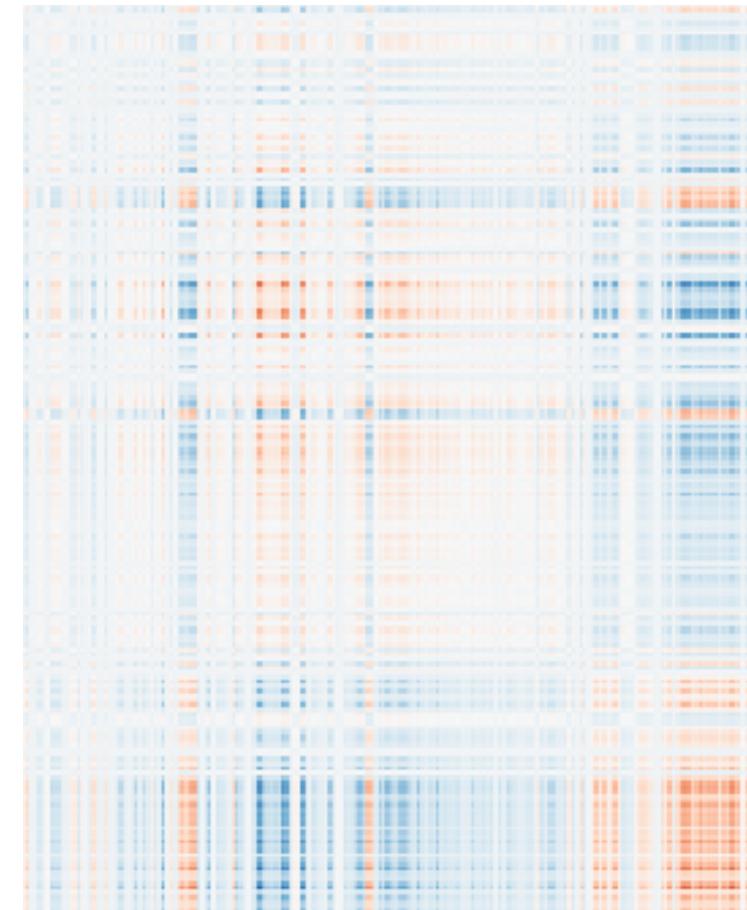
$$\hat{\Sigma} = X^\top X / m = V D^2 V^\top / m$$

N X N

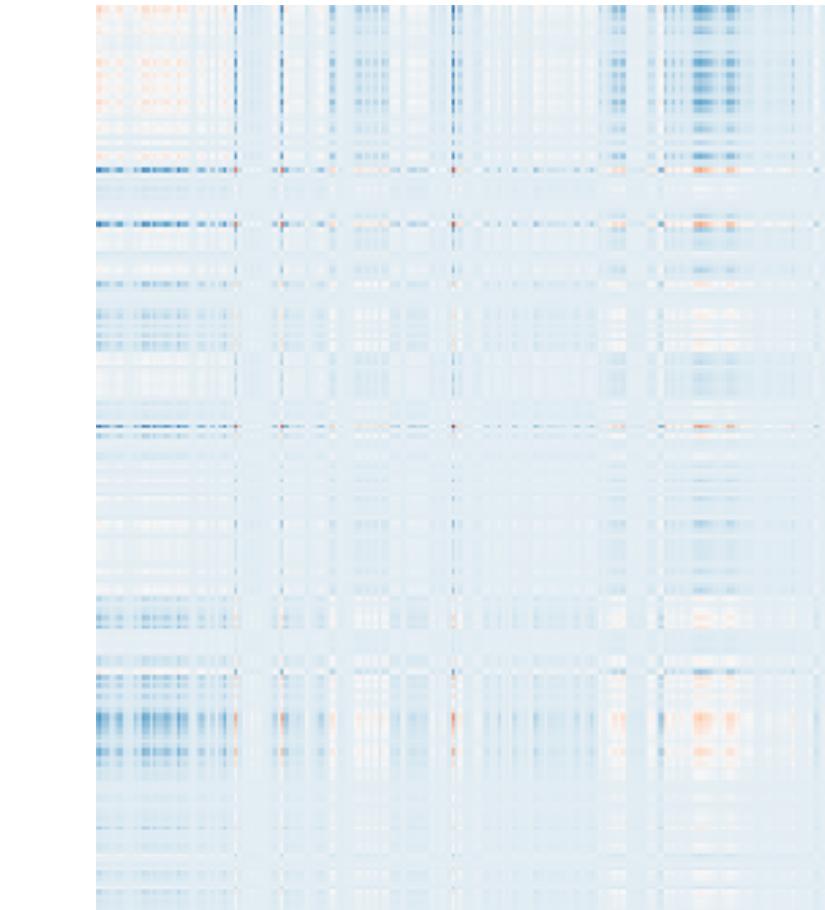


=

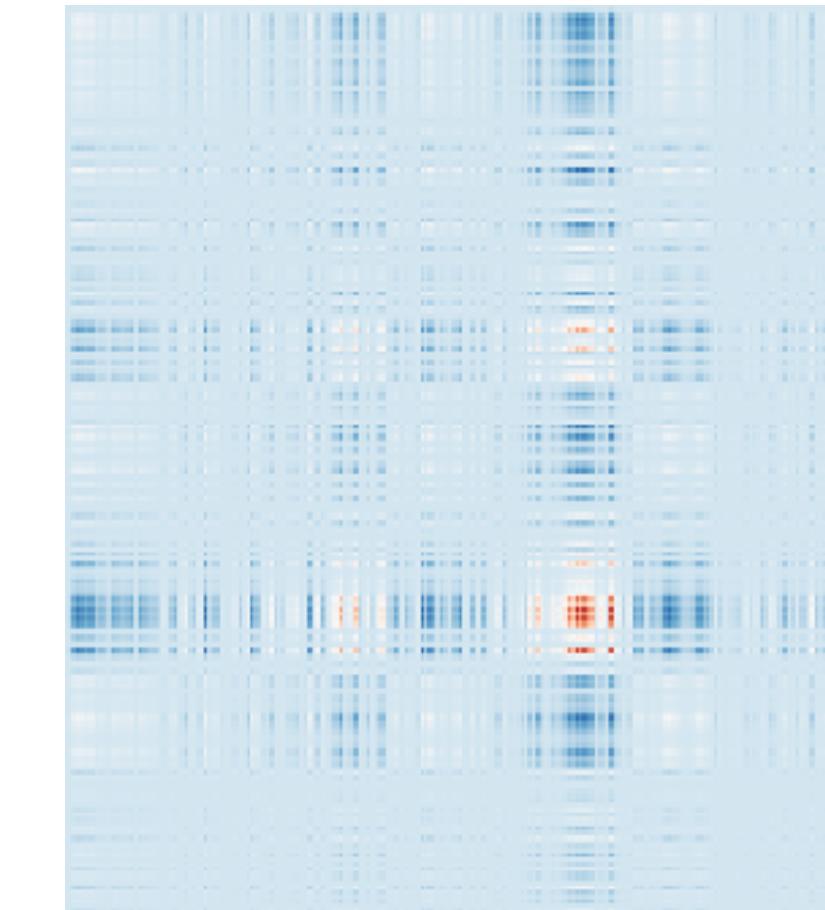
77.47%



48.53%



13.86%



+

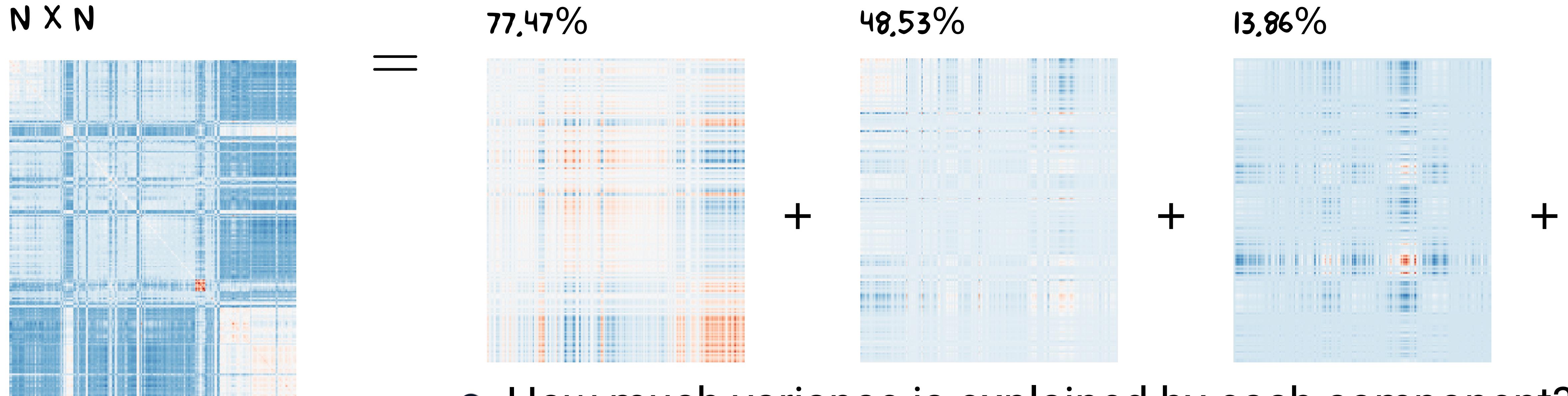
+

+

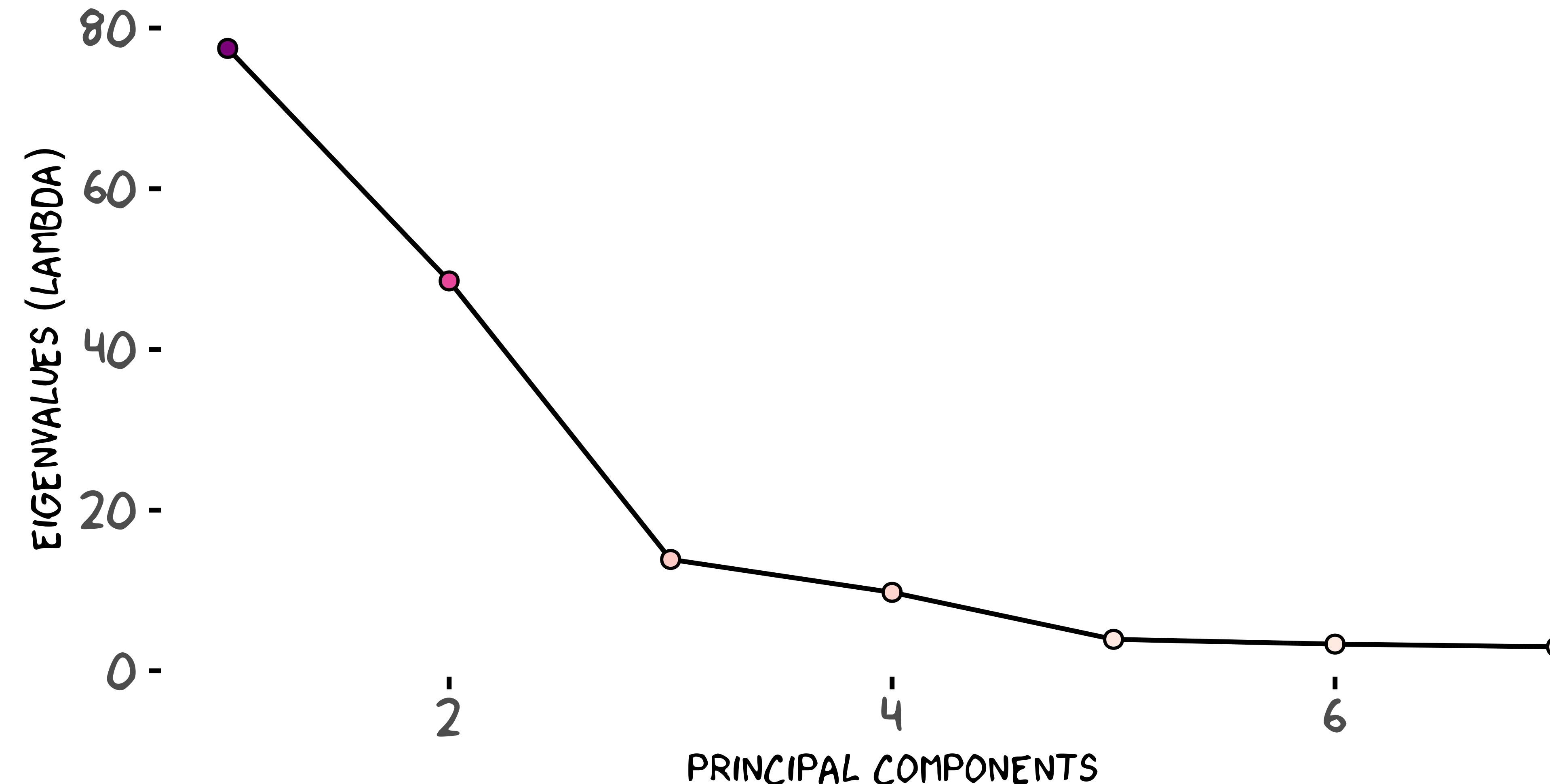
- How much variance is explained by each component?

PCs (SVs) decompose the covariance matrix

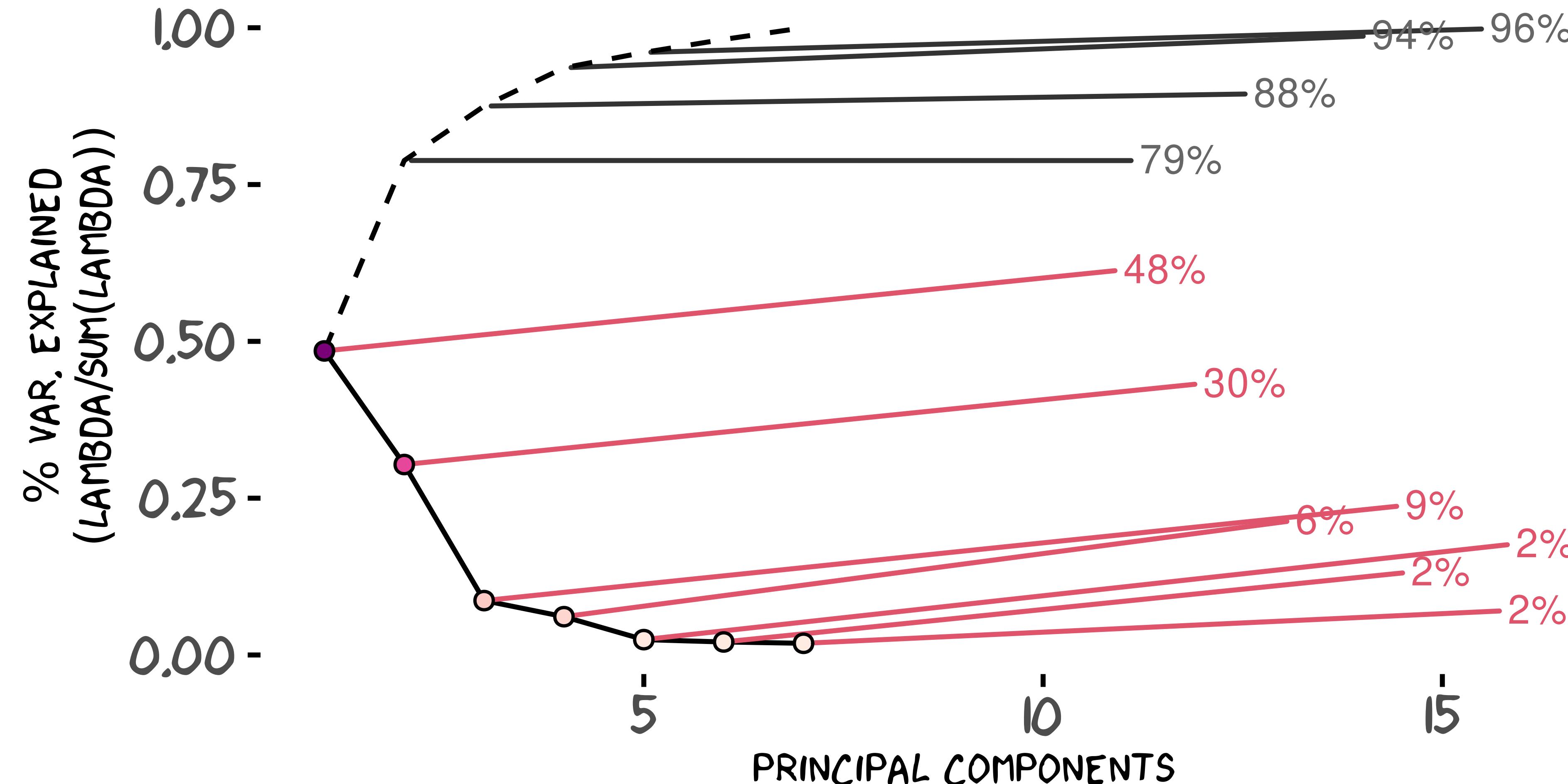
$$\hat{\Sigma} = X^\top X/m = VD^2V^\top/m = \sum_{k=1} \lambda_k \mathbf{v}_k \mathbf{v}_k^\top, \quad \lambda_k = D_k^2/m$$



How much variance is explained by each principal component?



How much variance is explained by each principal component?



Today's lecture

- 1 (review) Principal Component Analysis
- 2 Non-negative matrix factorization
- 3 Differential gene expression analysis

SVD works great. Why do we need another factorization method?

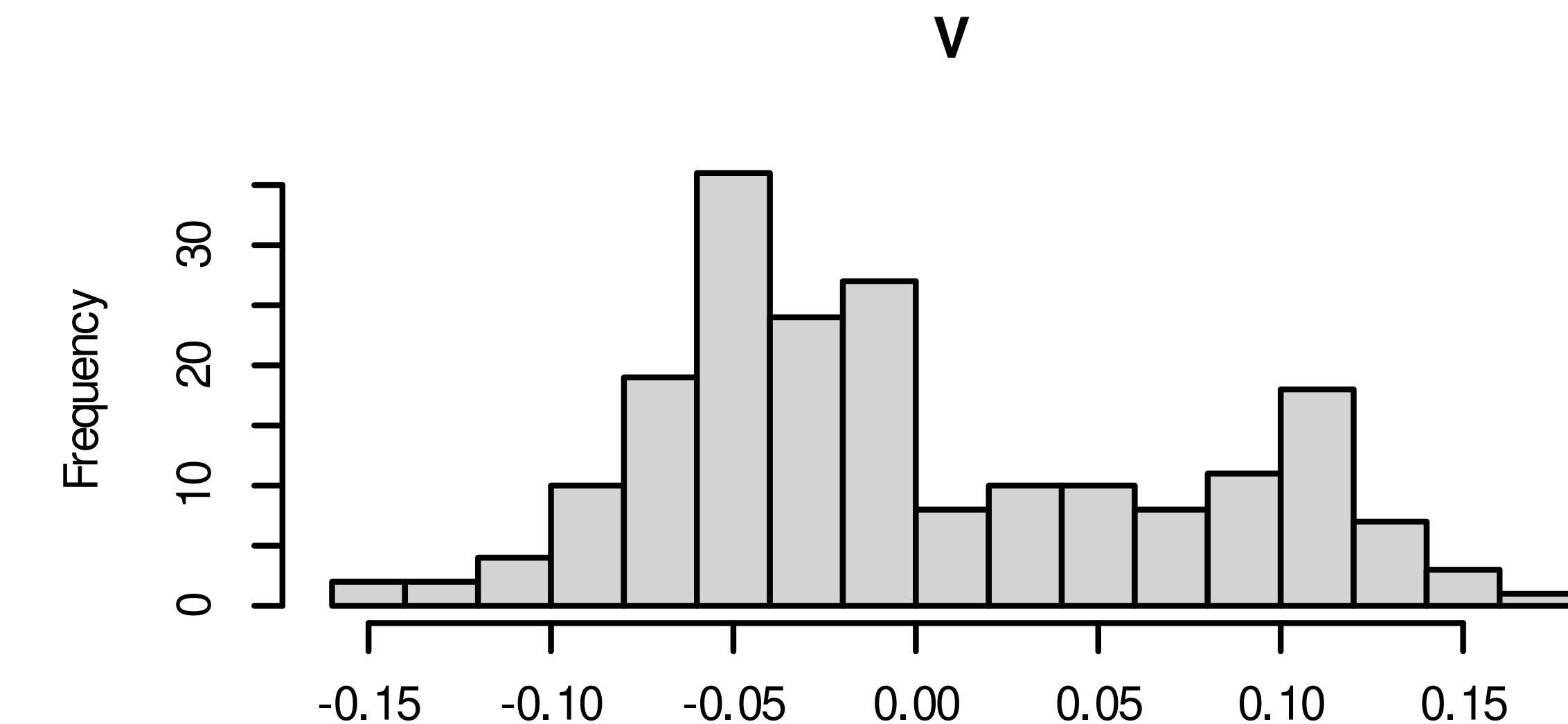
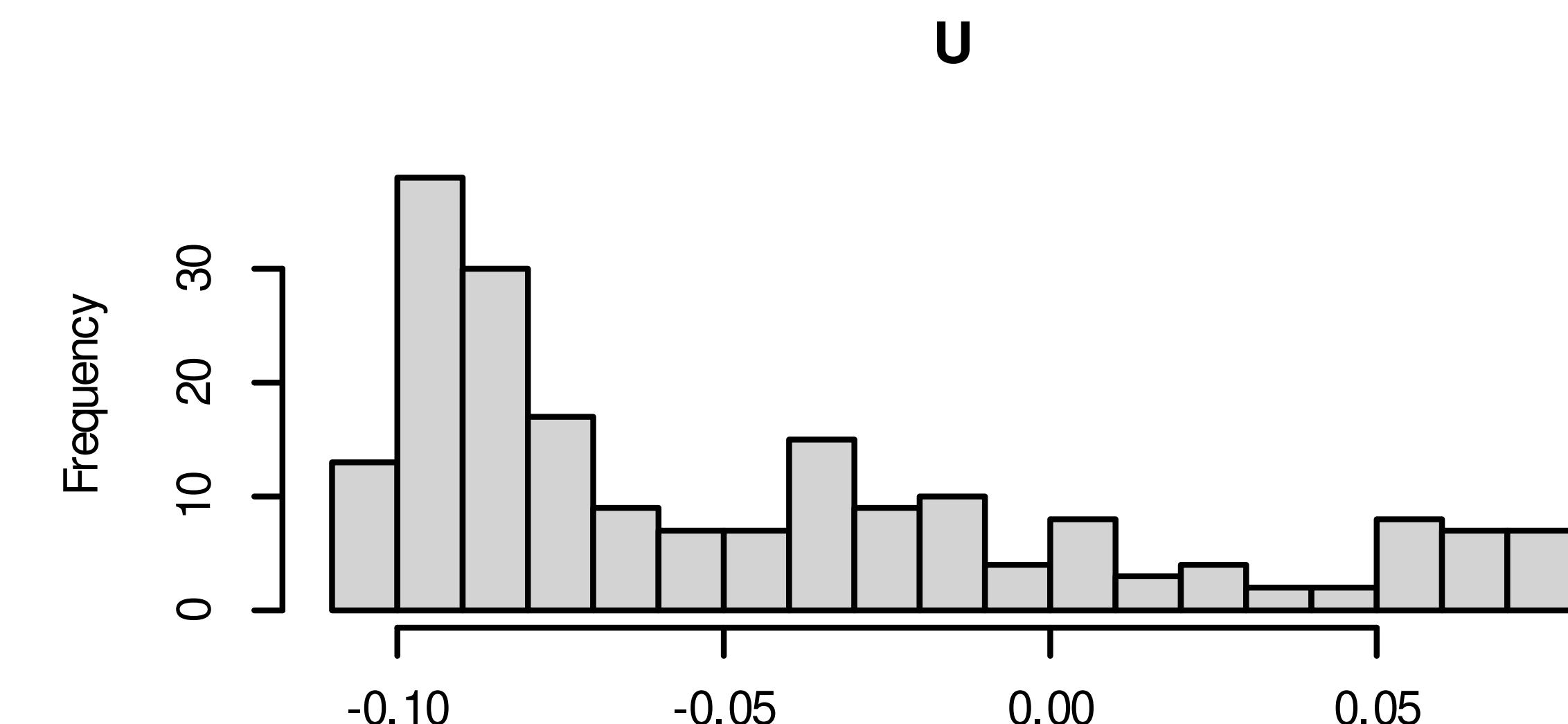
Why another factorization method?

- The count matrix X is non-negative.

```
sum(X < 0)
```

```
## [1] 0
```

- Singular vectors are signed.
 - The “positive” and “negative” values can cancel each other and become “zero.”
 - “Negative” contributions are difficult to interpret.

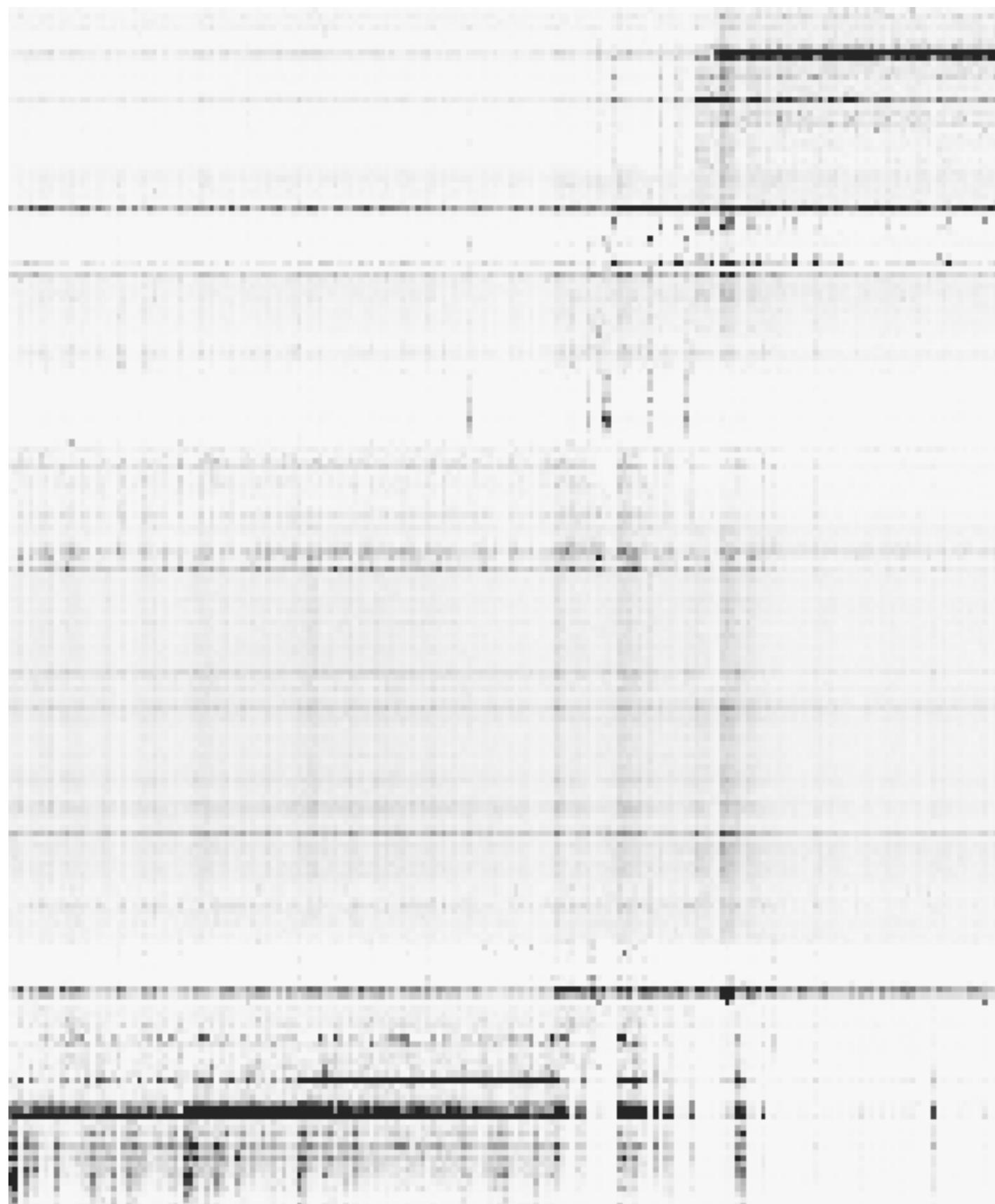


Non-negative factorization

```
out <- NMF::nmf(xx, 7)
U <- NMF::basis(out)
V <- t(NMF::coef(out))
```

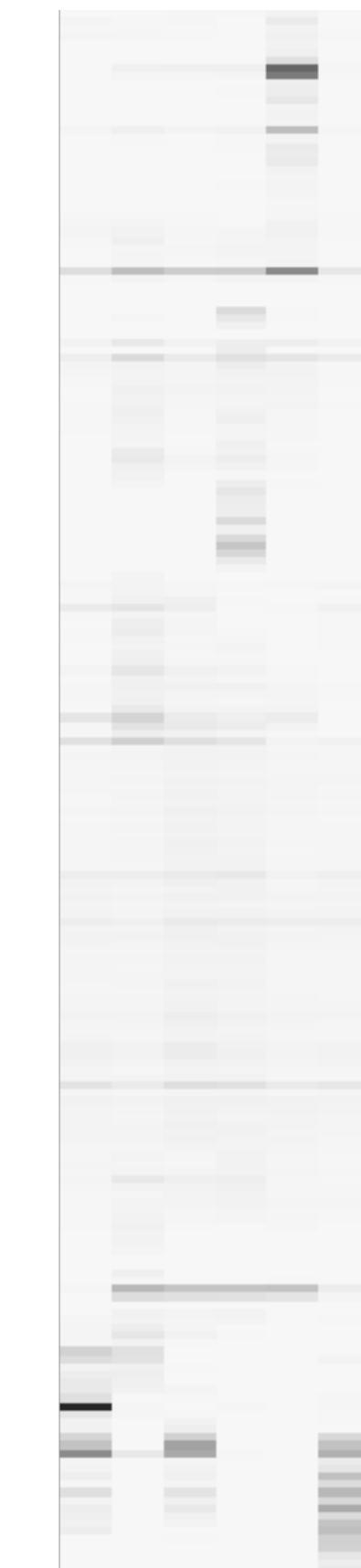
NMF naturally induces (1) gene sets (2) cellular topics

DATA (GENE \times CELL)



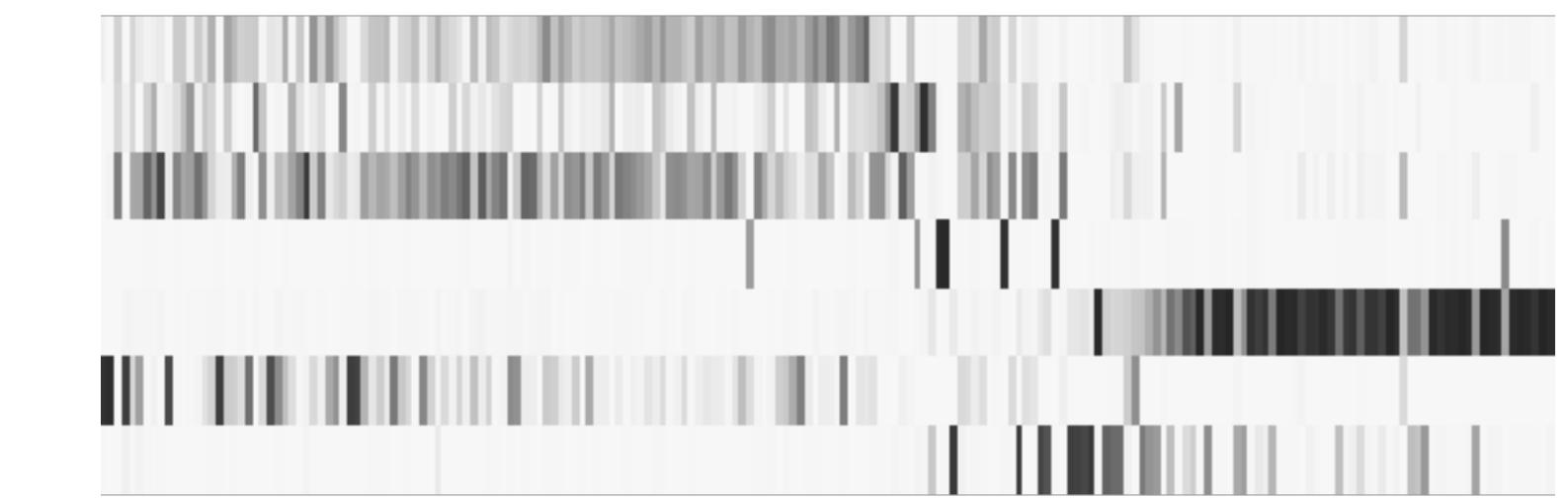
U

\sim



\times

TRANSPOSE(V)



Several notable R packages to fit NMF for single-cell data analysis

- fastTopics: estimates Poisson NMF and provides plotting and differential expression utility functions
- rliger: provides integrative analysis routines and fast online learning methods; but, it will select features arbitrarily

fastTopics relates NMF results to topic models

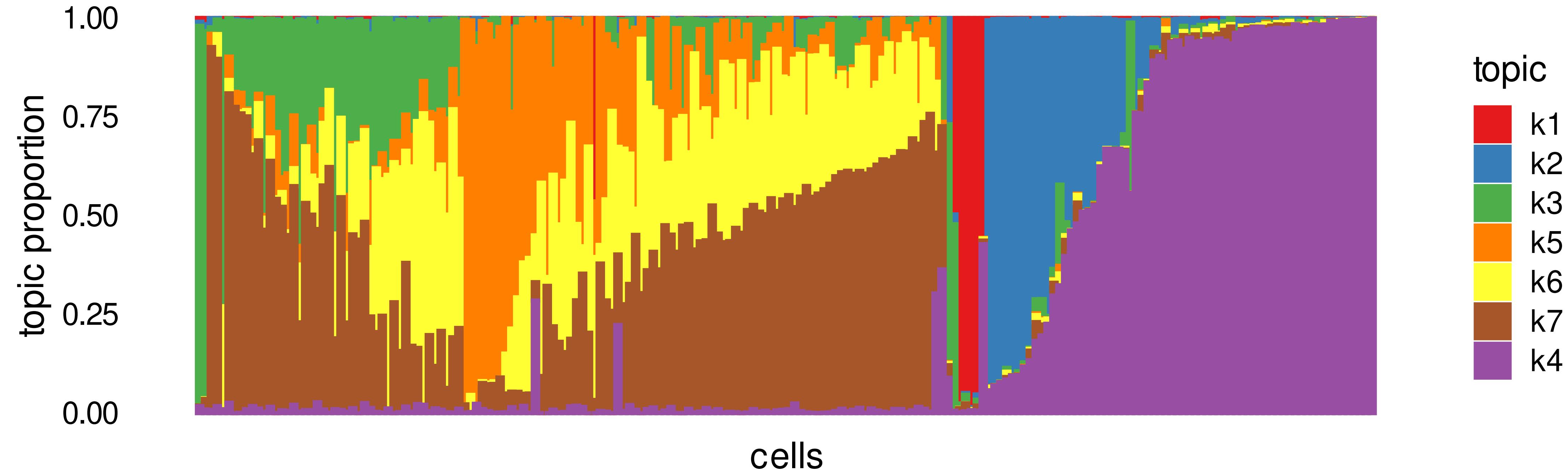
```
library(fastTopics)
out <- fit_topic_model(t(xx), k = 7)

## Initializing factors using Topic SCORE algorithm.
## Initializing loadings by running 10 SCD updates.
## Fitting rank-7 Poisson NMF to 200 x 200 dense matrix.
## Running 100 EM updates, without extrapolation (fastTopics 0.6-163).
## Refining model fit.
## Fitting rank-7 Poisson NMF to 200 x 200 dense matrix.
## Running 100 SCD updates, with extrapolation (fastTopics 0.6-163).
```

Carbonetto .. Stephens, Genome Biology (2023)

A structure plot in fastTopics package

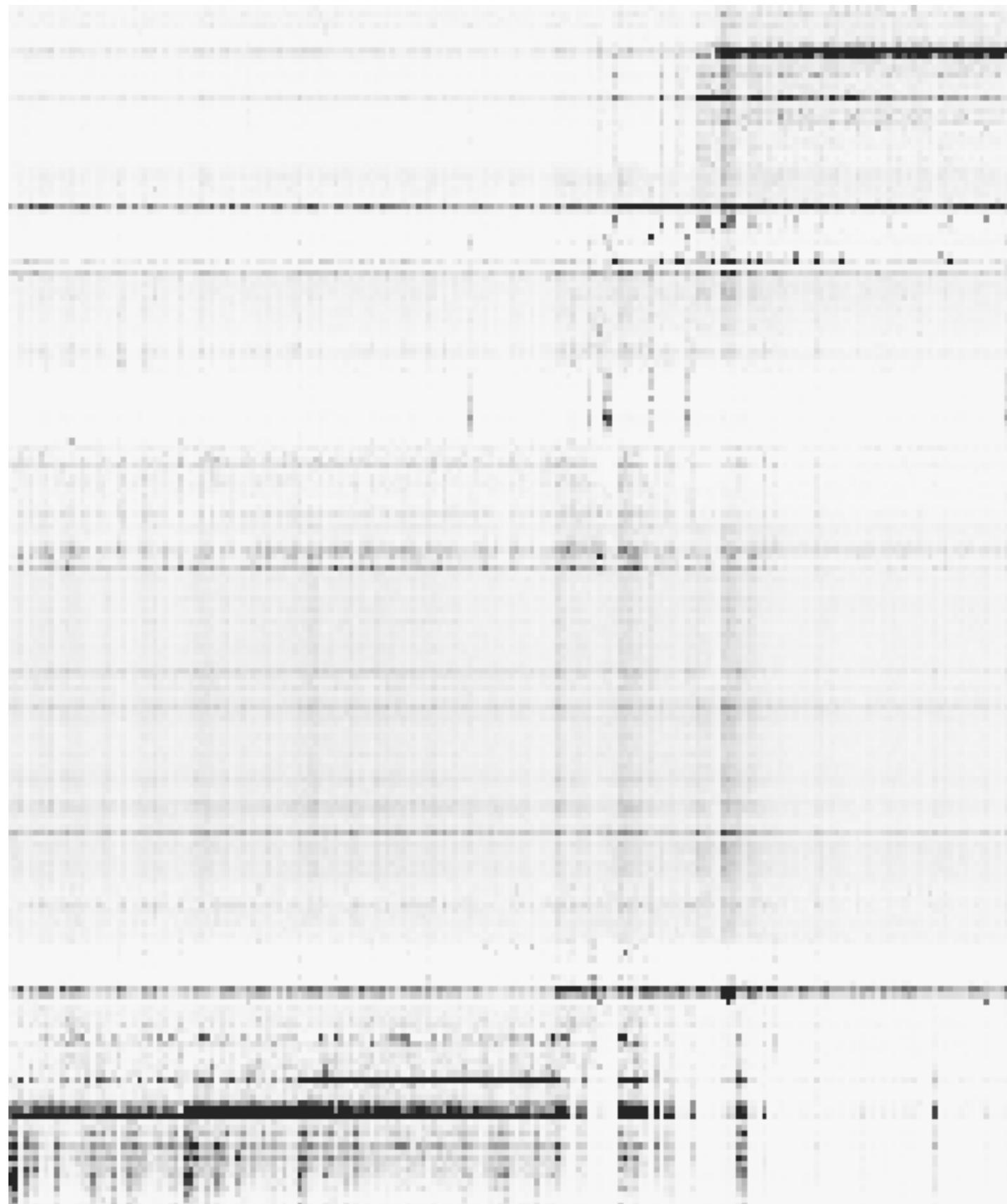
```
structure_plot(out, verbose = F) + xlab("cells")
```



Carbonetto .. Stephens, Genome Biology (2023)

Poisson NMF by fastTopics

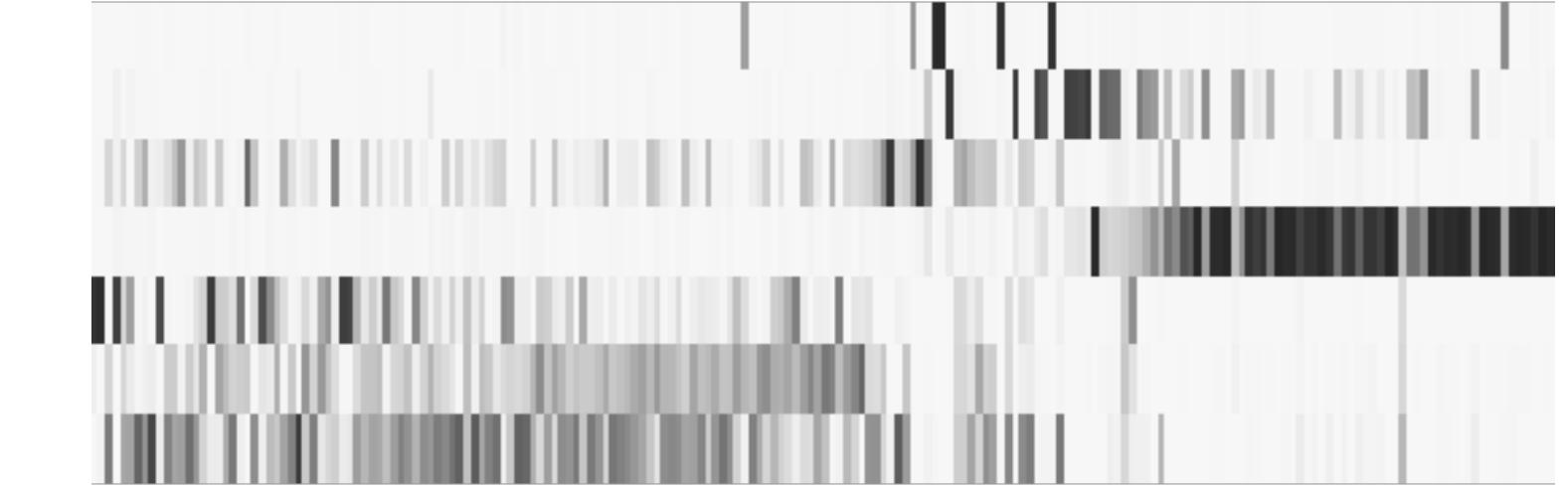
DATA (GENE X CELL)



GENE X F



FACTOR X CELL



~

×

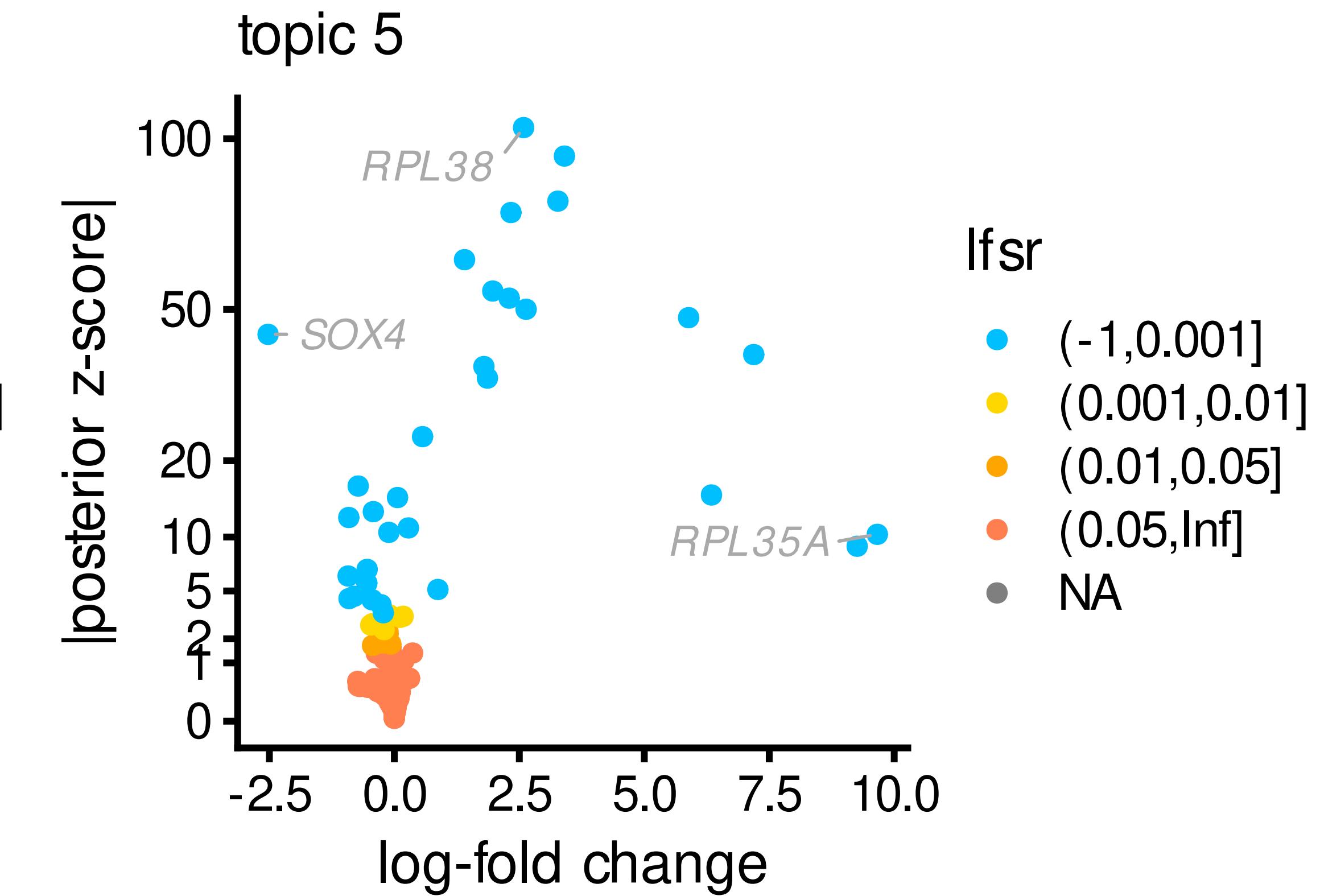
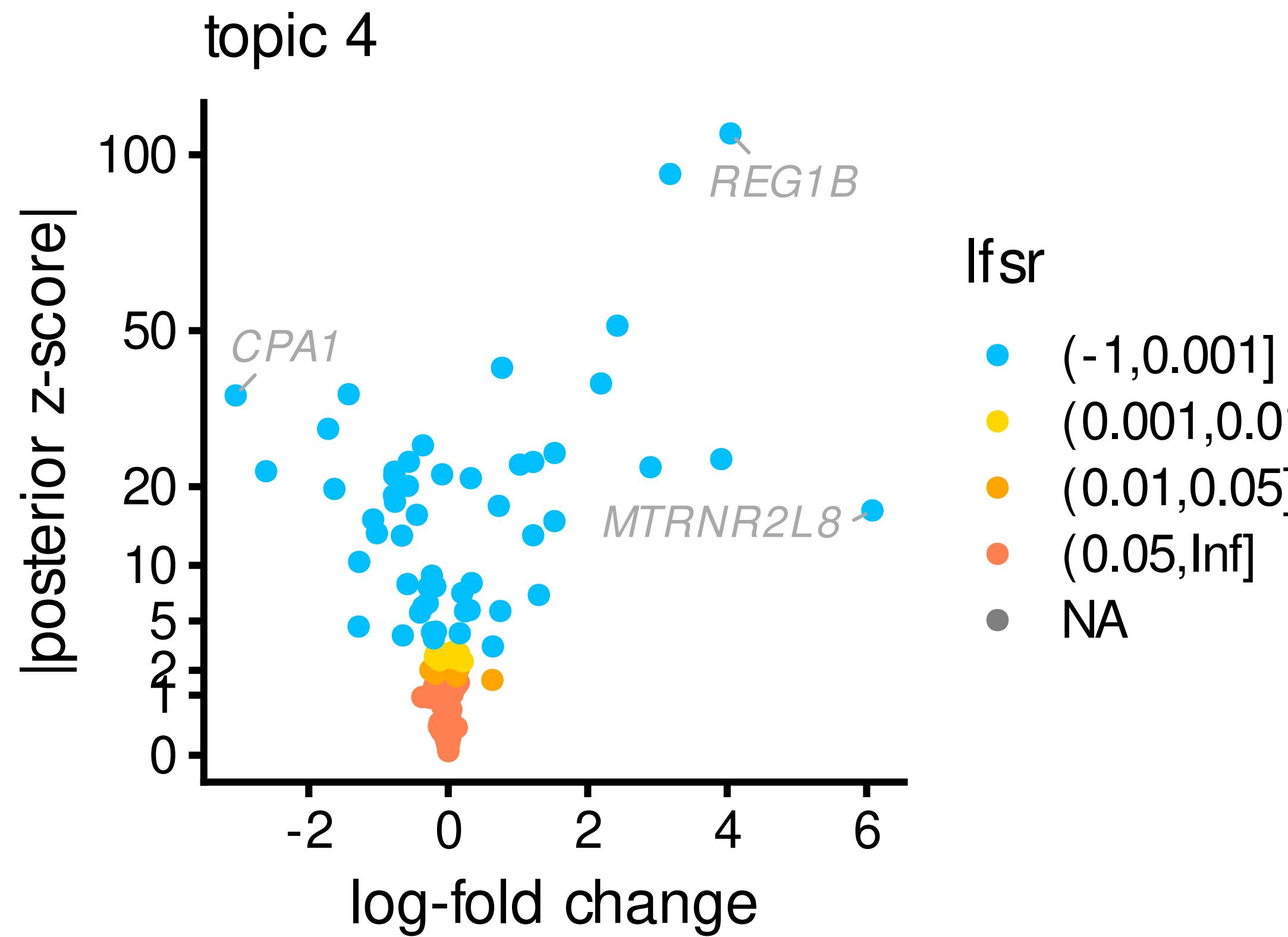
fastTopics' differential expression analysis to find topic-specific genes

```
de.out <- de_analysis(out, t(xx))

## Fitting 200 Poisson models with k=7 using method="scd".
## Computing log-fold change statistics from 200 Poisson models with k=7.
## Stabilizing posterior log-fold change estimates using adaptive shrinkage.
```

Carbonetto .. Stephens, Genome Biology (2023)

fastTopics' differential expression analysis to find topic-specific genes



rlicher provides a more scalable option for NMF estimation

```
library(rlicher)
.liger <- createLiger(list(rna = X))
.liger <- normalize(.liger)
.liger <- selectGenes(.liger, var.thresh = 0.2, do.plot = F)
.liger <- scaleNotCenter(.liger)
.liger <- online_iNMF(.liger, k = 7, miniBatch_size = 100)
```

```
## Starting Online iNMF...
```

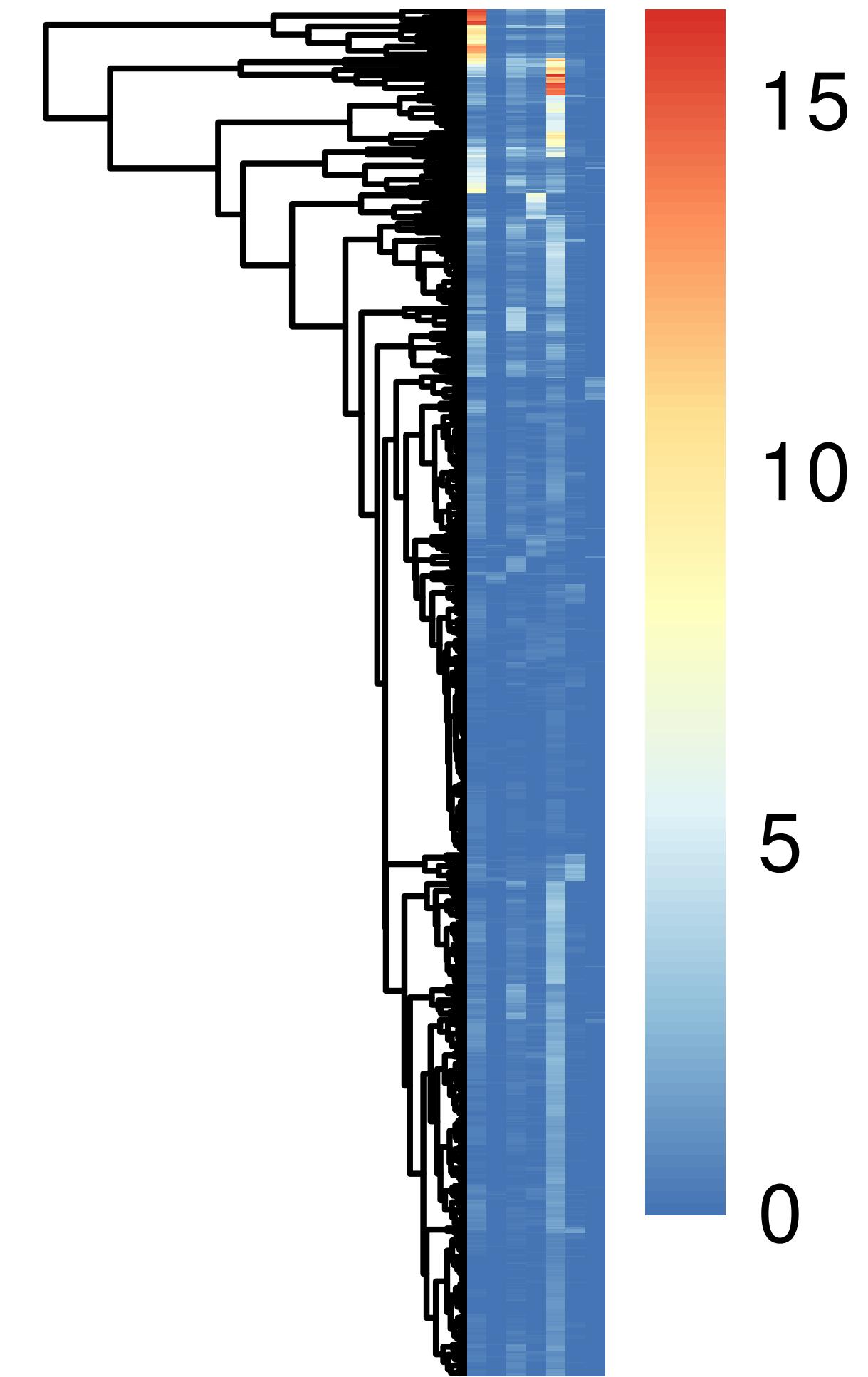
```
## |
```

```
.liger <- rlicher::quantile_norm(.liger)
.liger <- rlicher::runUMAP(.liger)
.liger <- louvainCluster(.liger, resolution = 0.25)
```

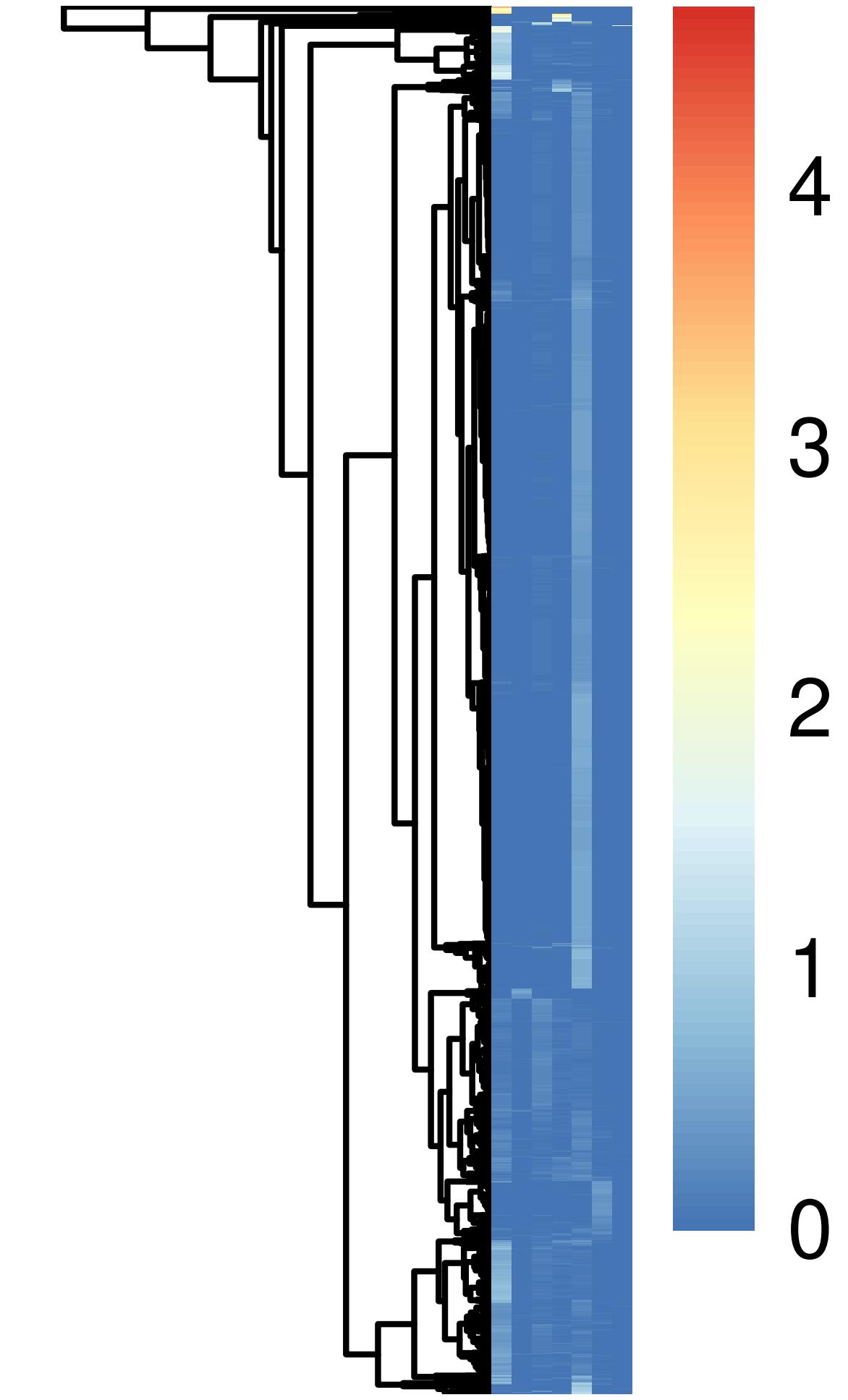
Gao .. Welch, Nature Biotech. (2021)

rlier's NMF result

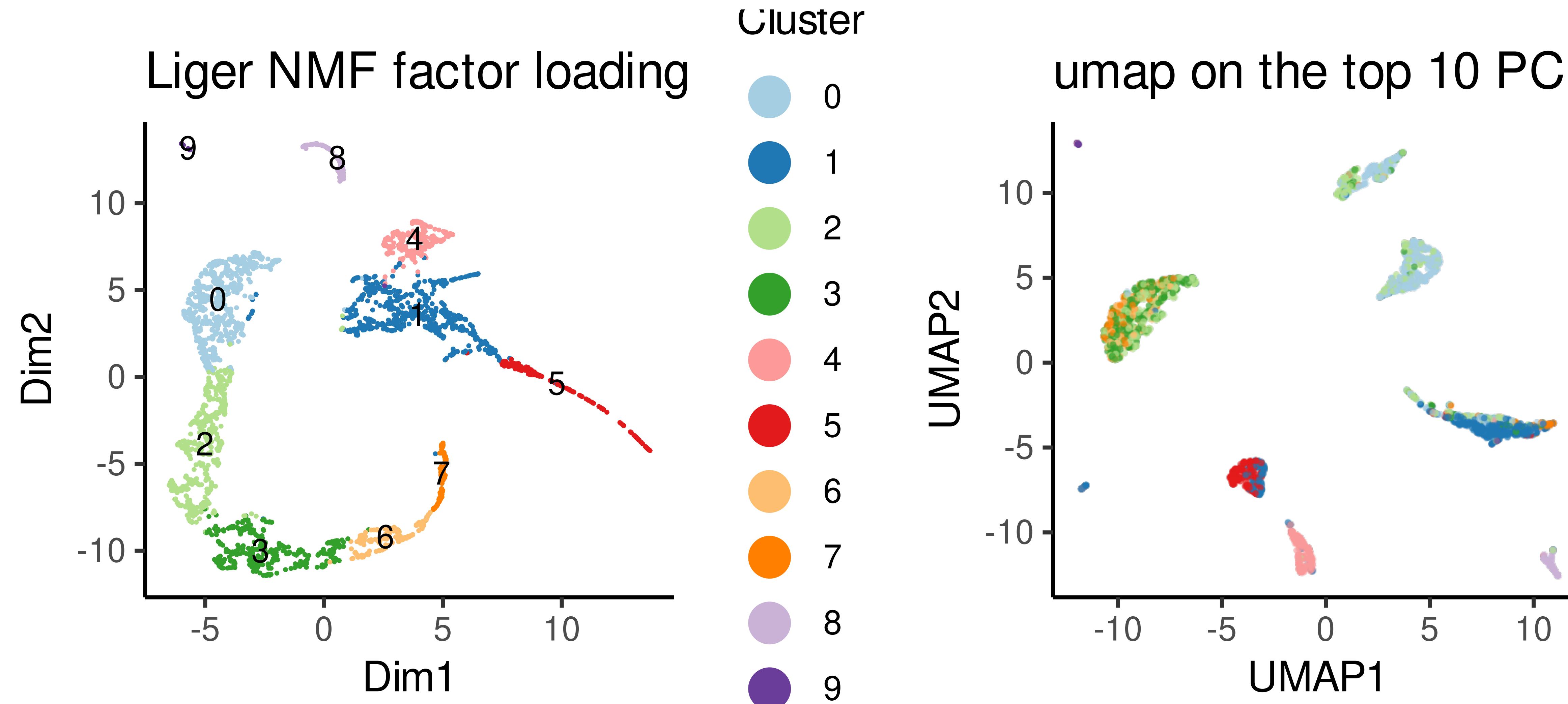
gene × factor matrix



cell × factor matrix

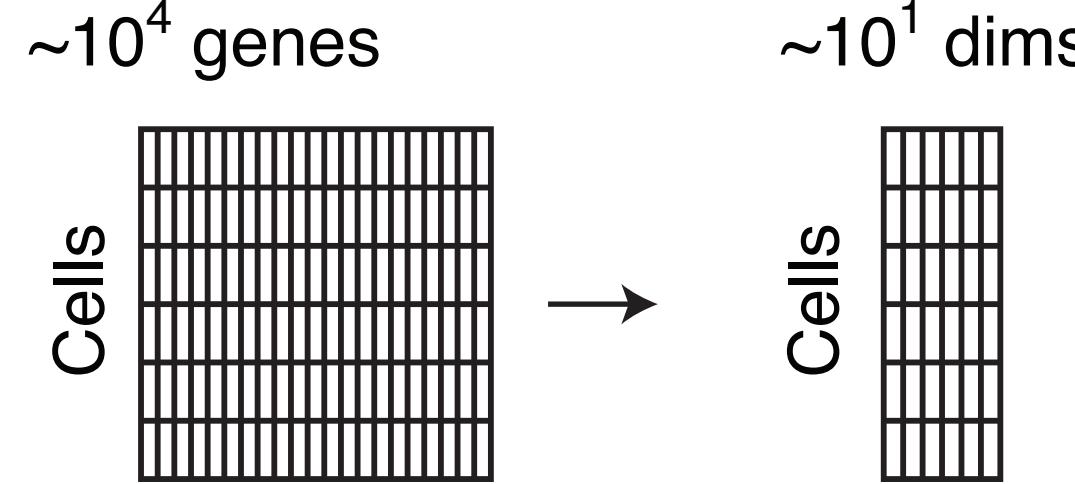


rLiger: clustering cells by the topic/factor loading

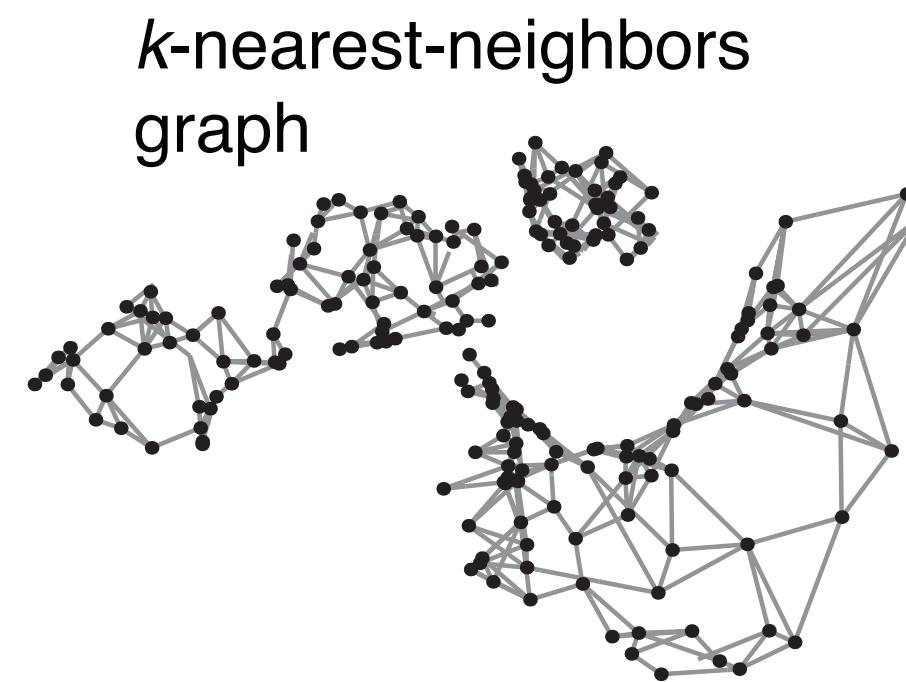


What's next after clustering?

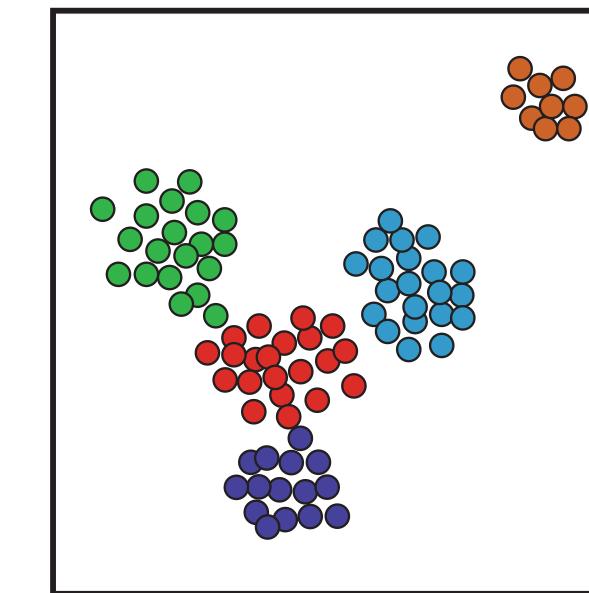
Reduction to a medium-dimensional space



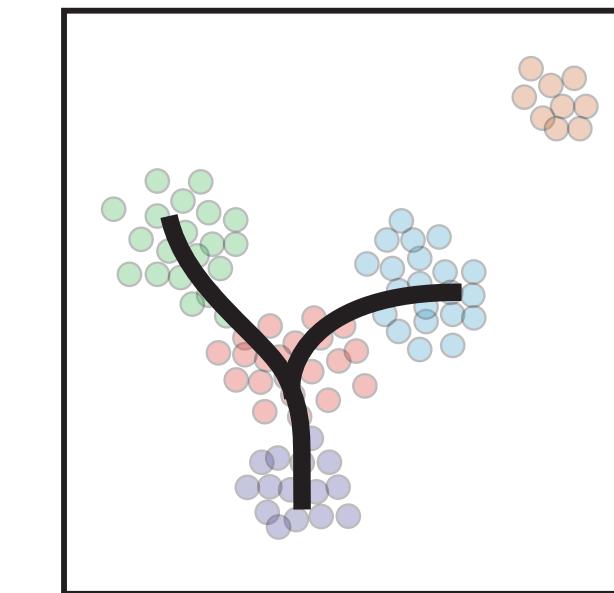
Manifold representation



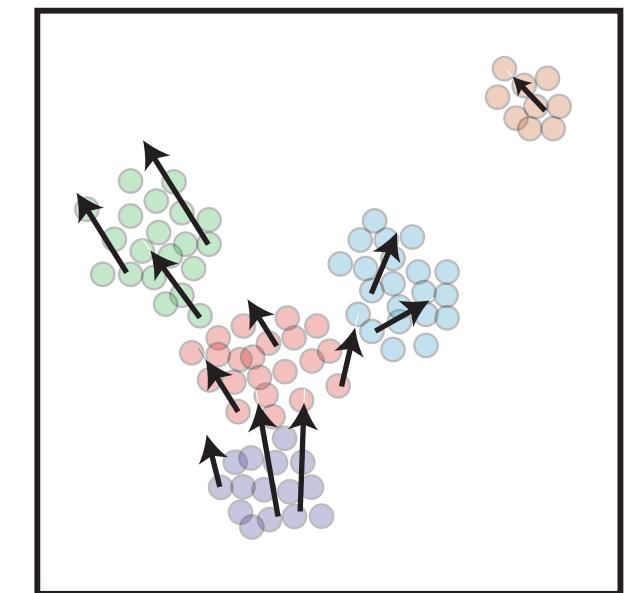
Clustering and differential expression



Trajectories

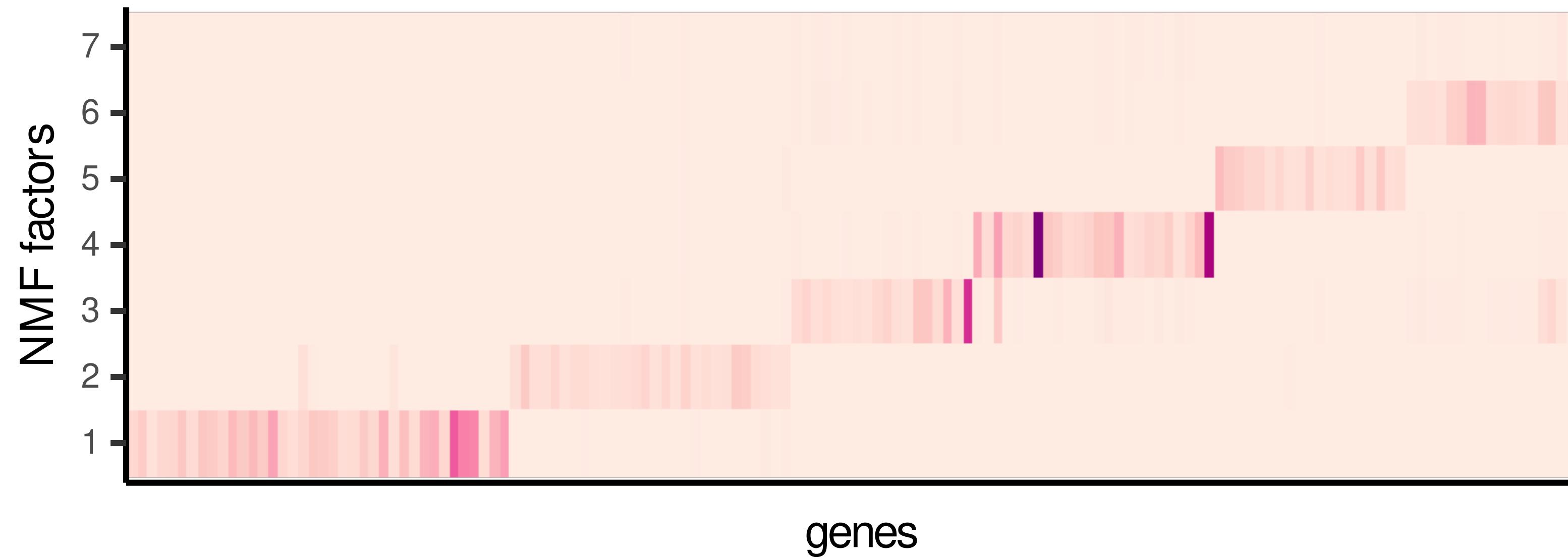


Velocity estimation

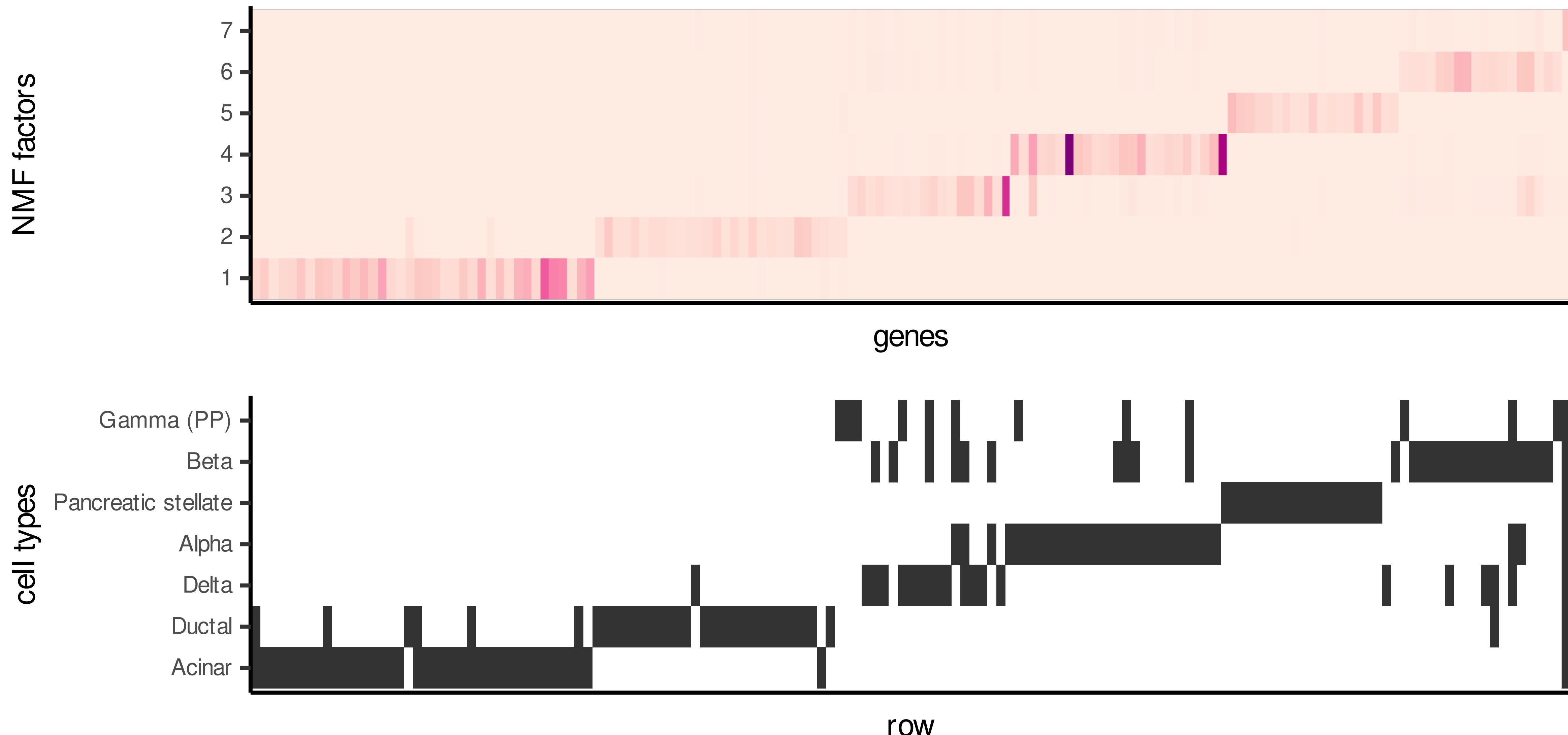


Karchenko, *Nature Methods* (2021)

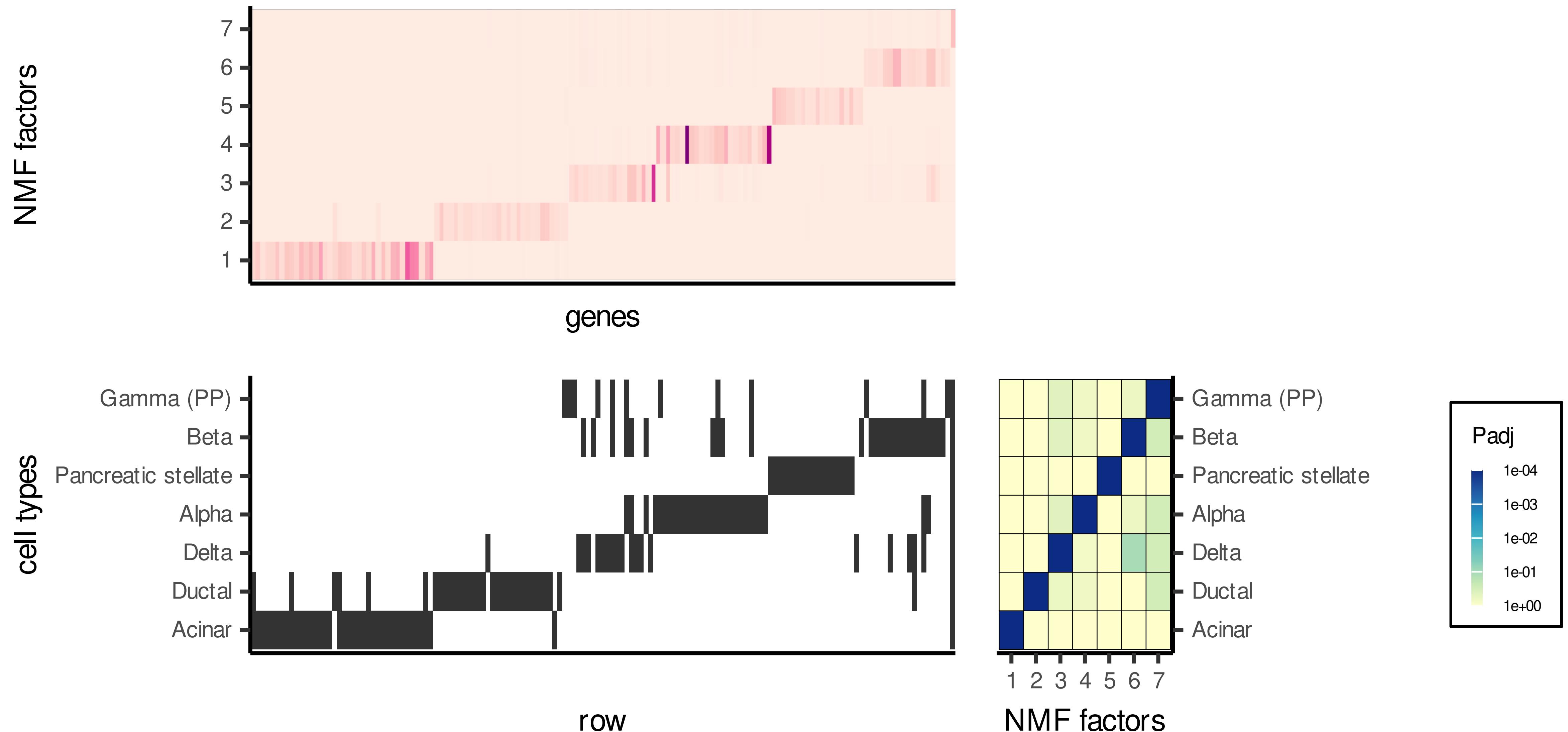
Once we have gene \times factor dictionary



Annotate topics to cell types by enrichment (fgsea)



Annotate topics to cell types by enrichment (fgsea)

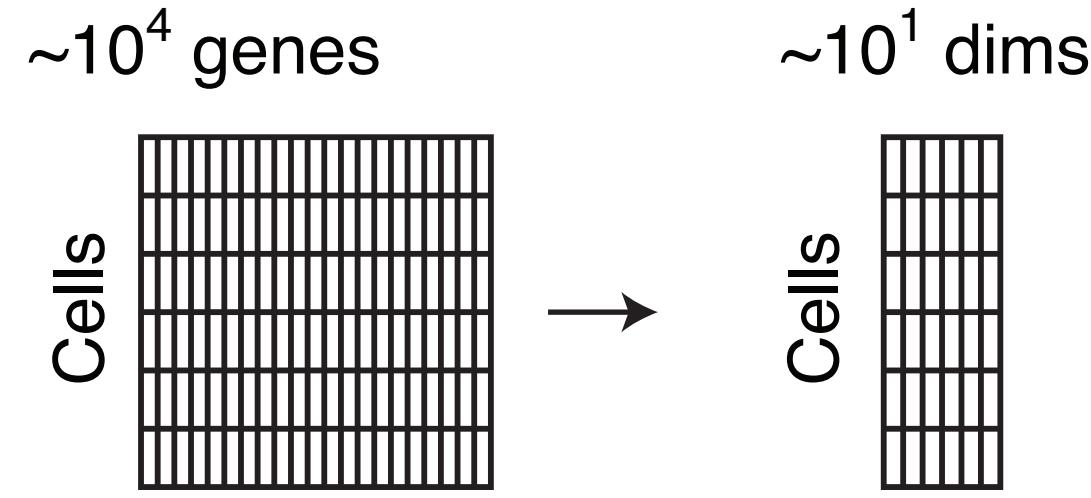


Today's lecture

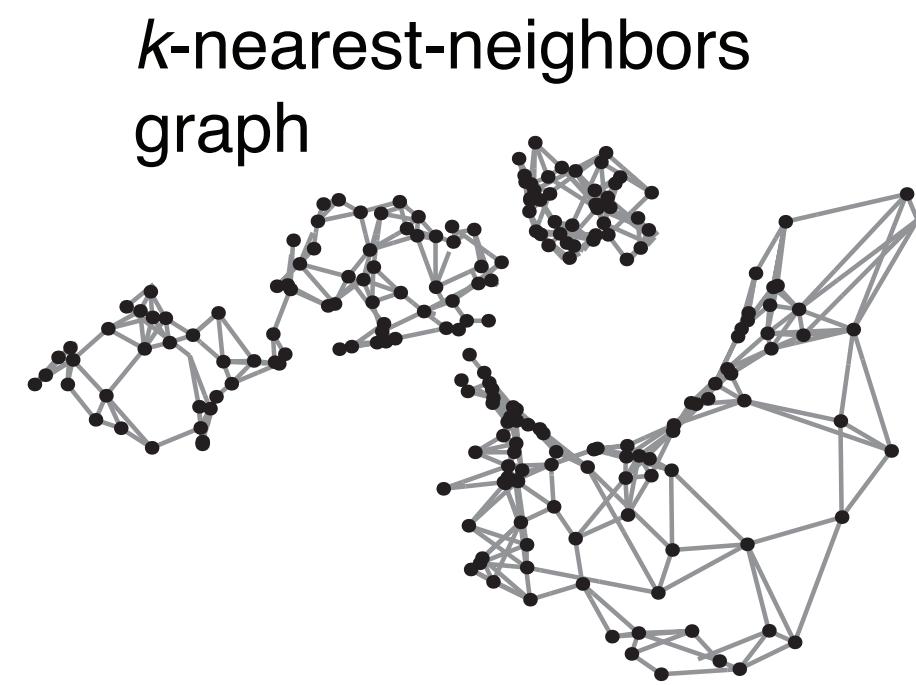
- 1 (review) Principal Component Analysis
- 2 Non-negative matrix factorization
- 3 Differential gene expression analysis

What's next after clustering?

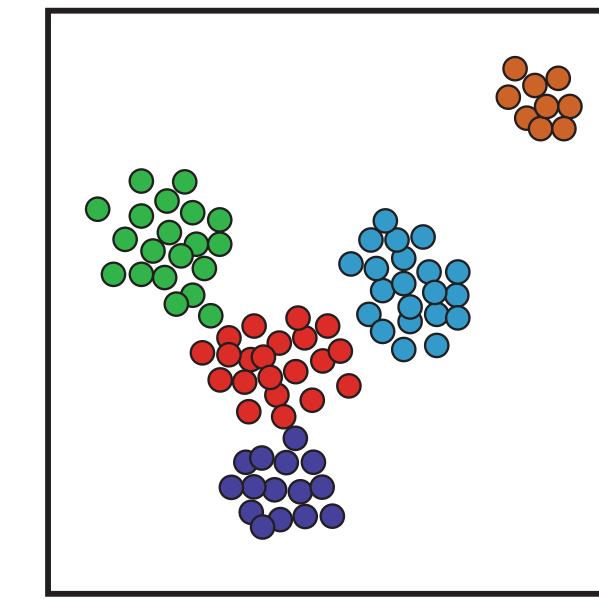
Reduction to a medium-dimensional space



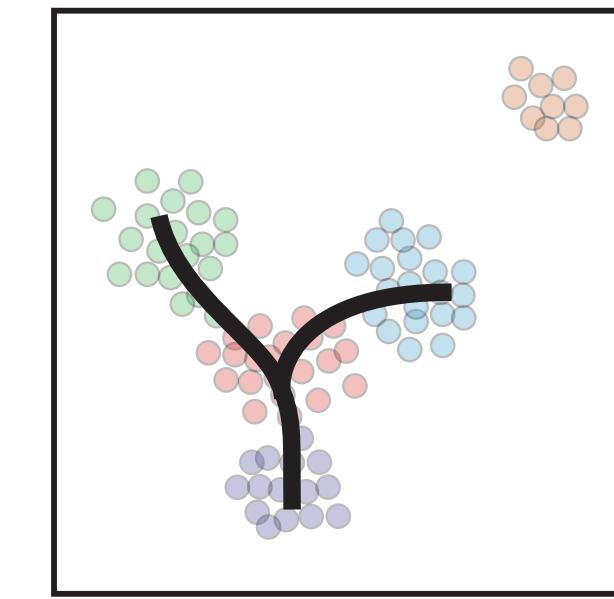
Manifold representation



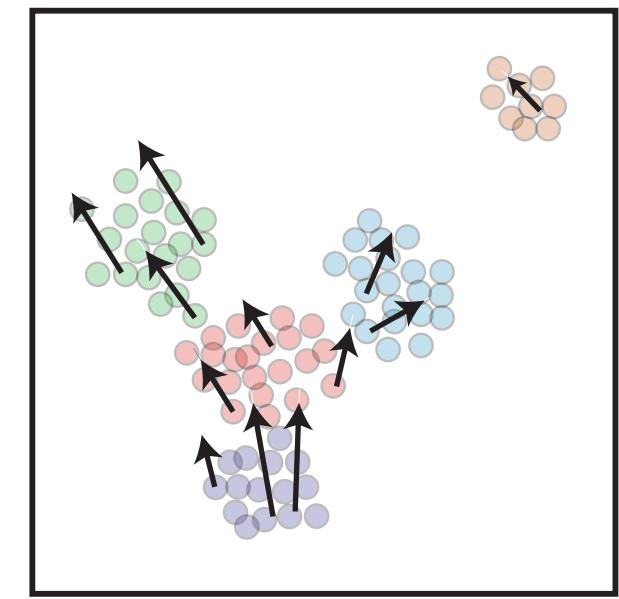
Clustering and differential expression



Trajectories



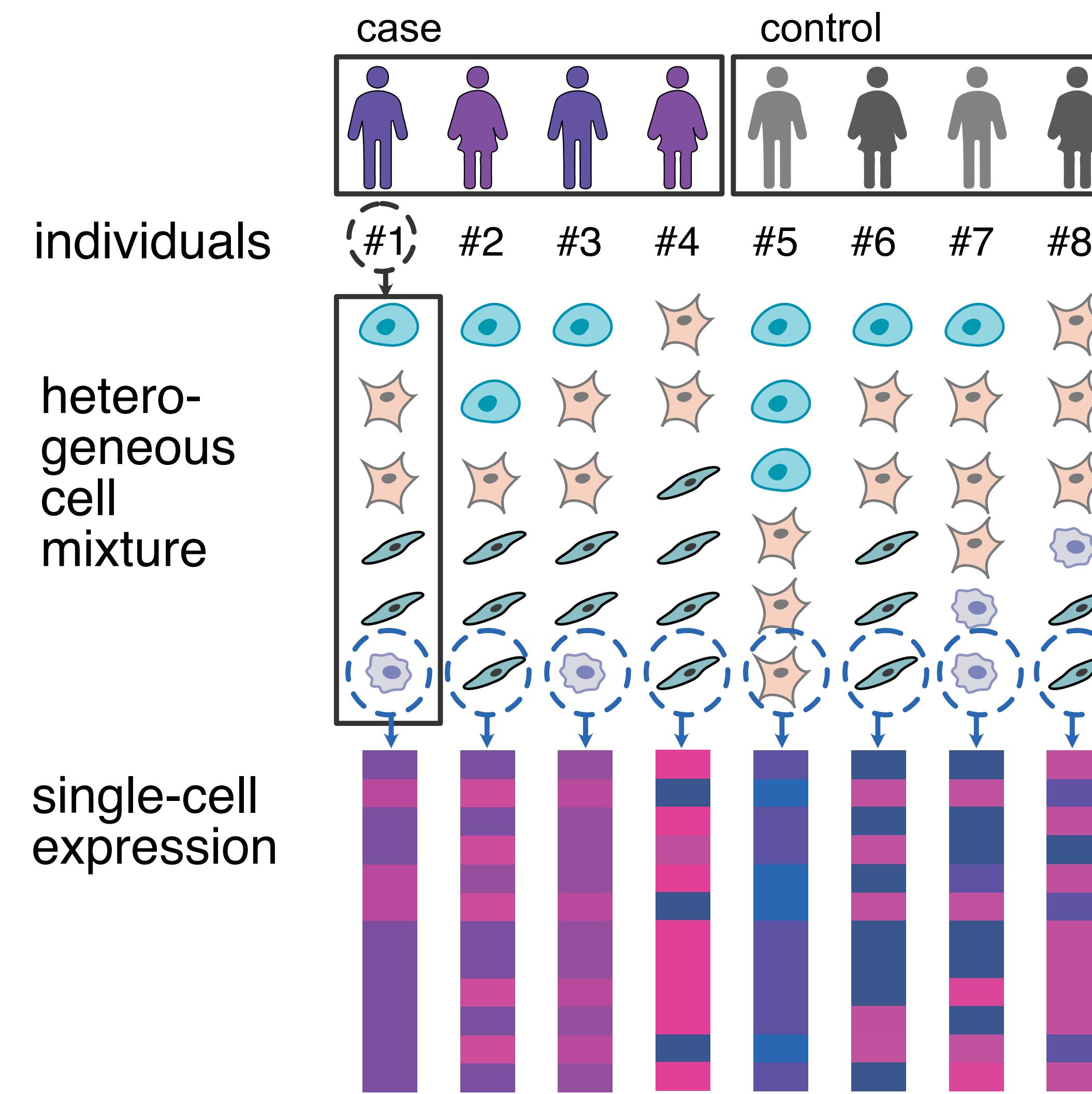
Velocity estimation



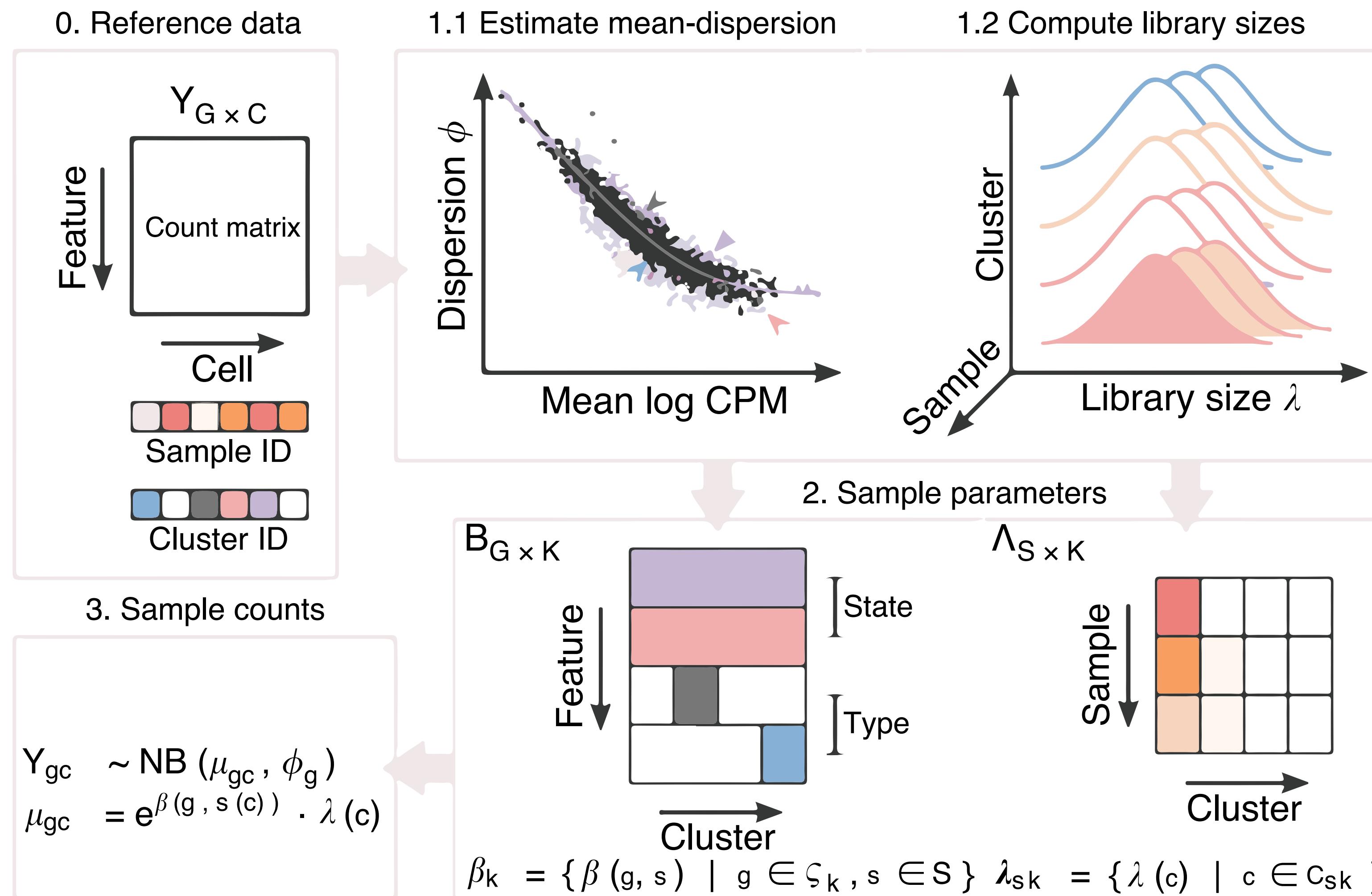
After cell type identification/clustering, we can dissect disease mechanisms

Karchenko, *Nature Methods* (2021)

Multi-individual single-cell data analysis

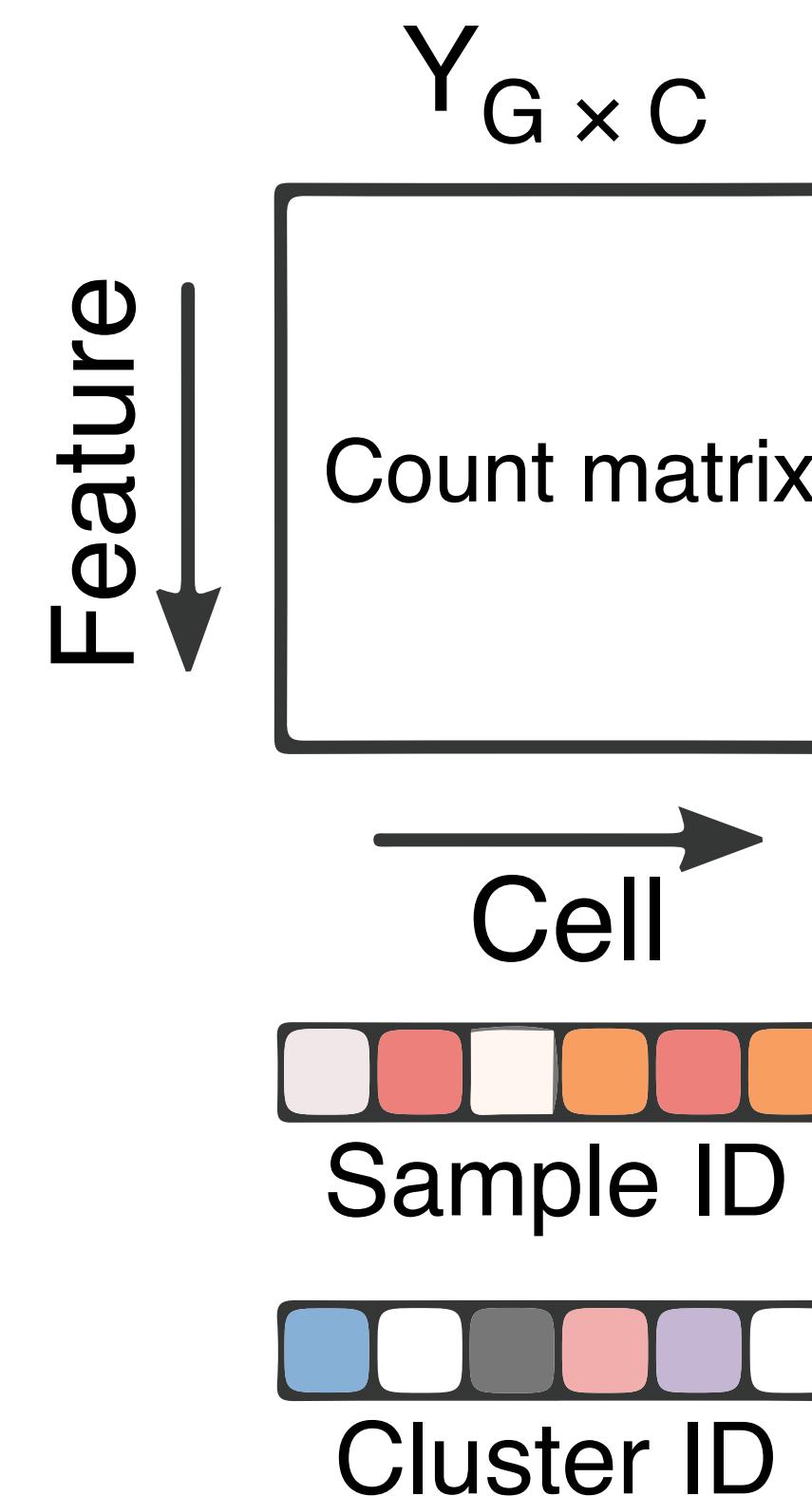


What is the best strategy to identify disease-specific genes from multi-individual studies?

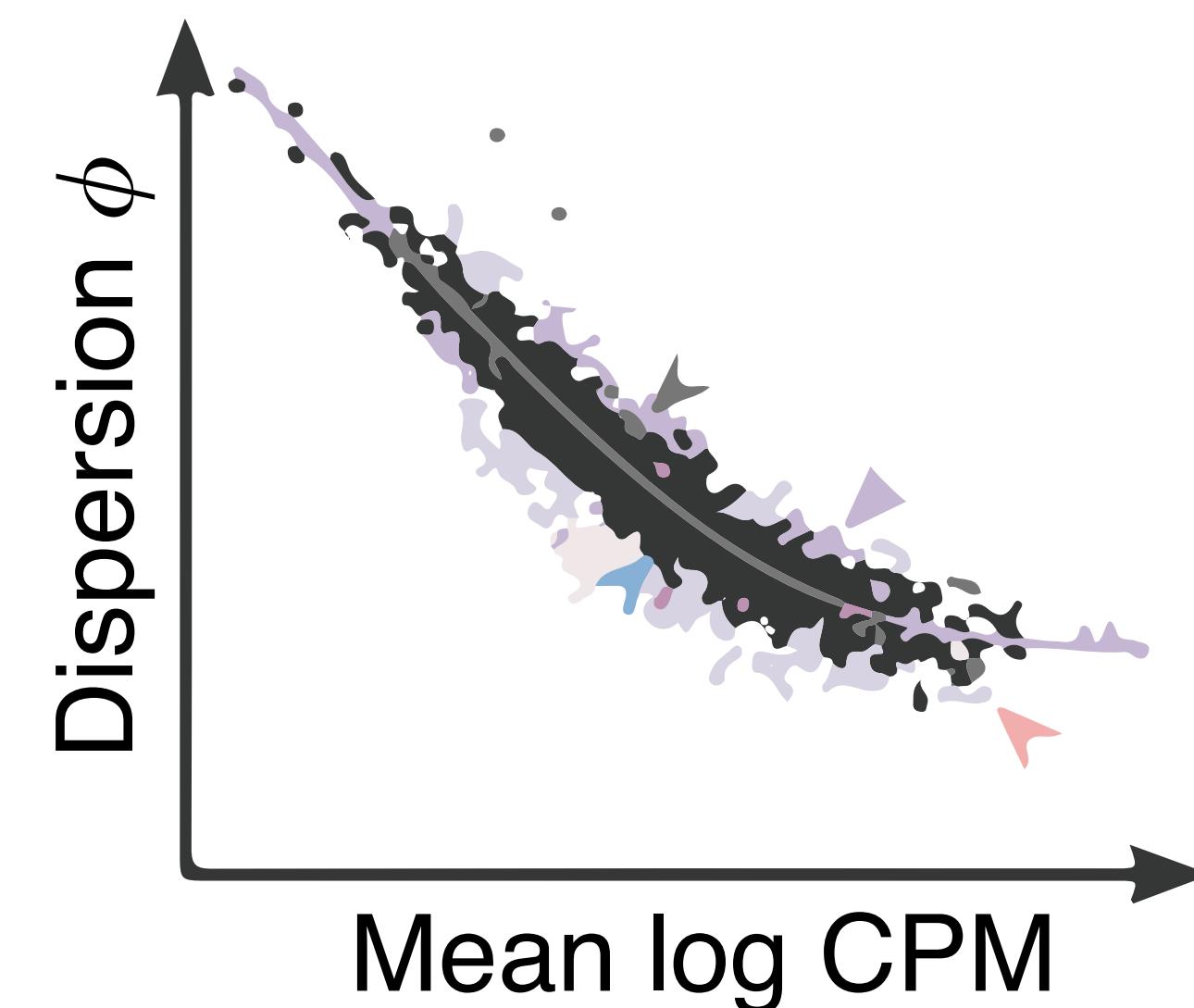


Learn parameters from reference data

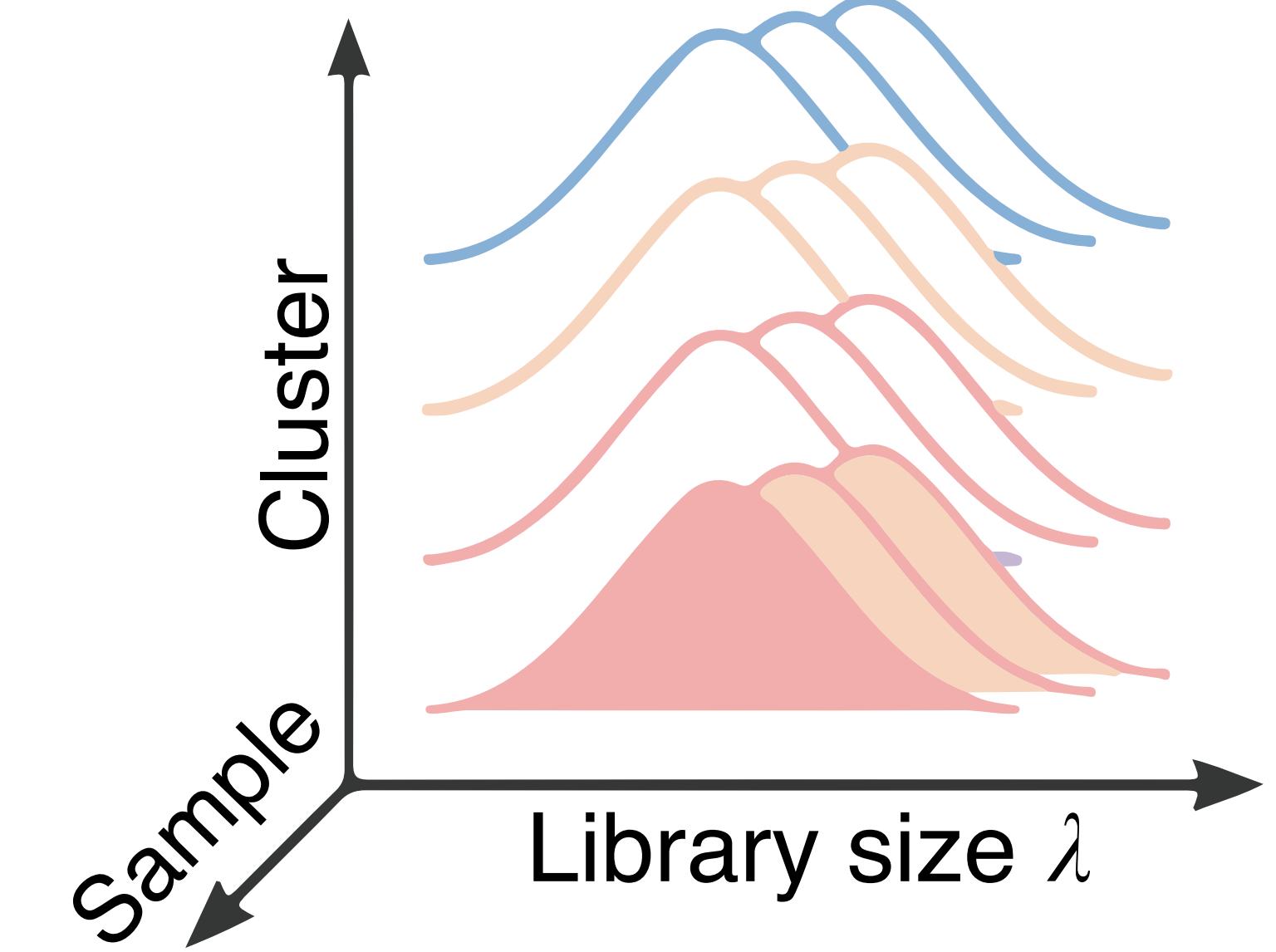
0. Reference data



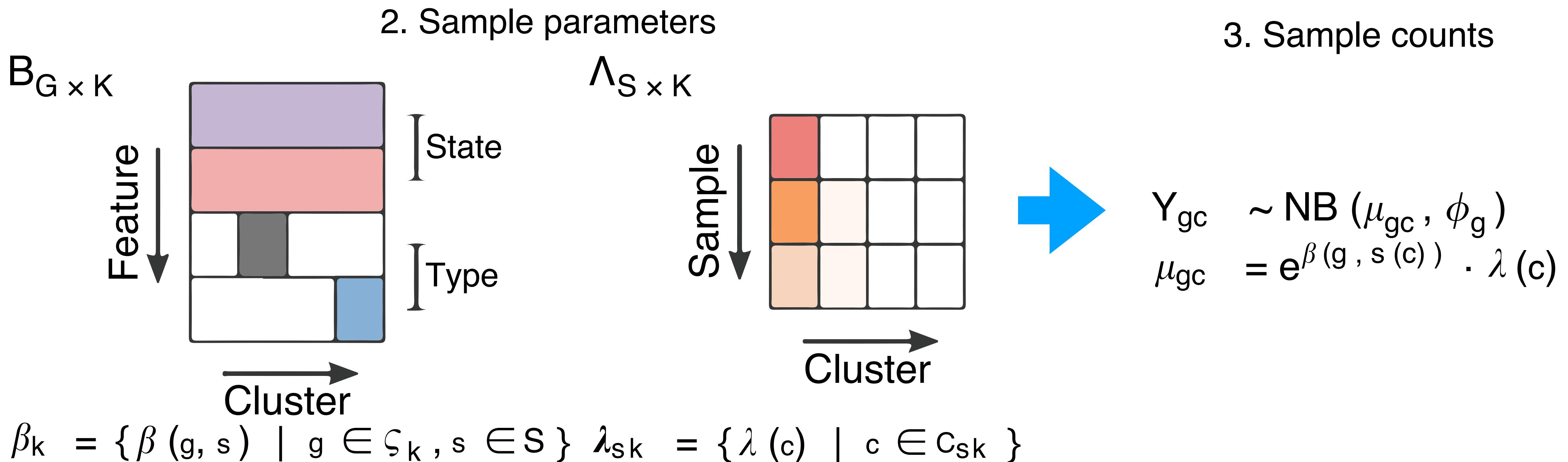
1.1 Estimate mean-dispersion



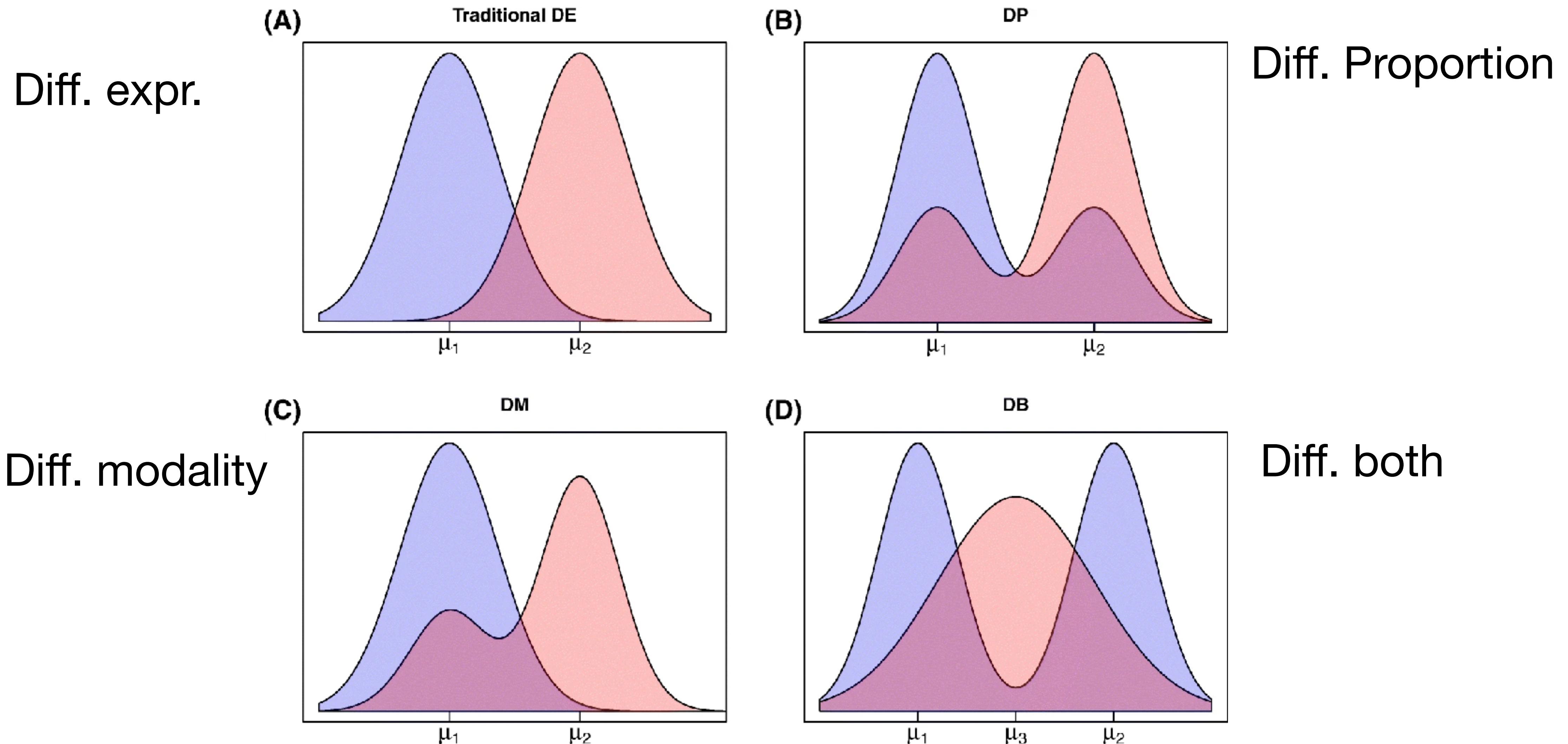
1.2 Compute library sizes



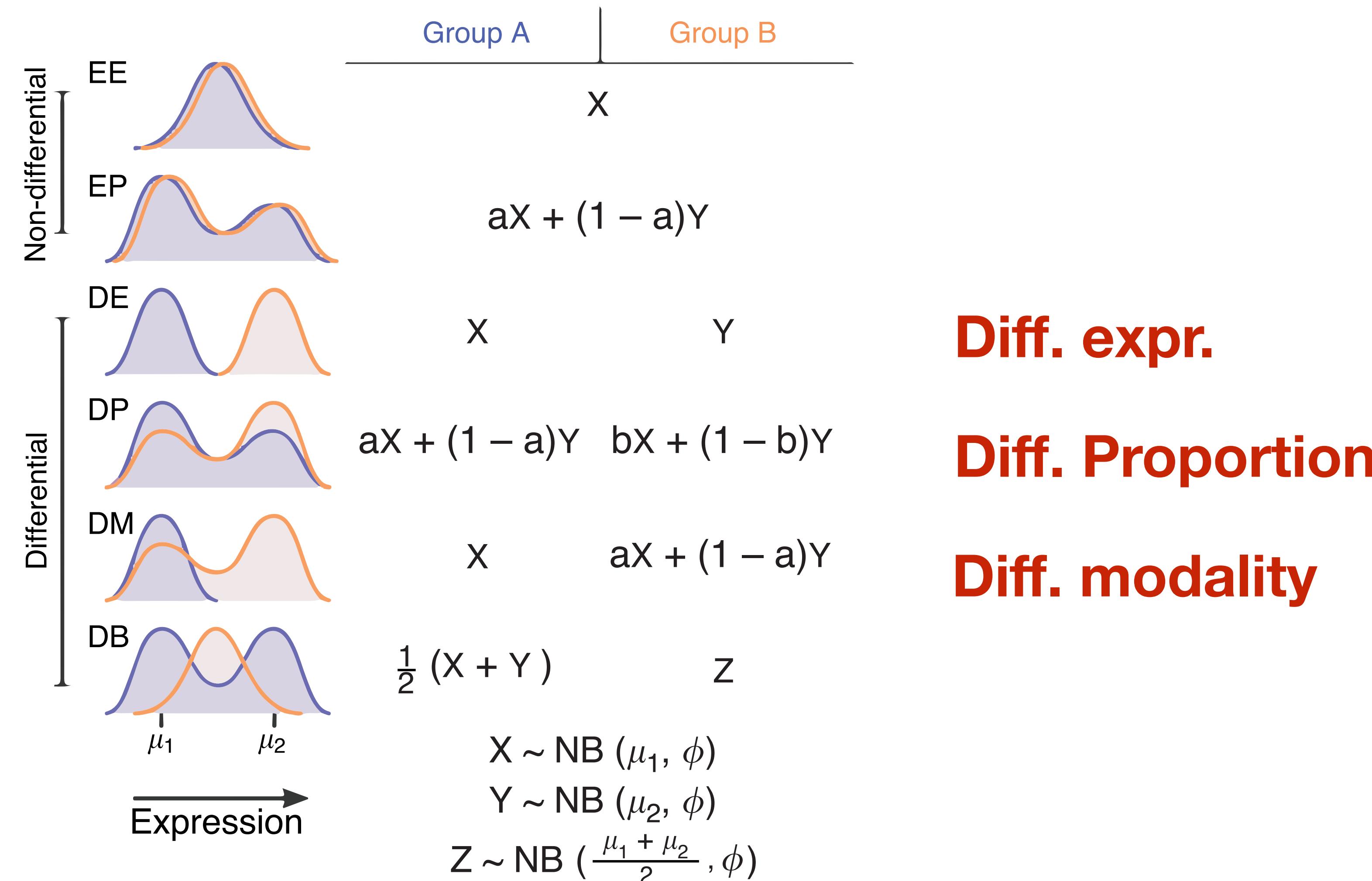
Generate single-cell data based on the realistic data-specific parameters



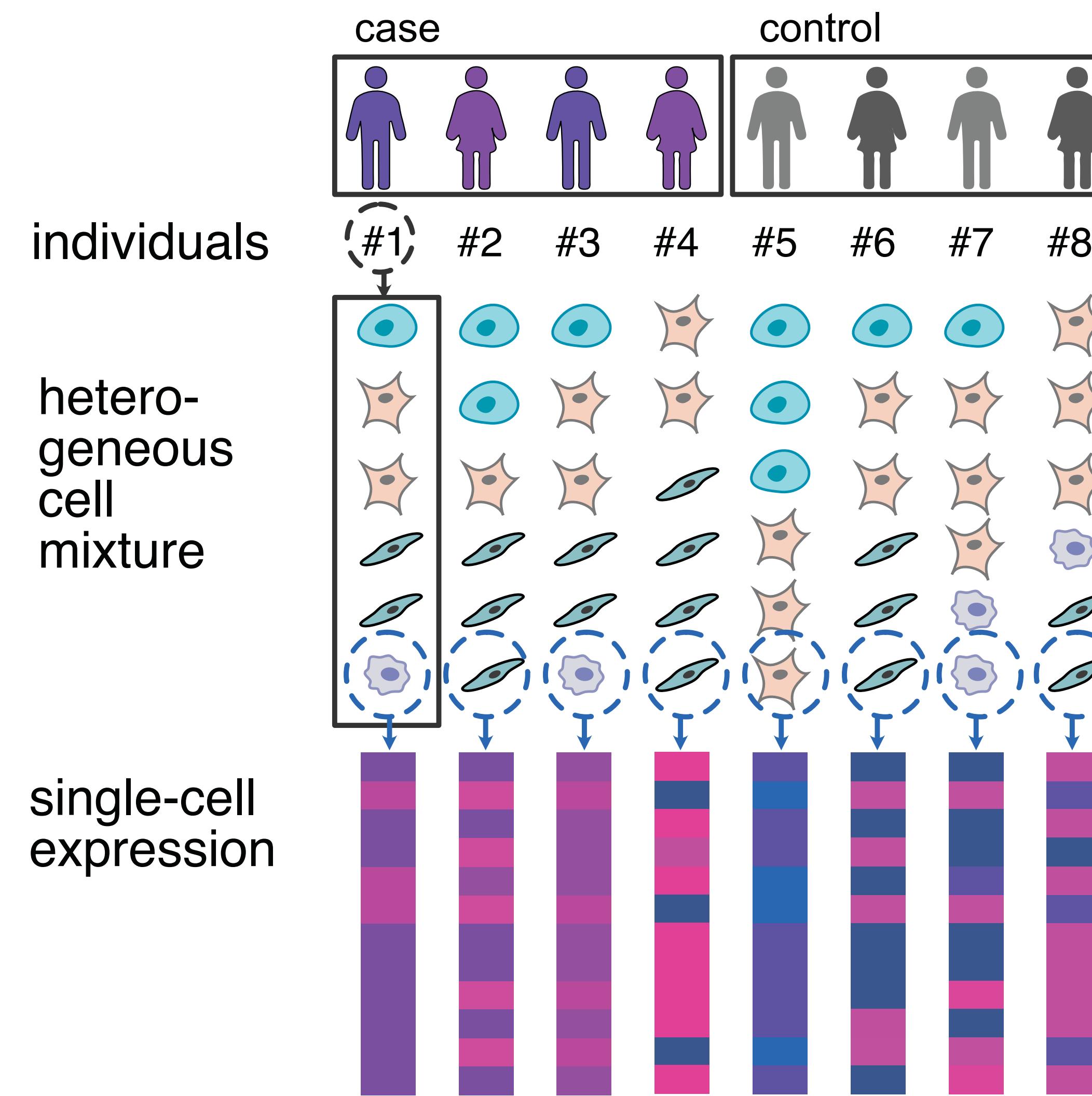
What are the possibilities of "difference" between cells derived from the case and control samples?



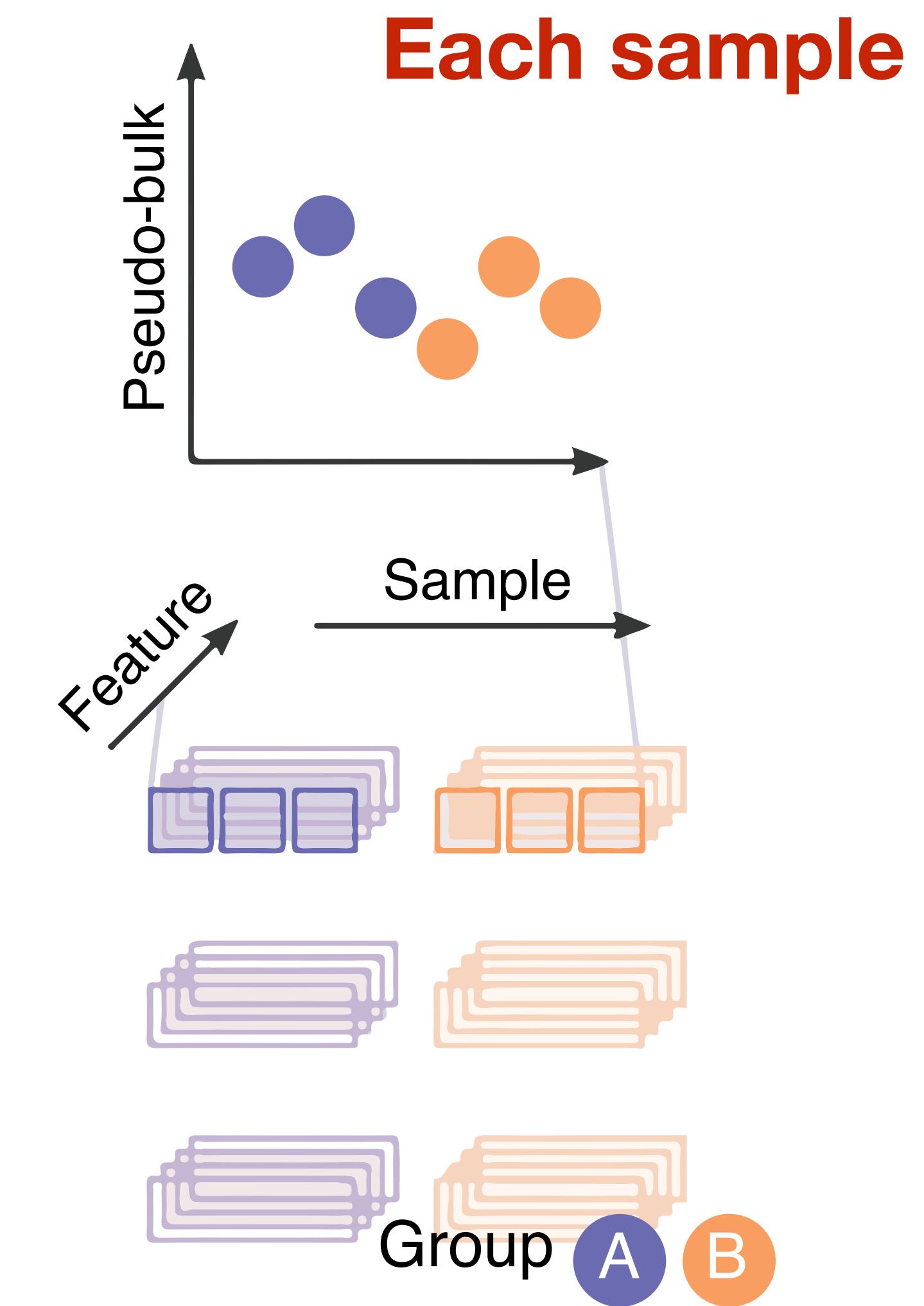
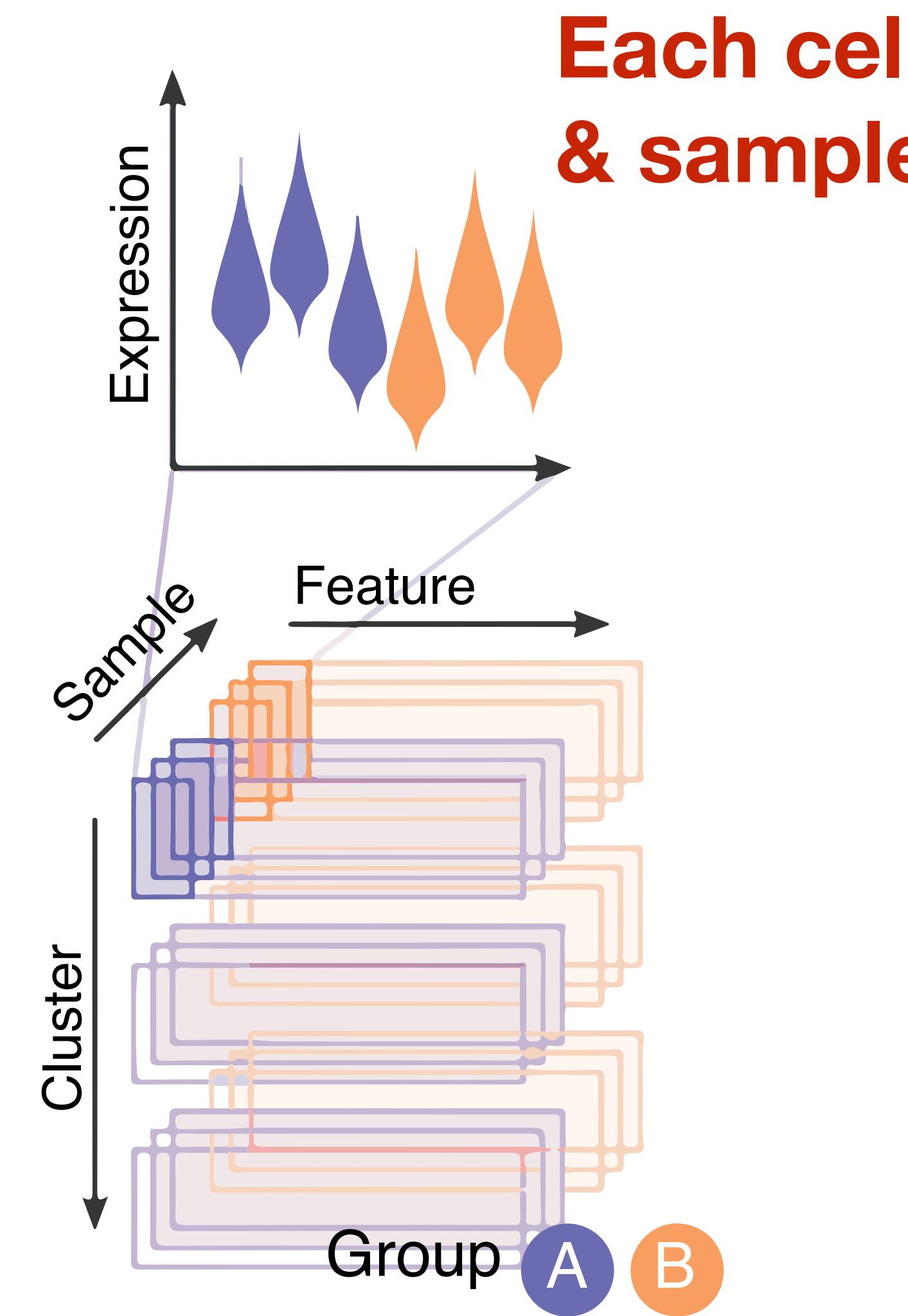
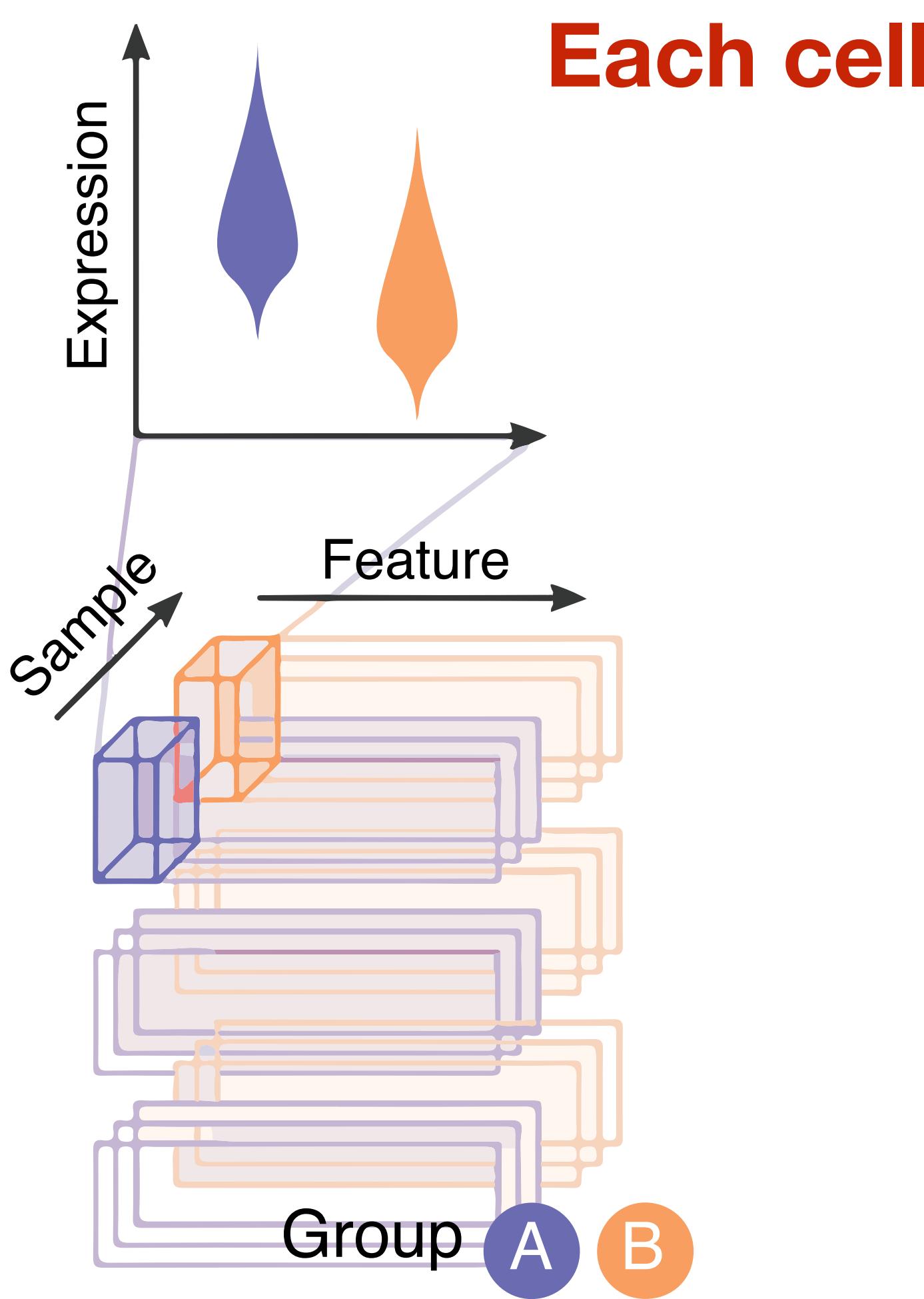
What are the possibilities of "difference" between cells derived from the case and control samples?



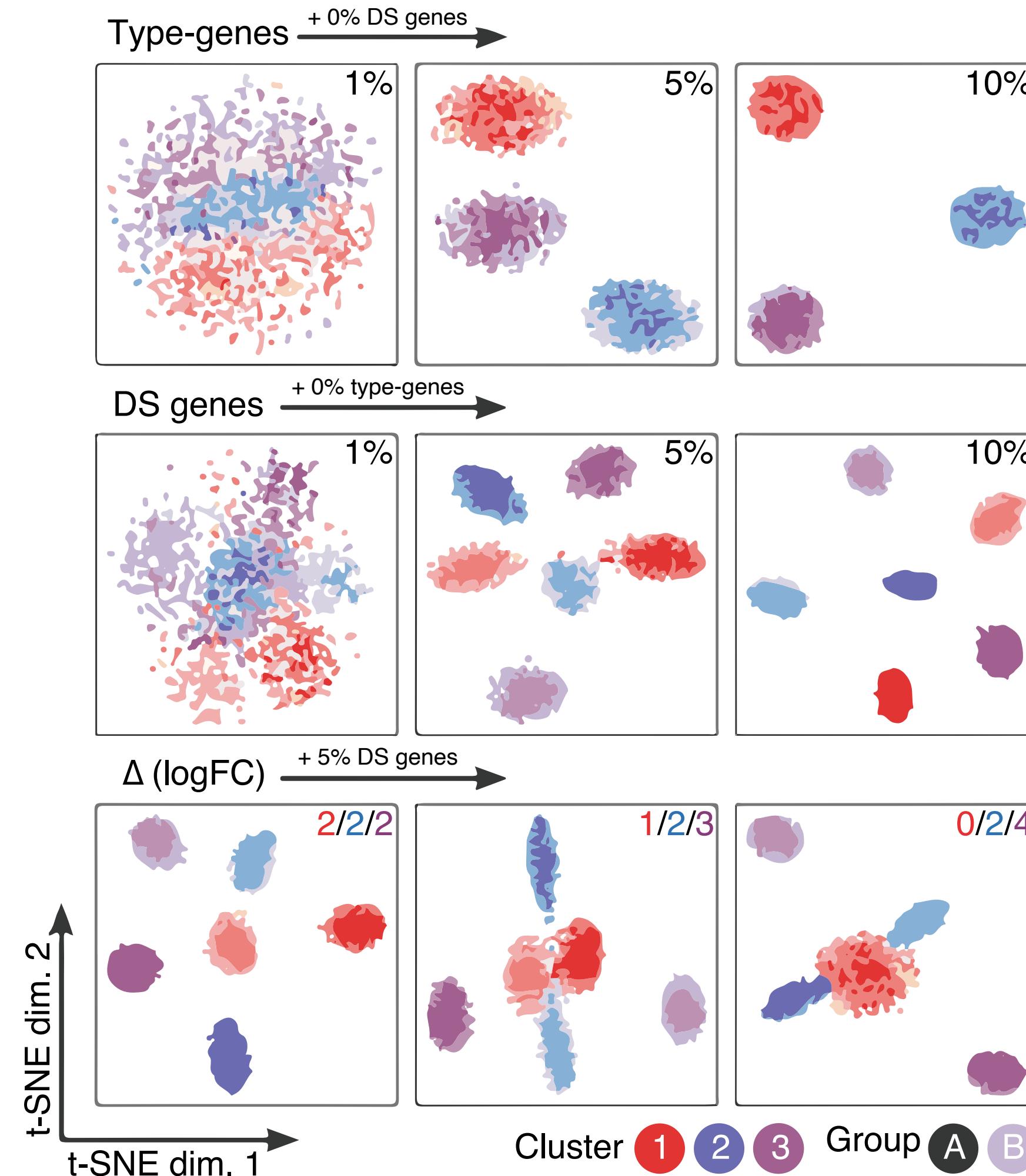
Multi-individual single-cell data analysis



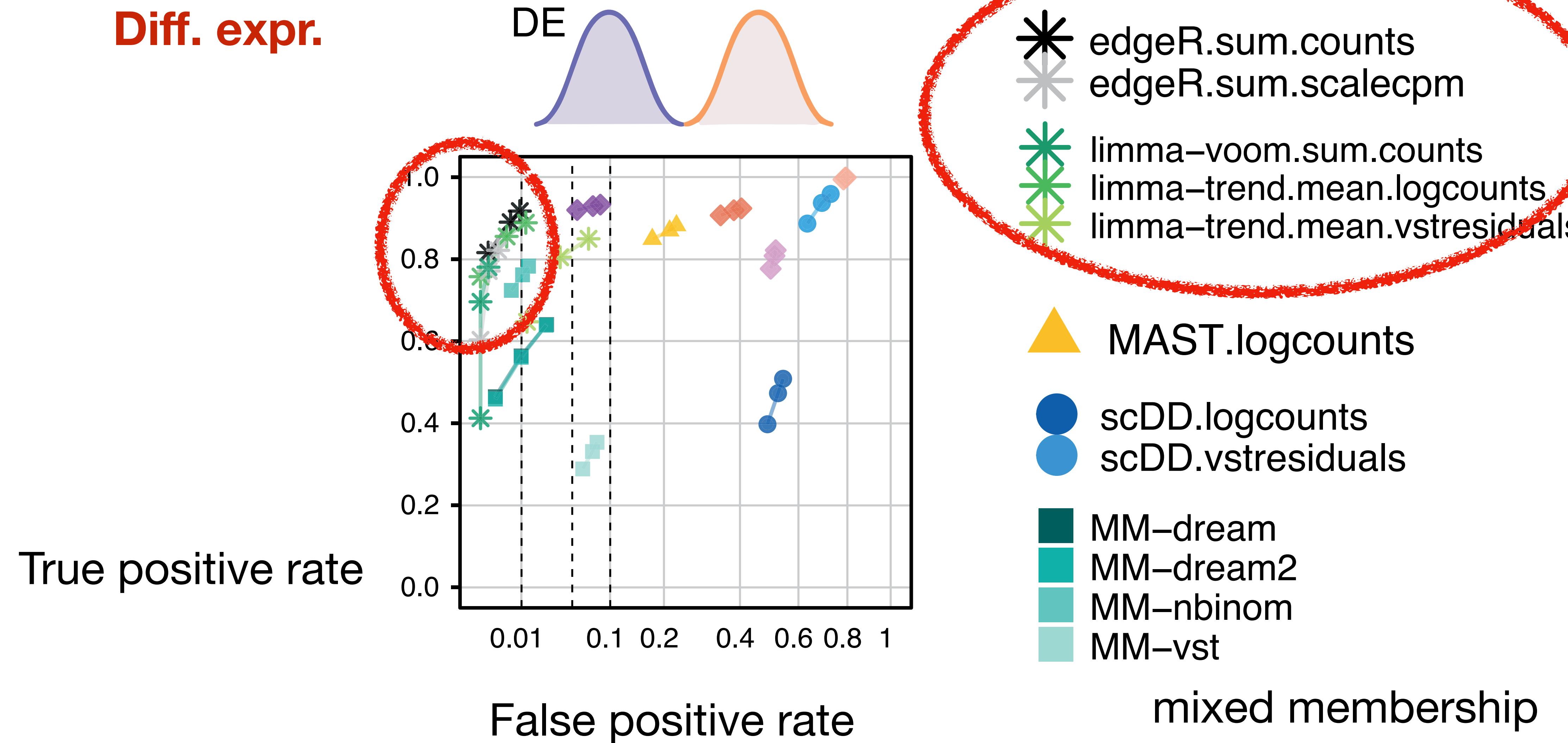
Multiple ways to define a single data point



Genes may implicate cell type identity & disease



Benchmark: Pseudo-bulk approaches work best



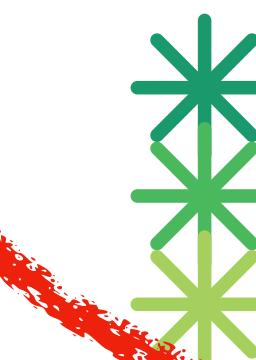
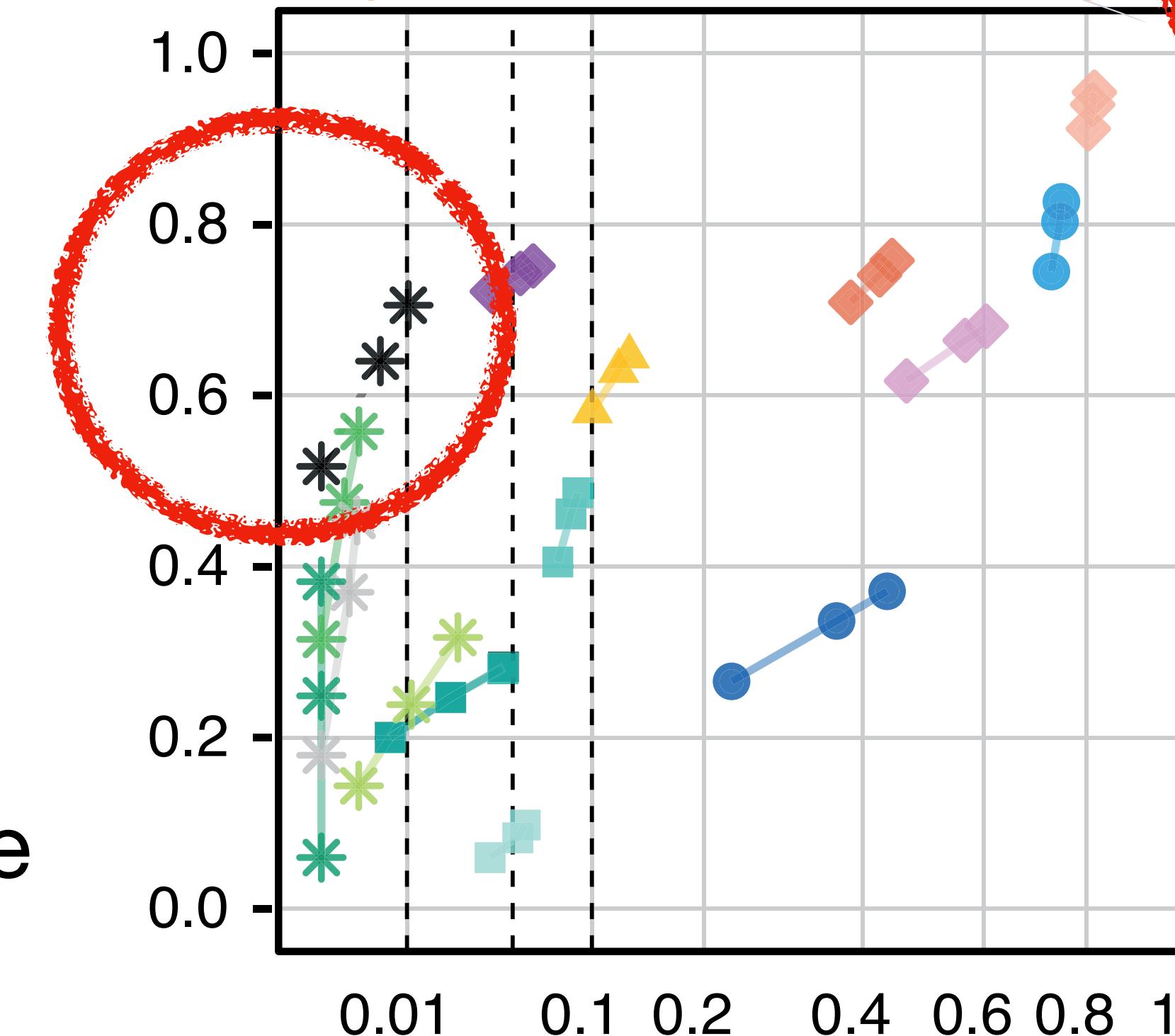
Pseudo-bulk approaches: high power, low FDR

Diff. Proportion

True positive rate

False positive rate

DP



edgeR.sum.counts
edgeR.sum.scalecpm

limma-voom.sum.counts
limma-trend.mean.logcounts
limma-trend.mean.vstresiduals

MAST.logcounts

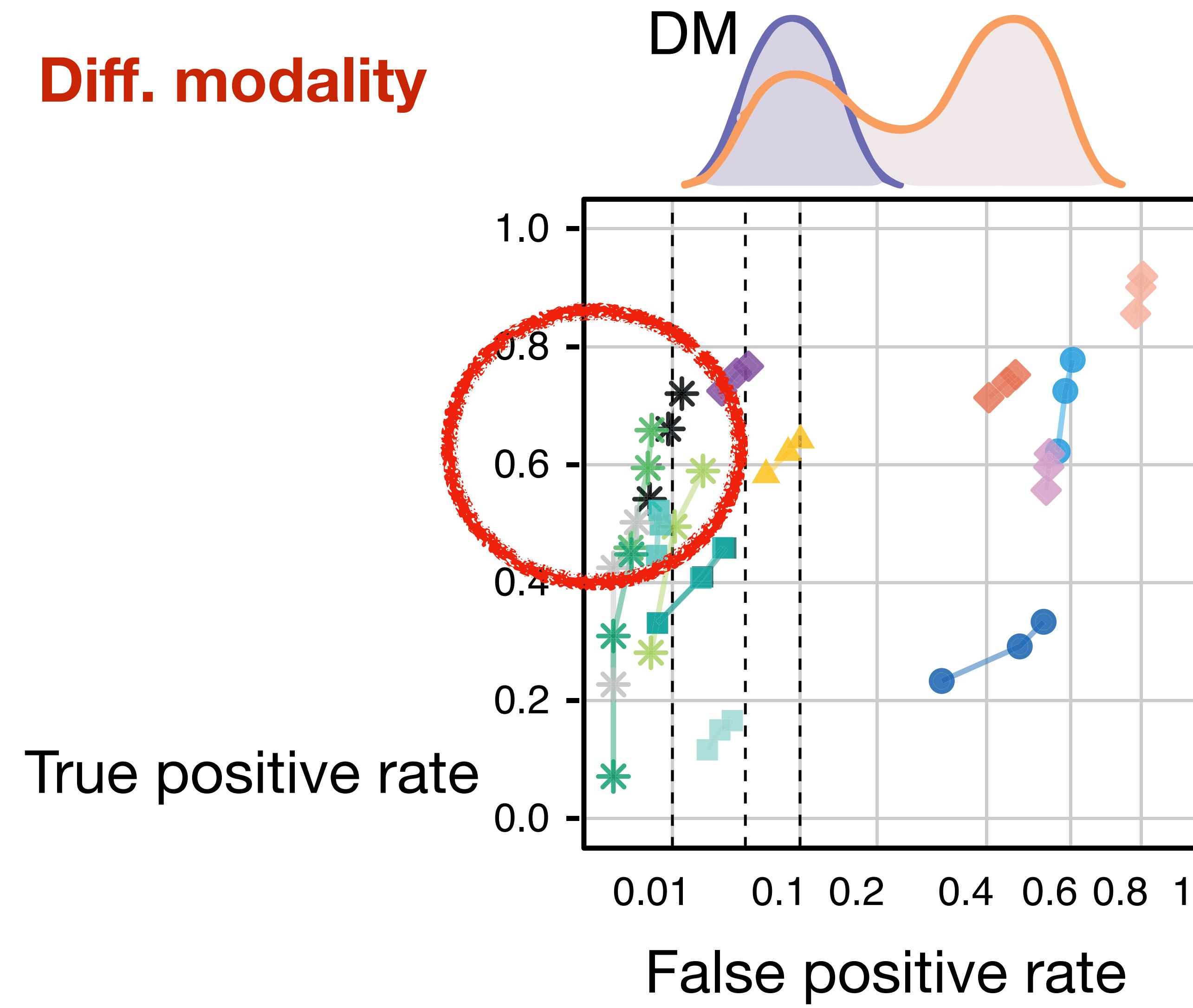
scDD.logcounts
scDD.vstresiduals

MM-dream
MM-dream2
MM-nbinom
MM-vst

mixed membership

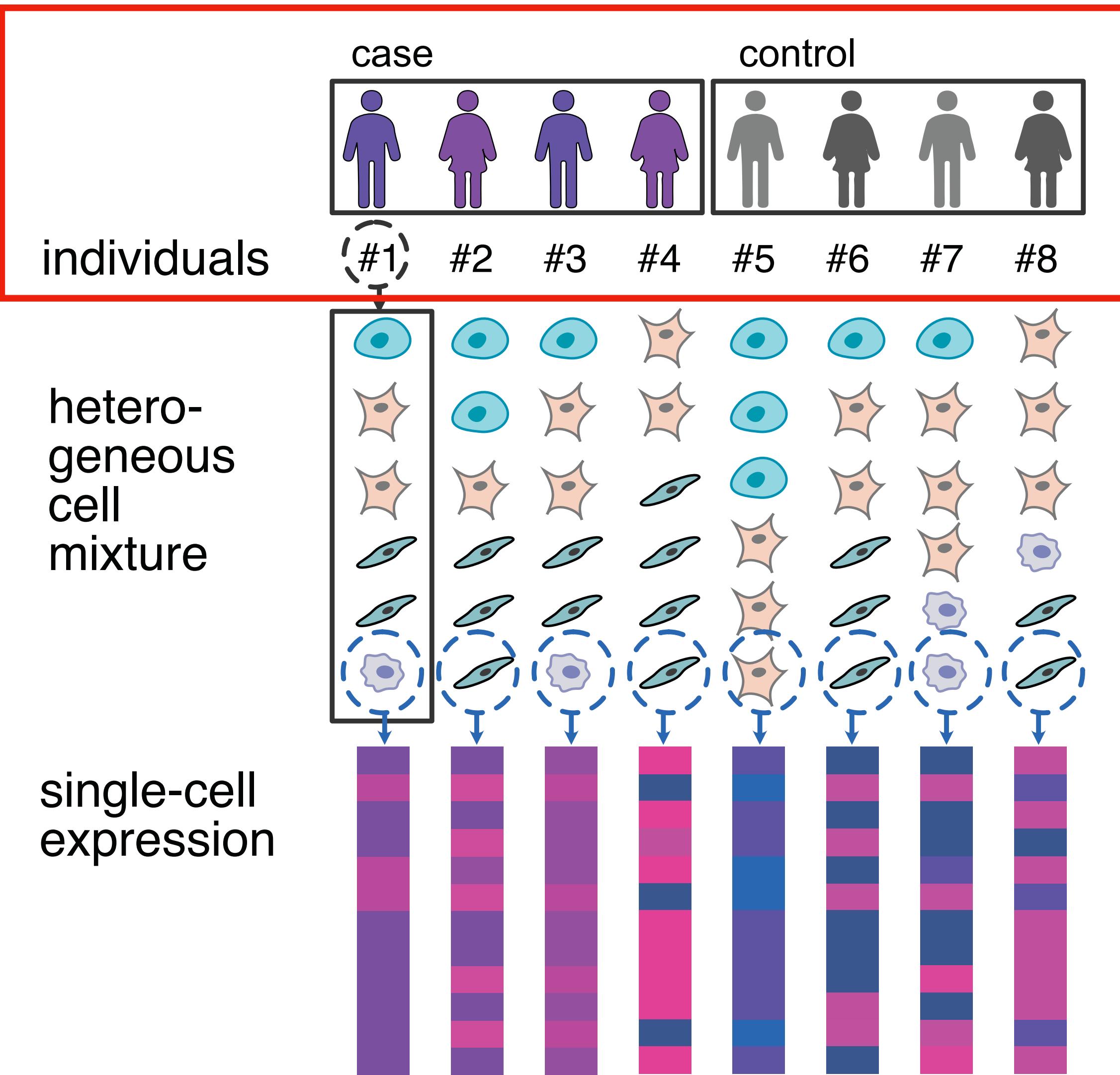
Pseudo-bulk approaches: high power, low FDR

Diff. modality

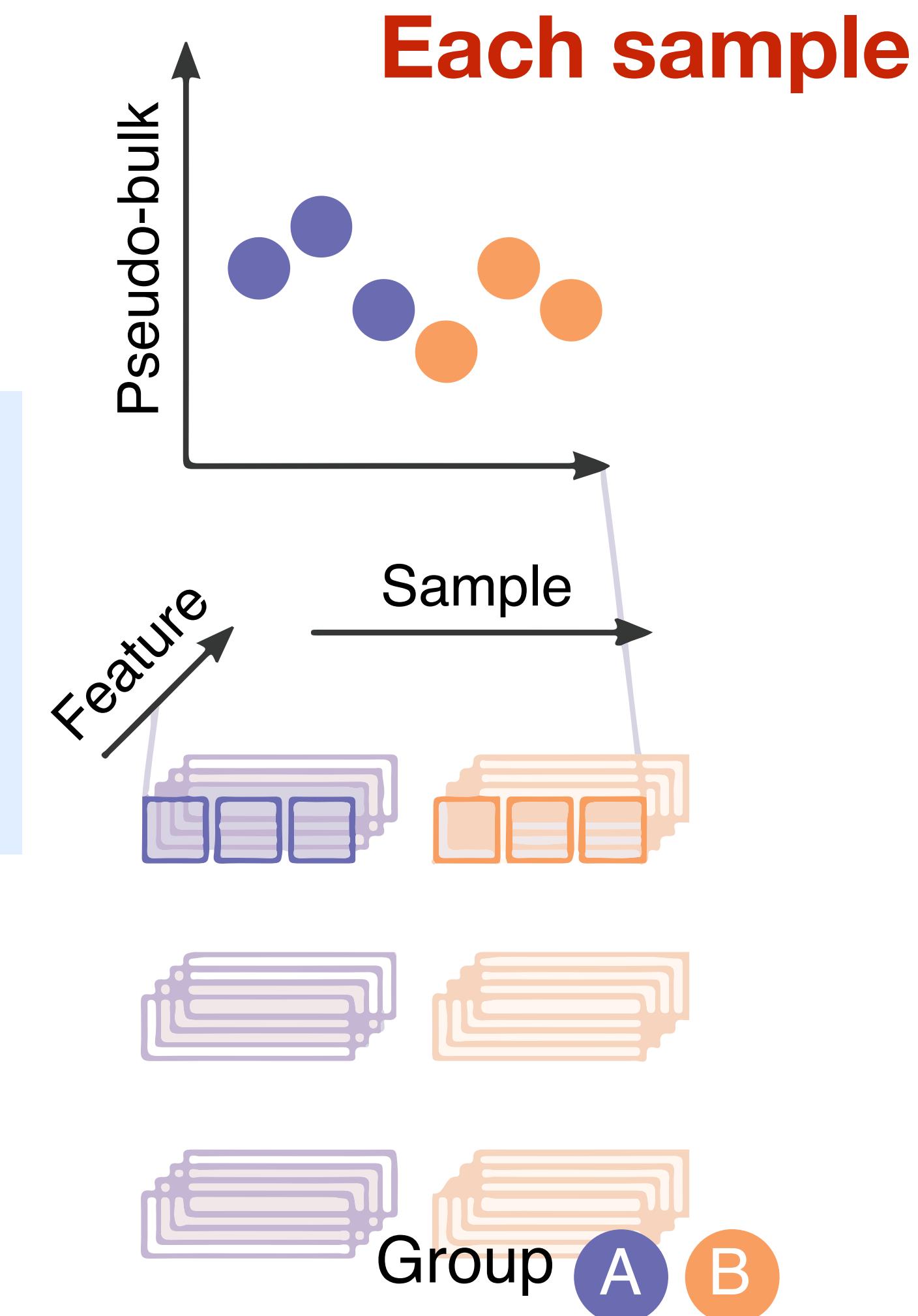


- * edgeR.sum.counts
edgeR.sum.scalecpm
- * limma-voom.sum.counts
limma-trend.mean.logcounts
limma-trend.mean.vstresiduals
- △ MAST.logcounts
- scDD.logcounts
scDD.vstresiduals
- MM-dream
MM-dream2
MM-nbinom
MM-vst
- mixed membership

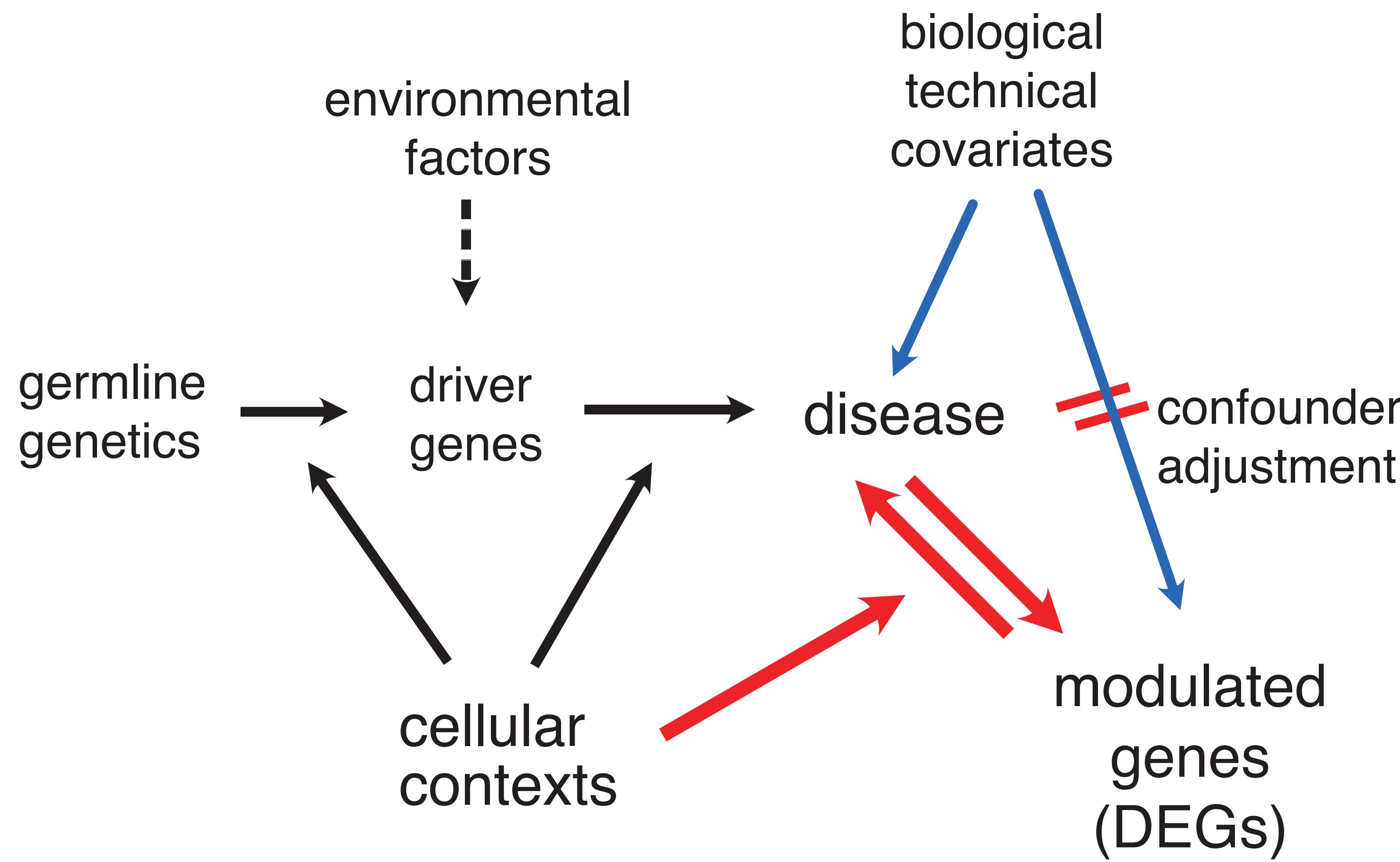
Each sample/individual within each cell type is a proper unit for DE analysis



How should we aggregate cells into pseudo-bulk samples?

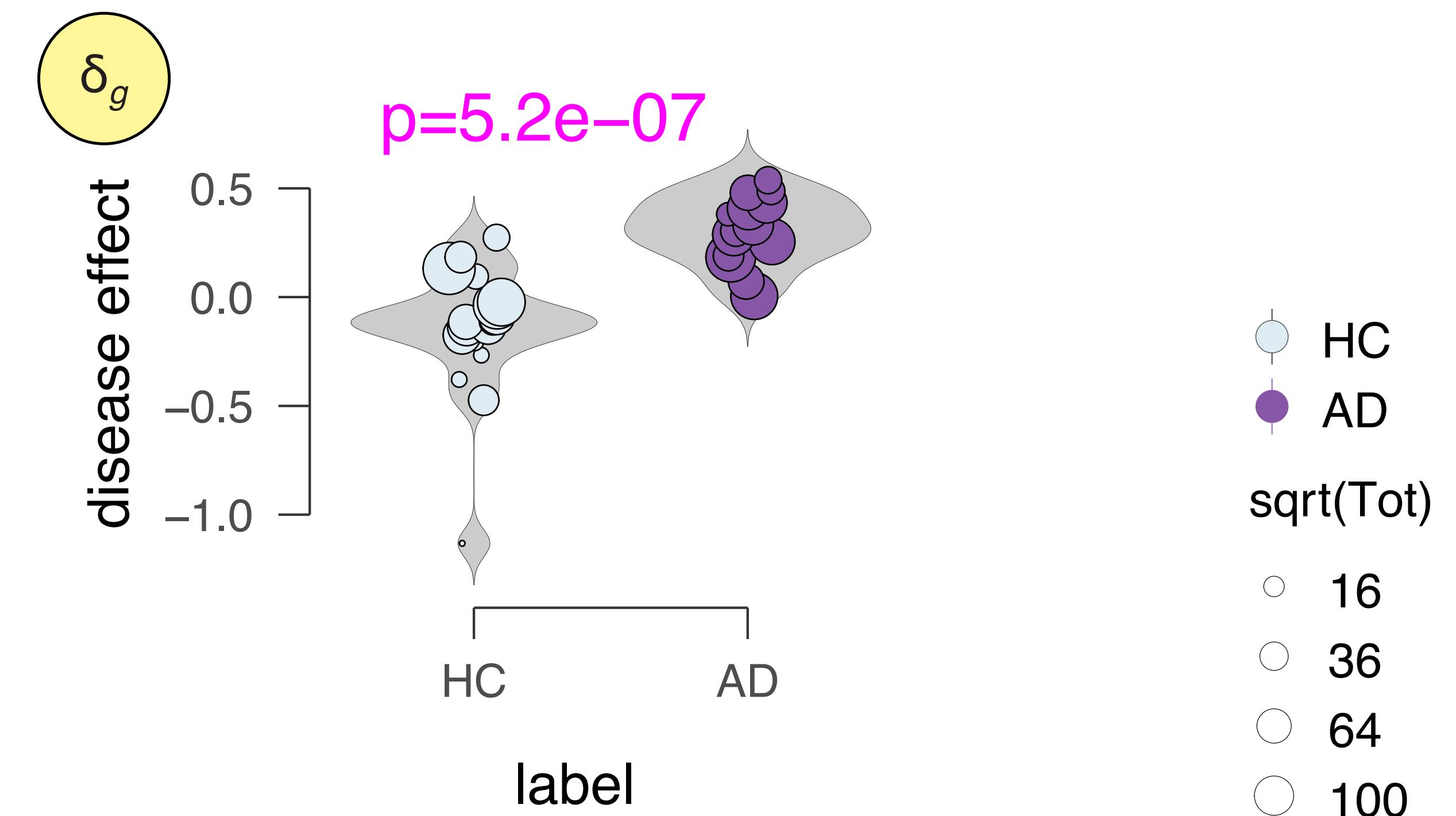
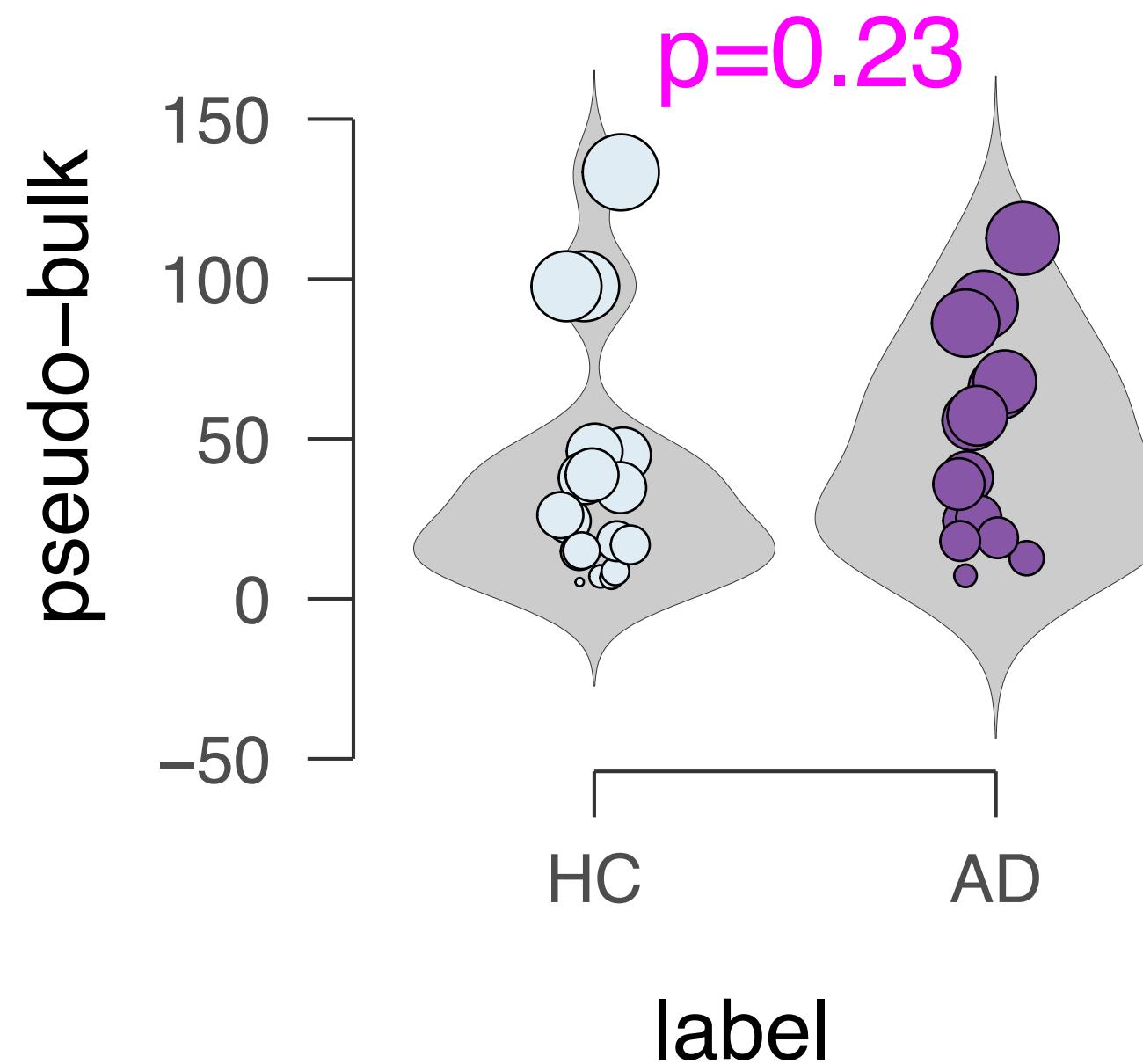


Differential expression analysis could be affected by confounding factors



Adjusting confounding factors will highlight true disease-specific effects in cell types

APOE in microglia



Unsupervised Learning in single-cell analysis

- **Review of PCA in single-cell data analysis**
 - What have we learned?
 - How can independent factors decompose variability?
- **Non-negative matrix factorization**
 - fastTopics & rliger for scalable inference
 - Post-hoc enrichment to relate factors to cell types
- **Can we identify cell type-specific disease genes?**
 - Pseudo-bulk-based DE approach

