Statistical Methods for single cell data analysis 2

Yongjin Park, UBC Path + Stat, BC Cancer

14

~~13~~ March 2024

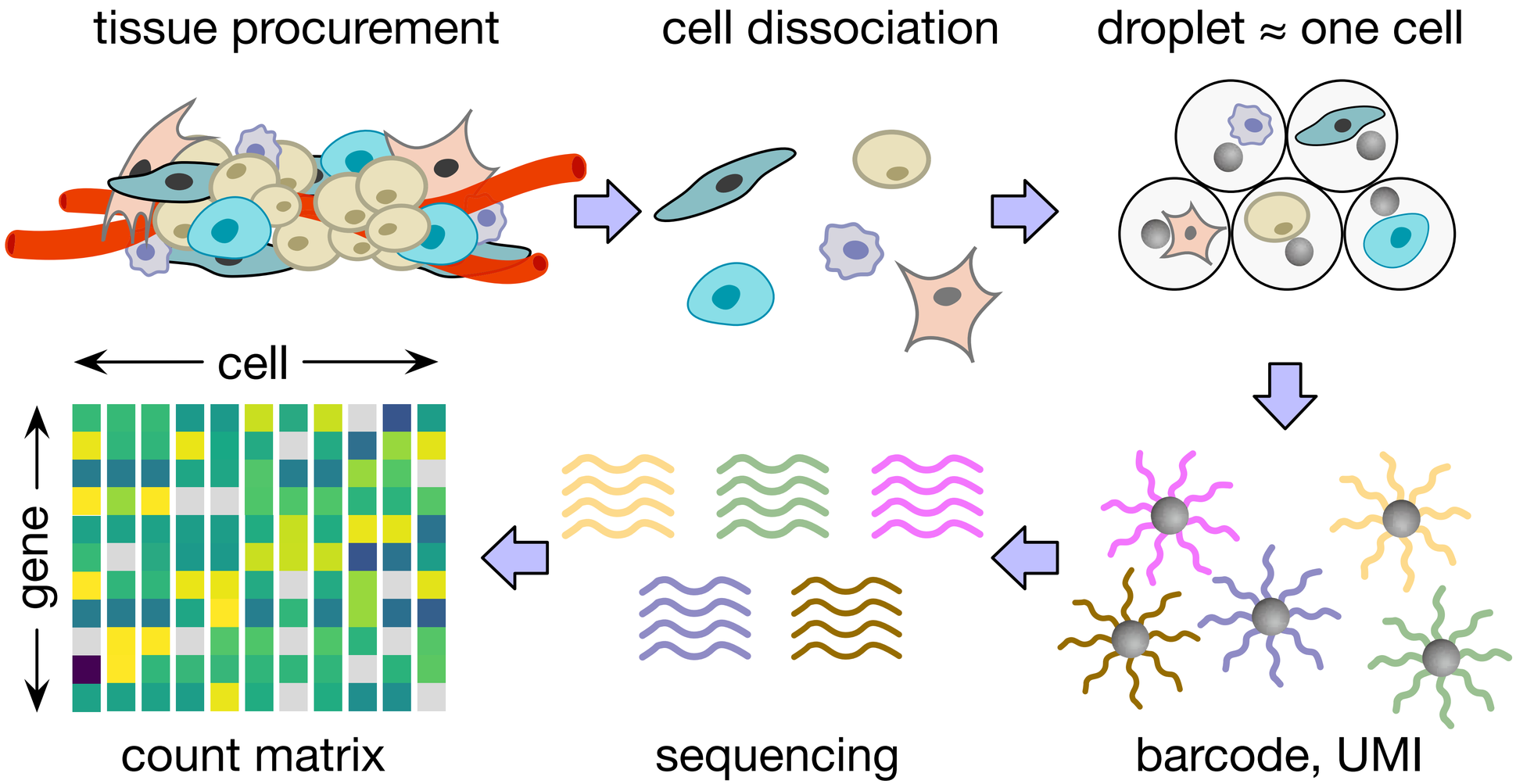
Unsupervised Learning in single-cell analysis

- **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`

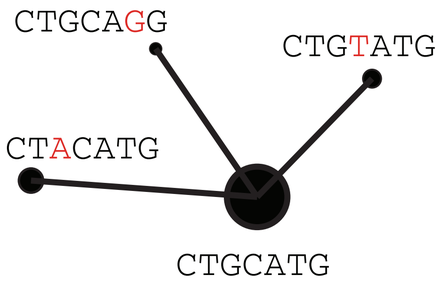
Droplet-based single-cell sequencing pipeline



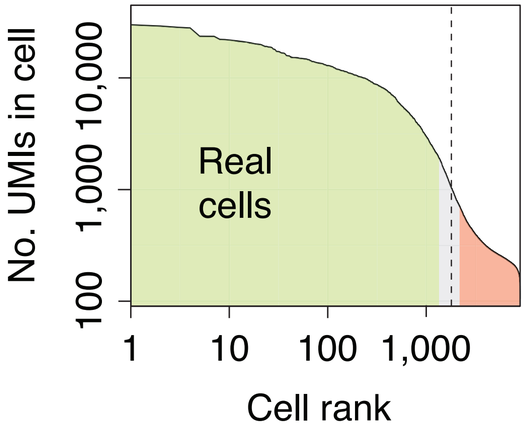
Macosko *et al.*, *Cell* (2015)

Overview of single-cell data analysis

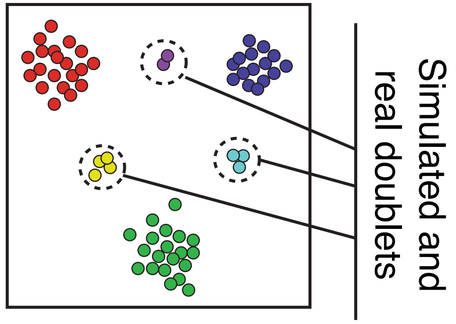
Alignment and molecular counting



Cell filtering and quality control



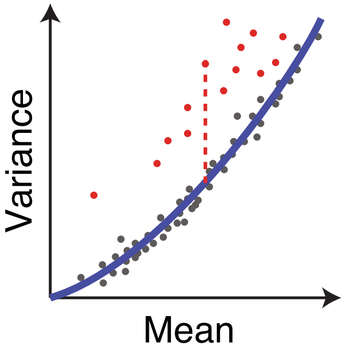
Doublet scoring



Cell size estimation

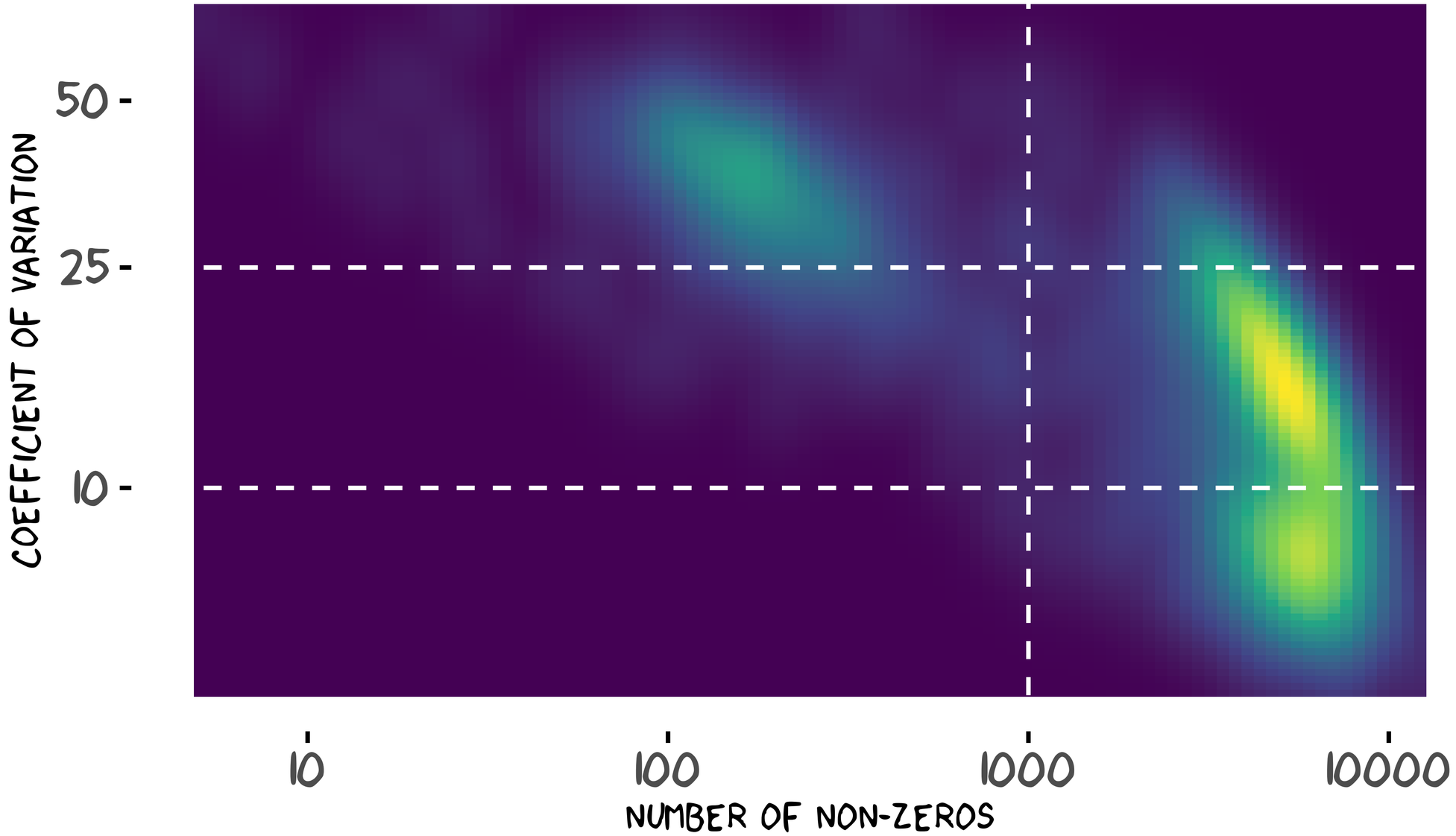
Cells	c_1	c_2	c_3
$Gene_1$	2	4	20
$Gene_2$	1	2	10
$Gene_3$	3	6	30
Cell depth:	6	12	60

Gene variance analysis



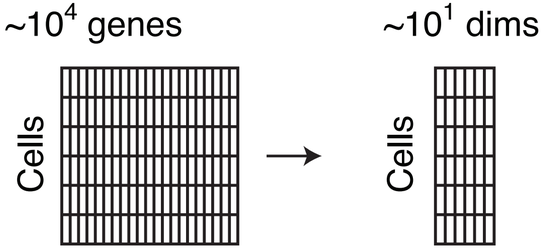
Karchenko, *Nature Methods* (2021)

Robust Q/C by putting multiple scores together



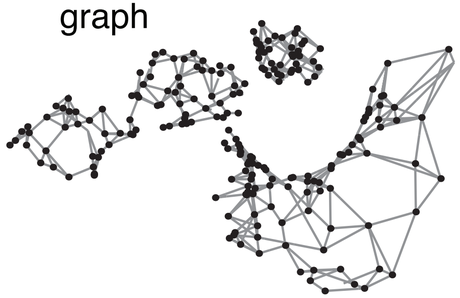
Overview of single-cell data analysis cont'd

Reduction to a medium-dimensional space

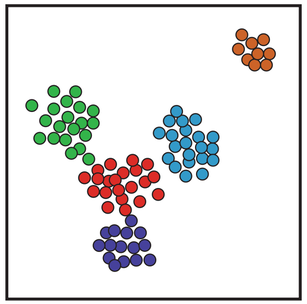


Manifold representation

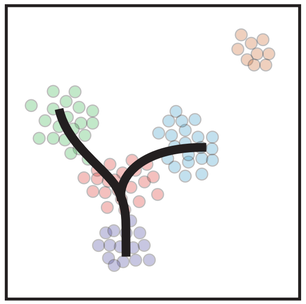
k-nearest-neighbors graph



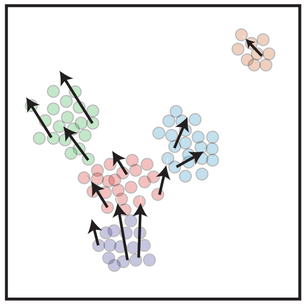
Clustering and differential expression



Trajectories



Velocity estimation



Karchenko, *Nature Methods* (2021)

Today’s lecture

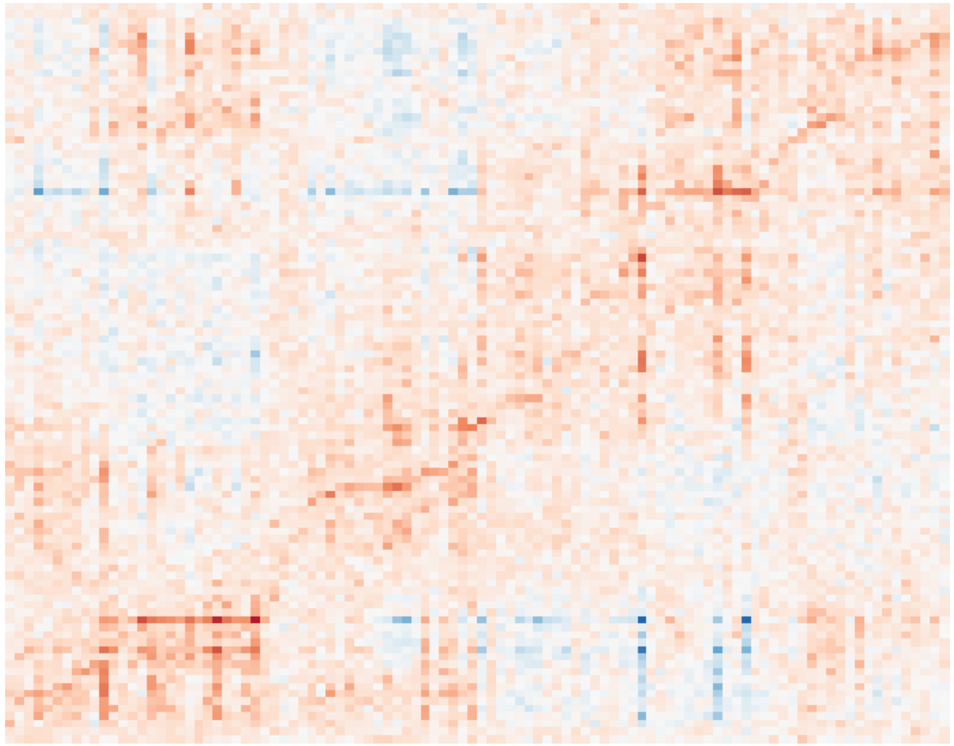
1. (review) Principal Component Analysis
2. Non-negative matrix factorization
3. Differential gene expression analysis

The goal is to find hidden patterns

A data matrix was generated by the following:

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

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?

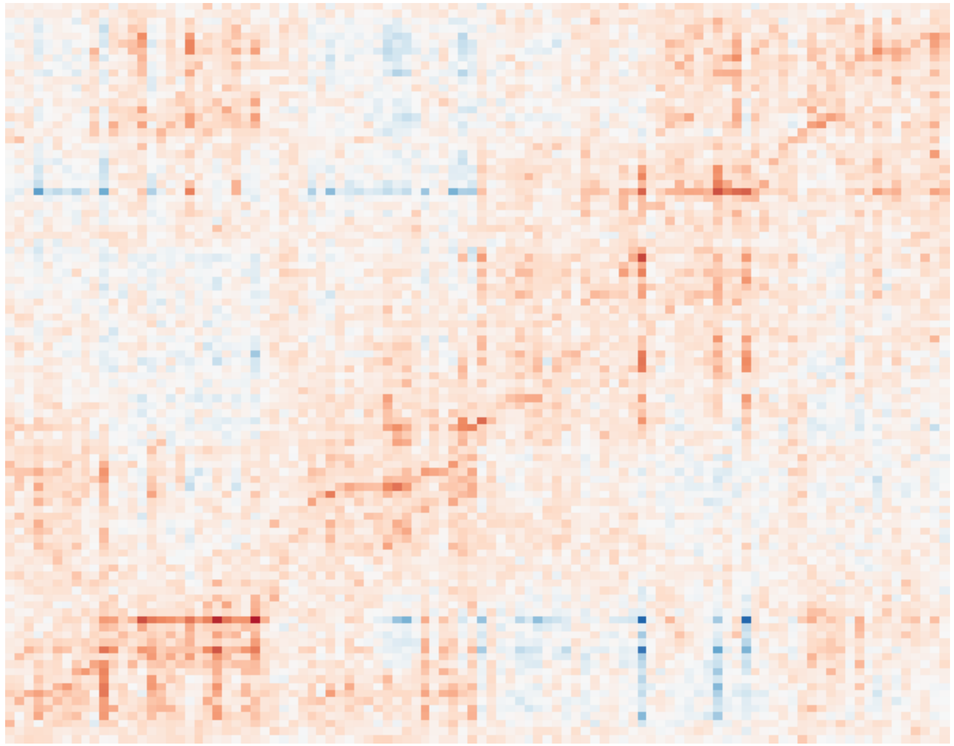
$$X \to UV^\top$$

The goal is to find hidden patterns

A data matrix was generated by the following:

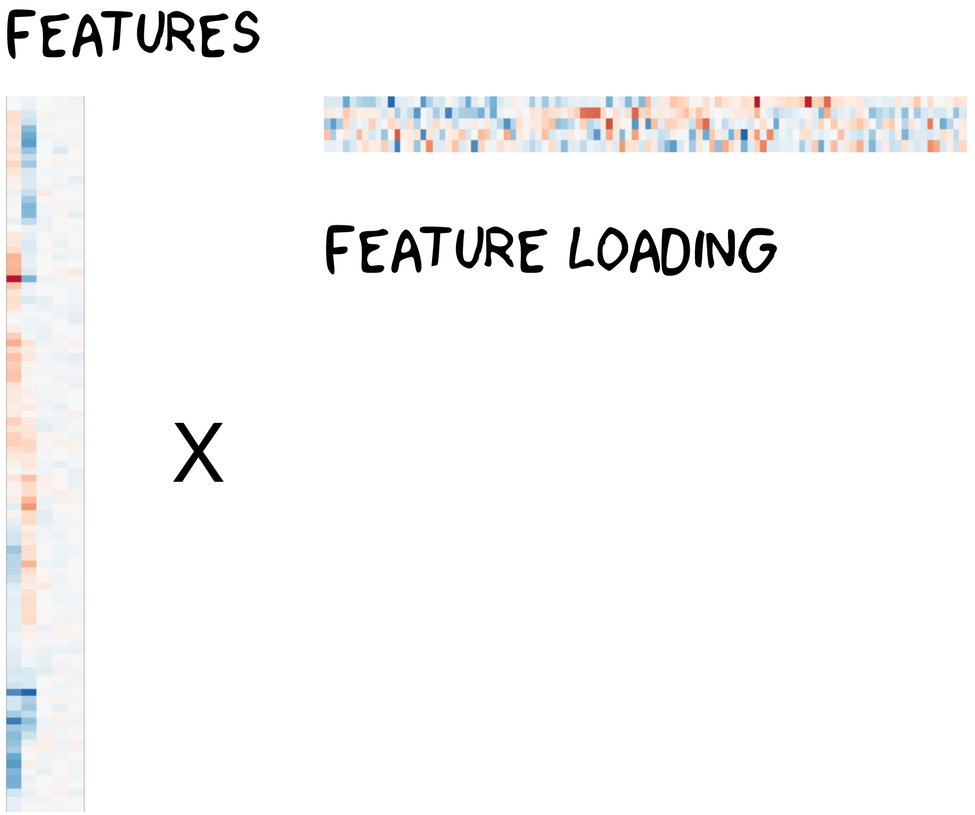
$$\mathbf{x}_i = \mathbf{u}_i V^\top + \epsilon_i$$

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

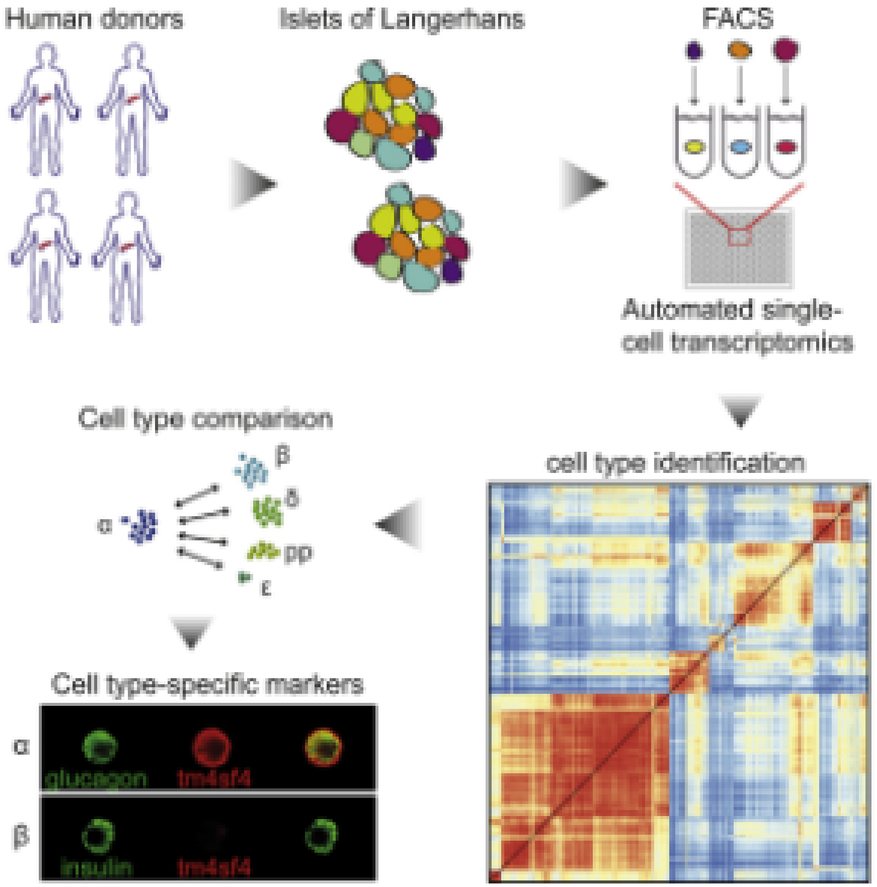


Can we reverse engineer the U and V matrices from the data X?

$$X \to UV^\top$$



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



It's a sparse matrix...

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

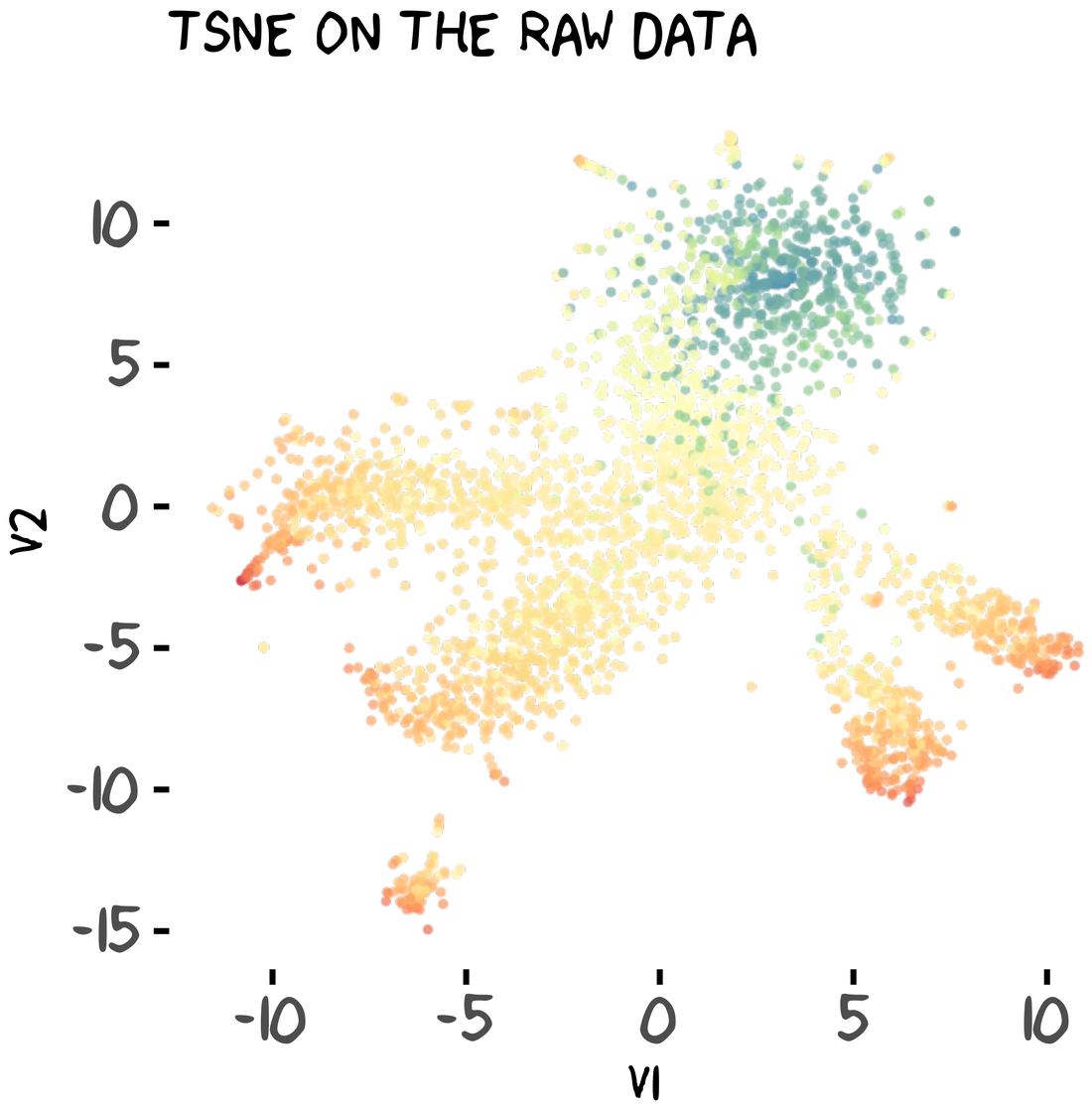
```
## 10 x 5 sparse Matrix of class "dgCMatrix"
##           D28-1_1 D28-1_2 D28-1_3 D28-1_4 D28-1_5
## A1BG-AS1        .       .       .       .       .
## A1BG            .       .       1       1       .
## A1CF            6       .       2       6       .
## A2M-AS1         .       .       .   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

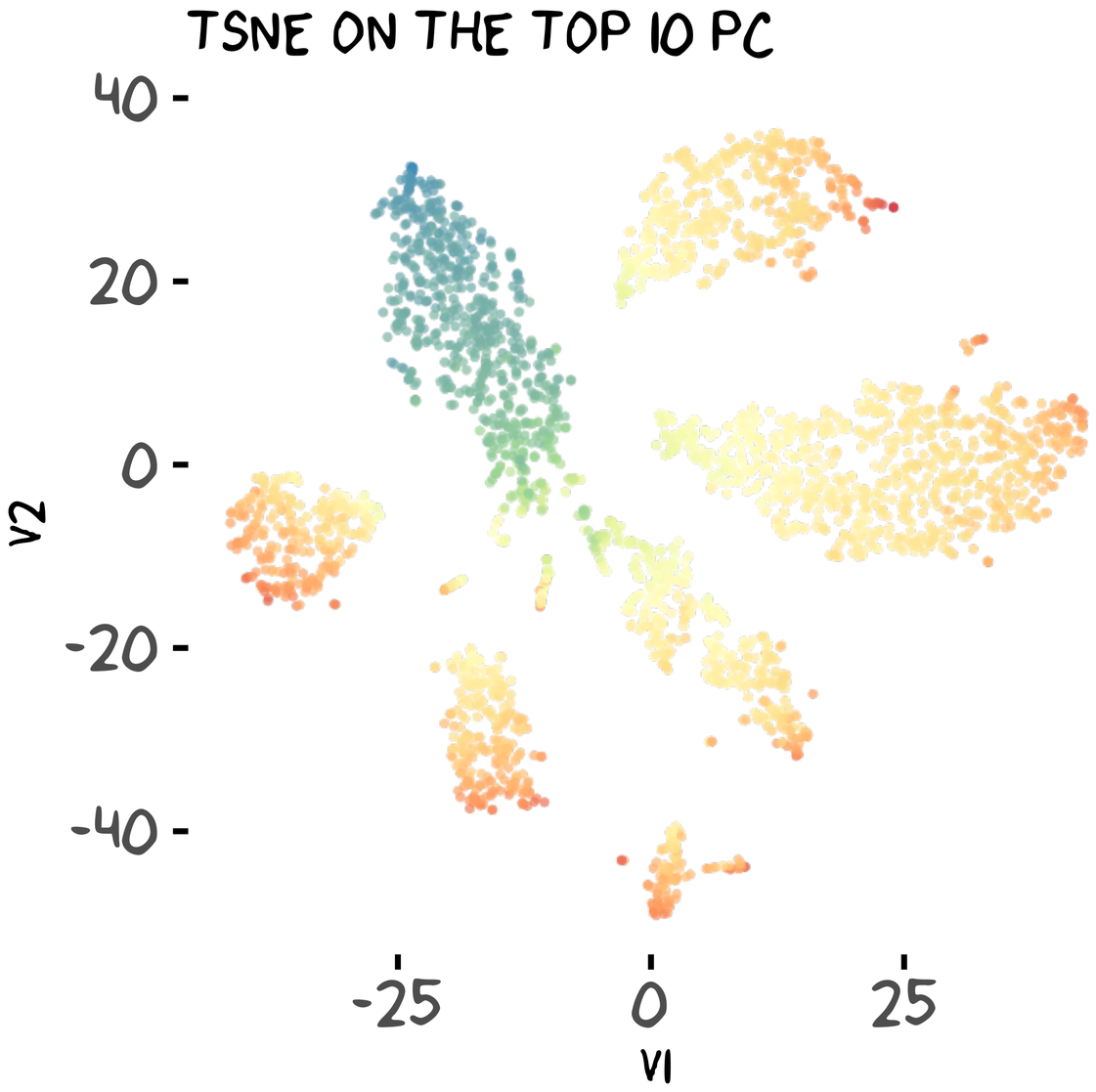
- rows: 19,049 genes
- columns: 3,072 cells

Why do we need to reduce dimensionality?

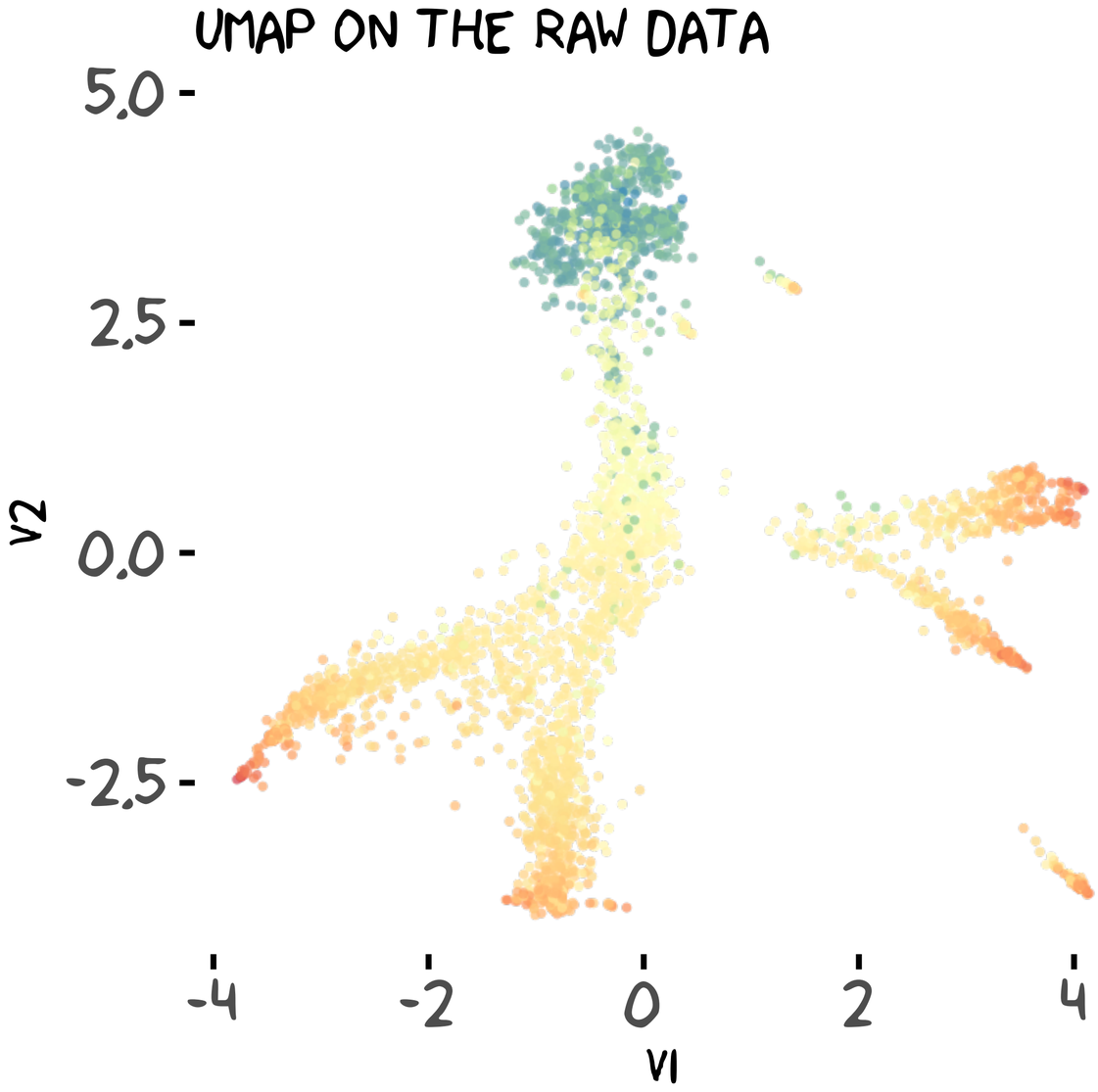
1. High dimensionality, robust downstream analysis
2. Pattern recognition: meta genes, meta cells, etc.
3. Reduce unwanted variation of outliers
4. Aggregate weak contributions of many features

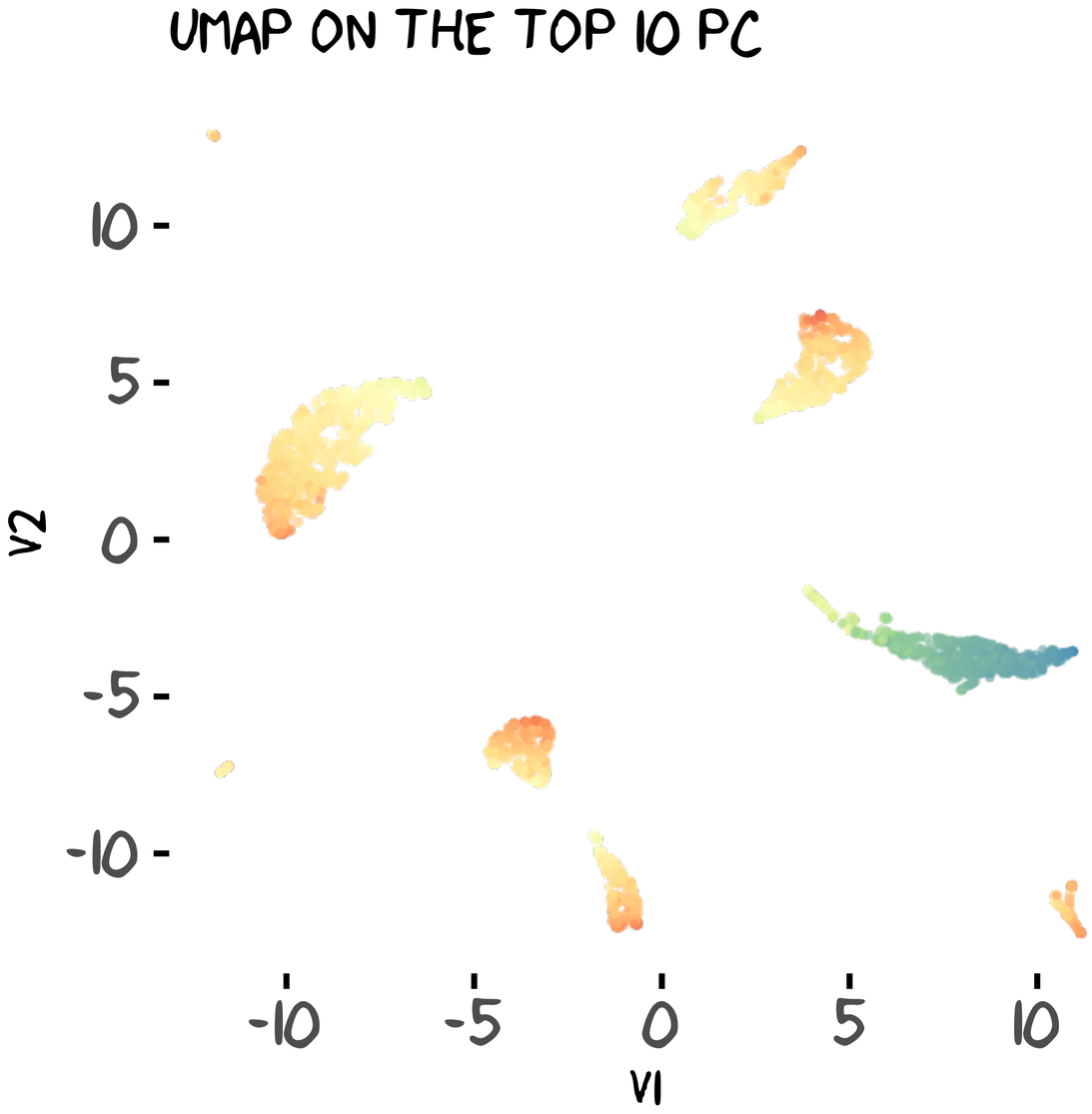
Visualization with or without dimensionality reduction





Visualization with or without dimensionality reduction





A key question: How do we learn U and V?

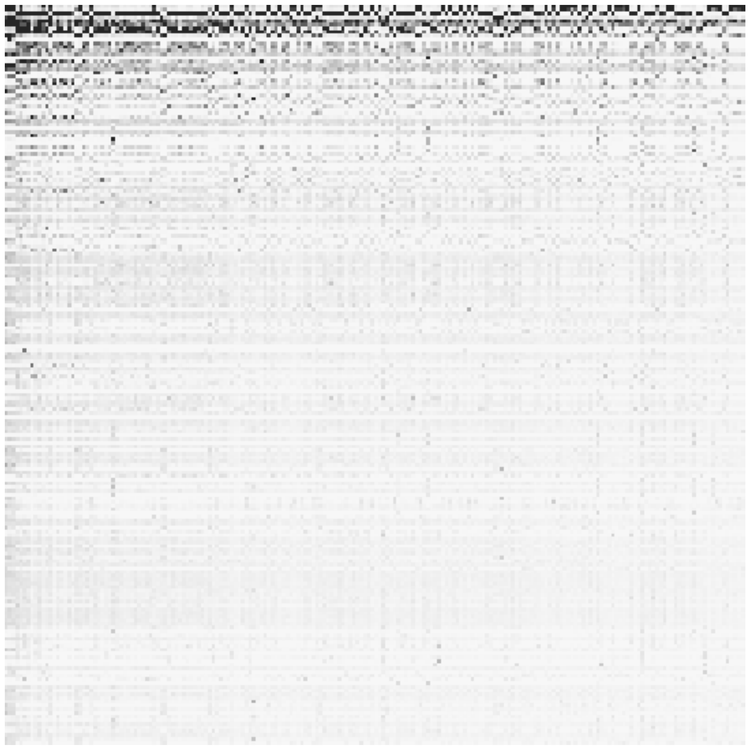
For many columns, $X = UV + E$, or

$$\begin{pmatrix} X_{11} & \cdots & X_{1n} \\ X_{21} & \cdots & X_{2n} \\ \vdots & & \vdots \\ X_{p1} & \cdots & X_{pn} \end{pmatrix} = \begin{pmatrix} U_{11} & \cdots & U_{1k} \\ U_{21} & \cdots & U_{2k} \\ \vdots & \vdots & \vdots \\ U_{p1} & \cdots & U_{pk} \end{pmatrix} \begin{pmatrix} V_{11} & \cdots & V_{1n} \\ \vdots & & \vdots \\ V_{k1} & \cdots & V_{kn} \end{pmatrix} + \cdots$$

- If we **knew** U, we would be able to solve weights V.
- And vice versa…

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

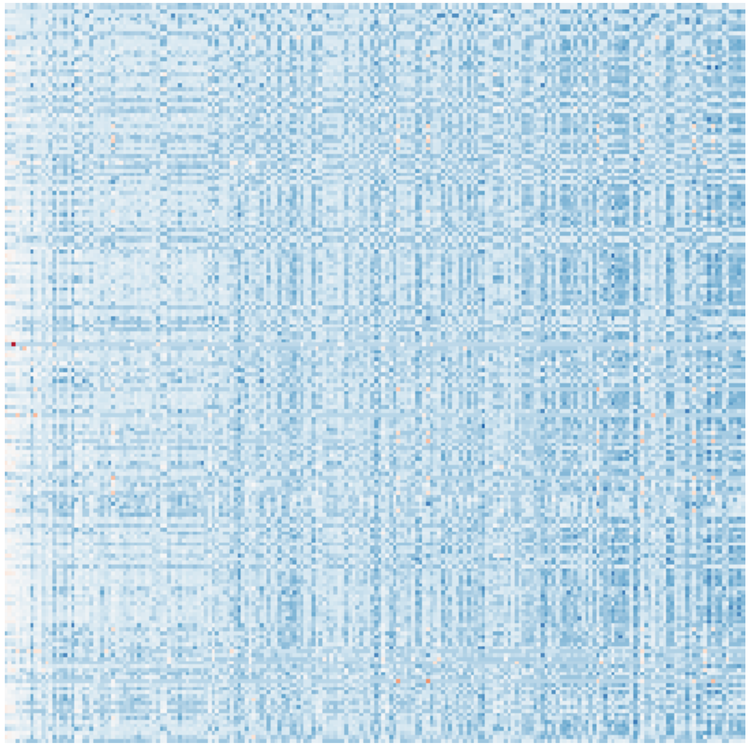
Top 200 genes and cells



- 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

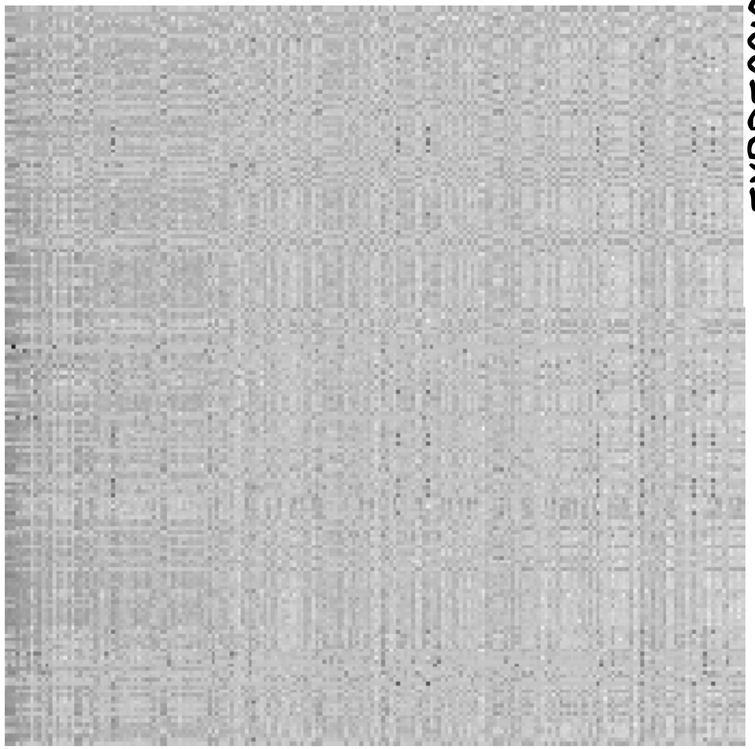
Top 200 genes and cells

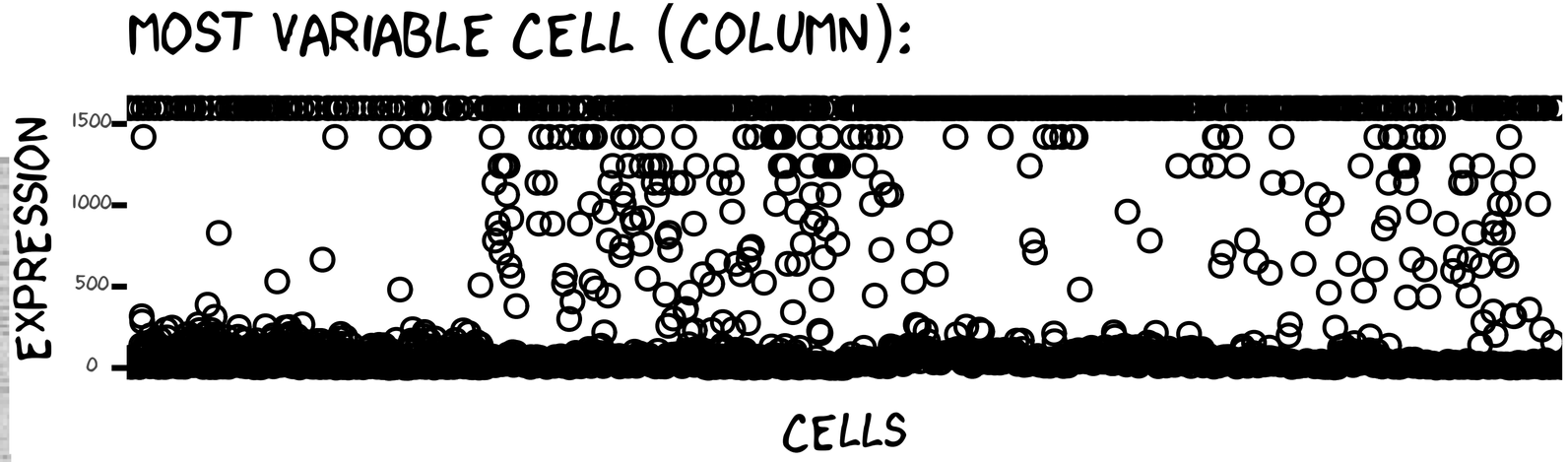


- We will call such a high-dimensional matrix X $(m \times n)$
- X has m=19,049 rows (transcripts/genes/features)
- X has n=3,072 columns (cells/#data points)
- The rows were log-transformed and scaled by `scale()` for visualization
- Each cell is a 19,049-dimensional vector!
- Each gene is a 3,072-dimensional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

1-dimensional representations/summary of data matrix

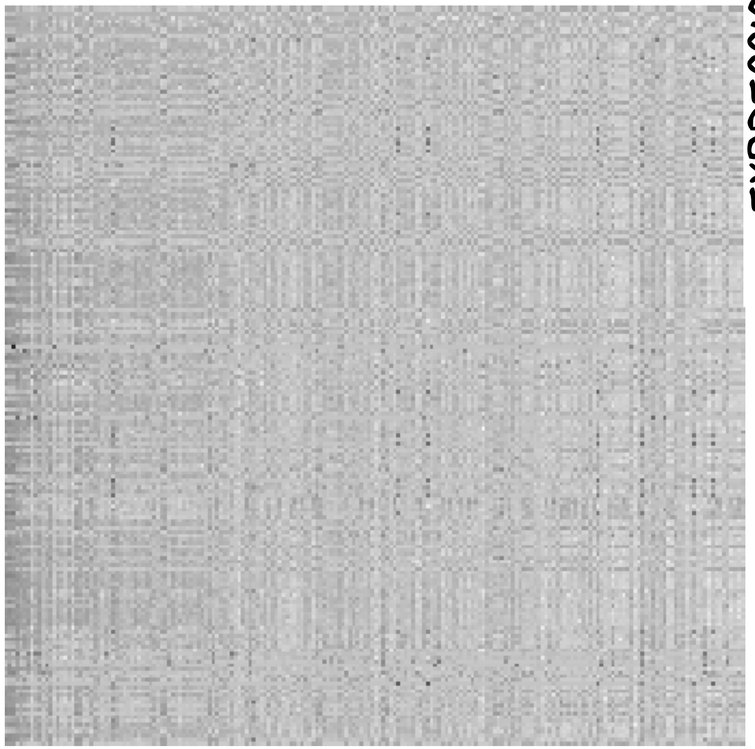
Top 200 genes and cells:

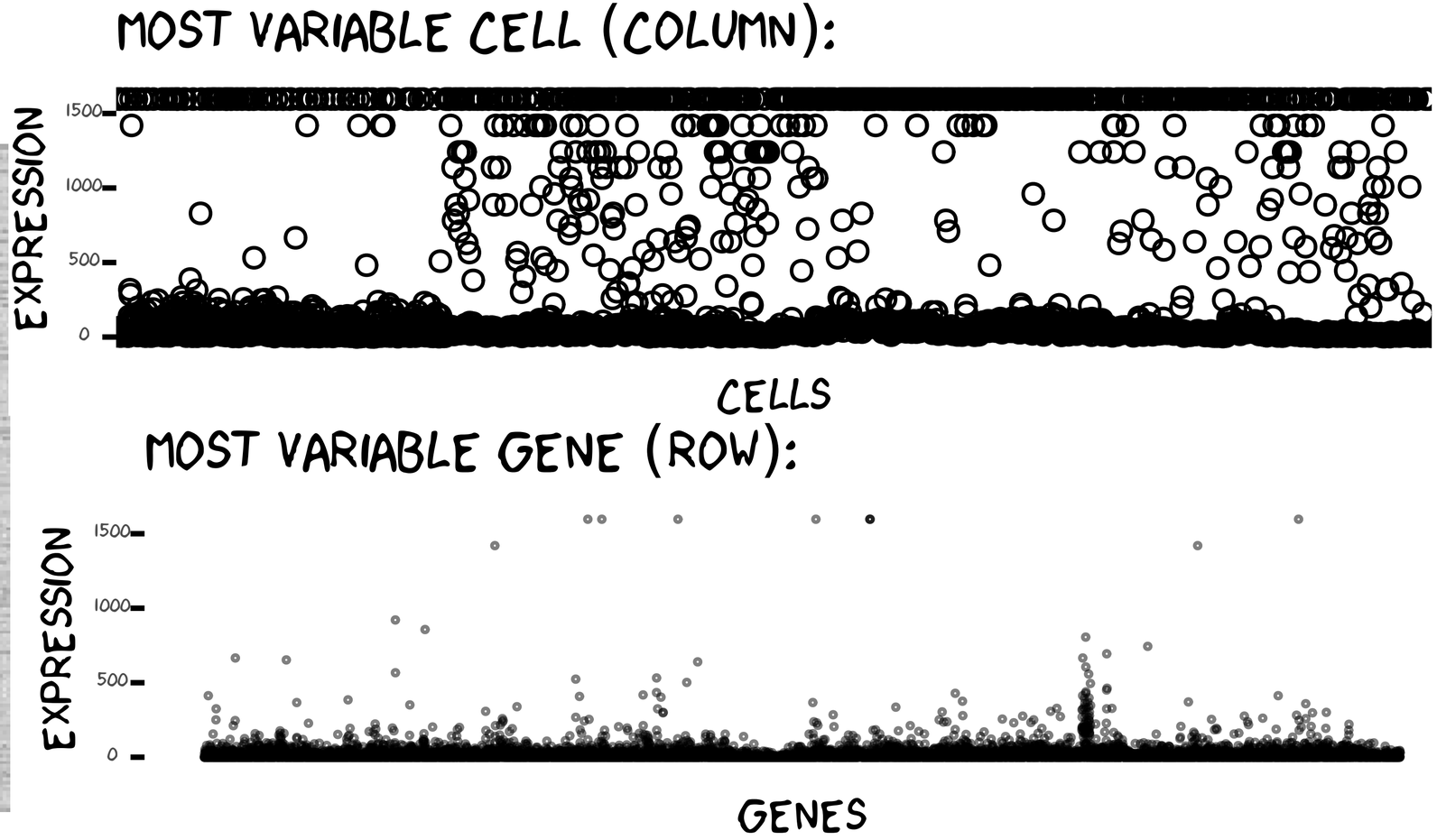




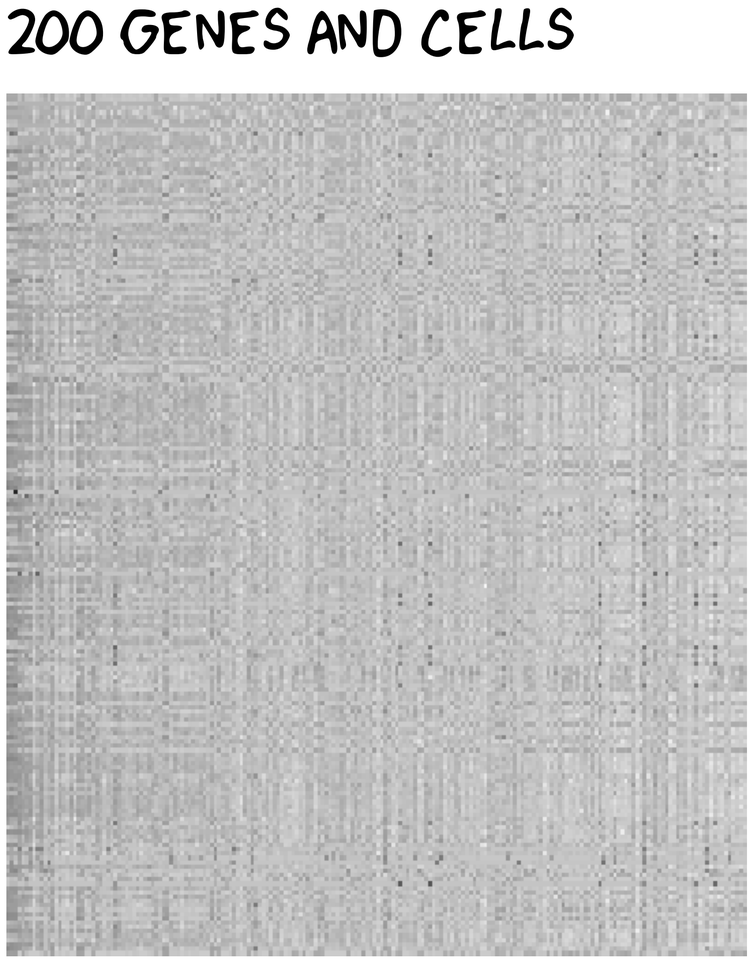
1-dimensional representations/summary of data matrix

Top 200 genes and cells:



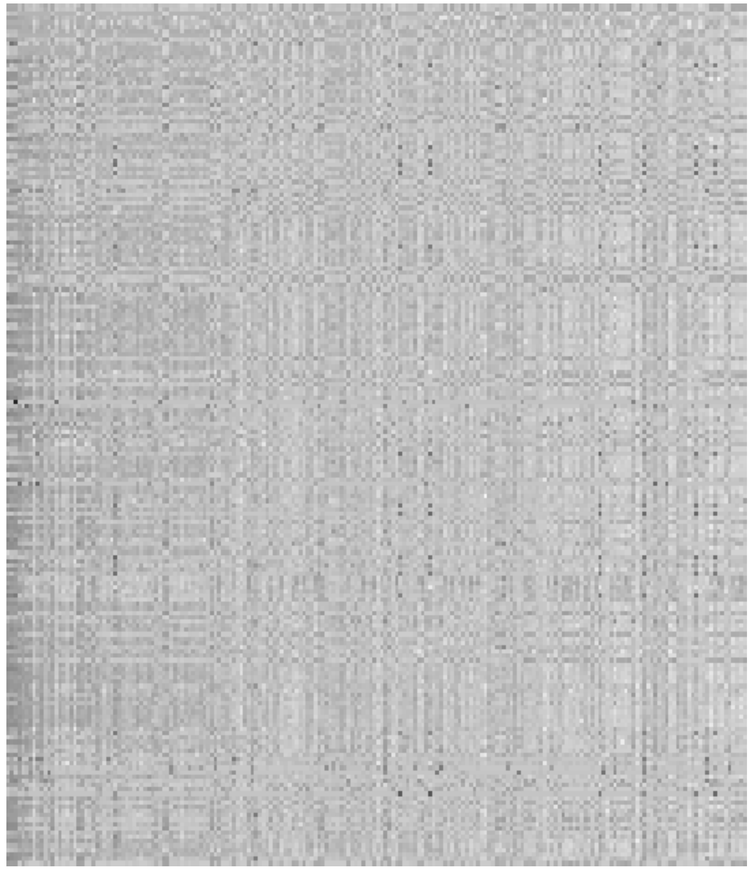


Can you see some common patterns?

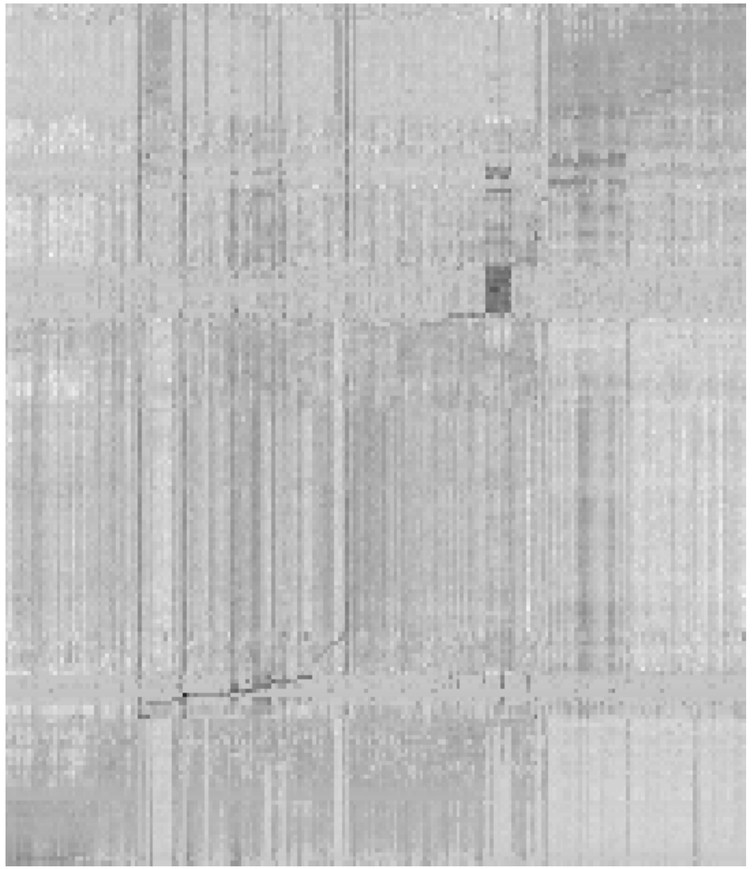


Can you see some common patterns?

200 GENES AND CELLS



REARRANGE THEM...



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

Principal Component Analysis

(Pearson 1901)

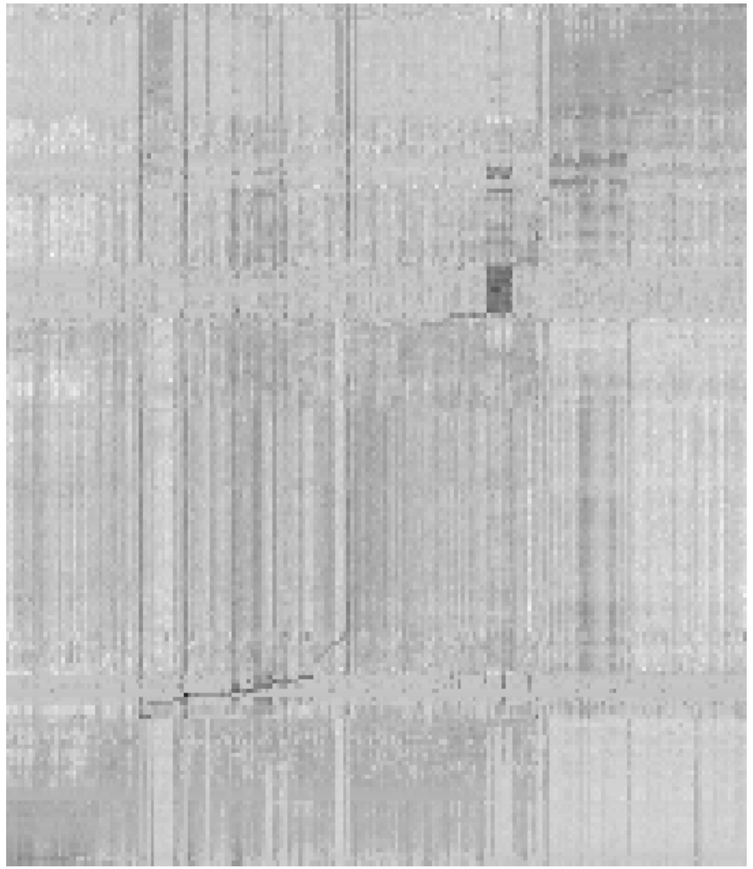
- Projection [of the original data] that minimizes the projection cost between the original and projected
- The cost = mean squared distance

(Hotelling 1933)

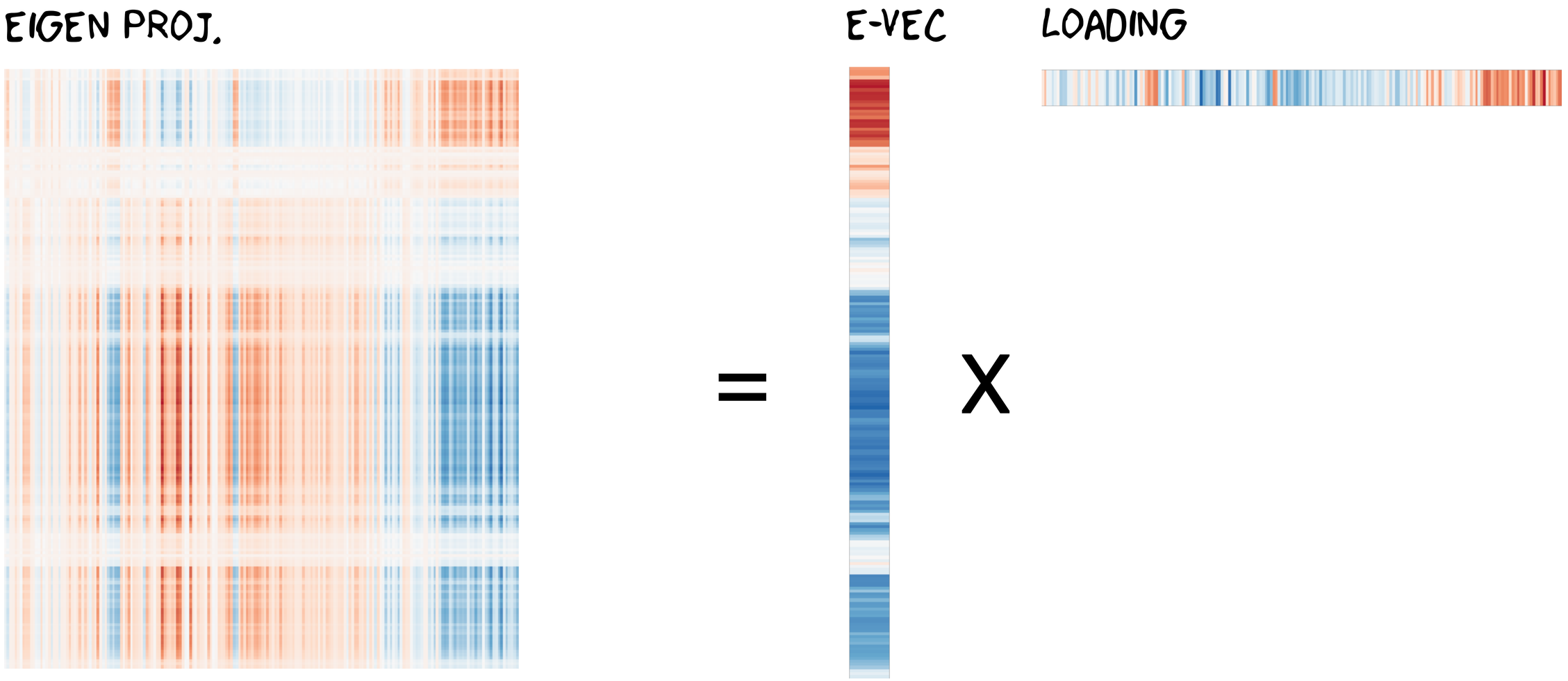
- Orthogonal projection of data into a lower-dimensional **[principal]** sub-space,
- such that the total **variation of the projected** is maximized

Let’s see how much the first eigenvector can explain

TOP 200 GENES AND CELLS

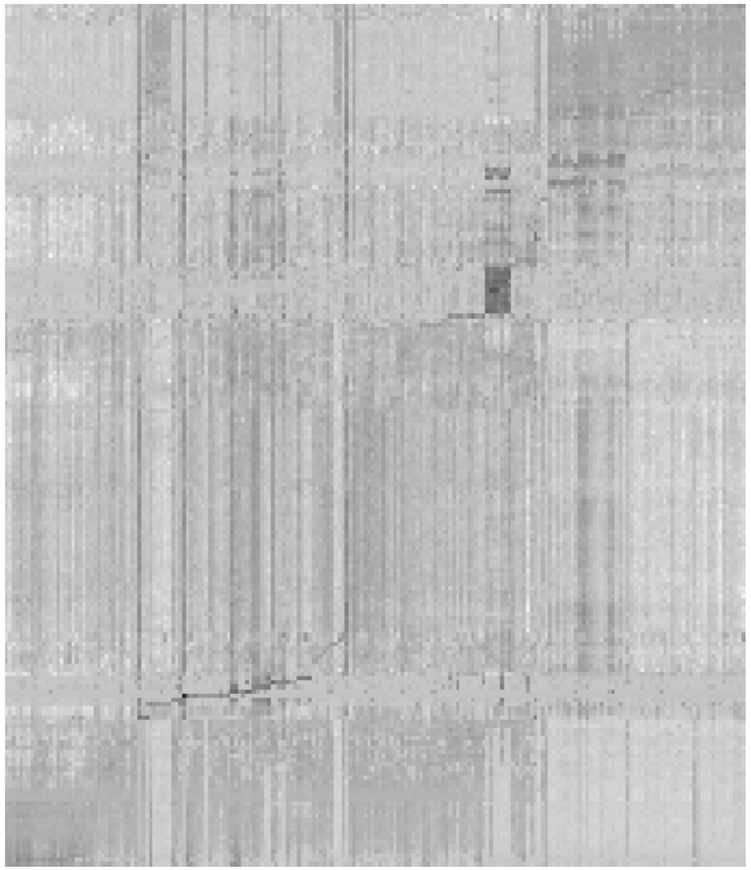


Let’s see how much the first eigenvector can explain

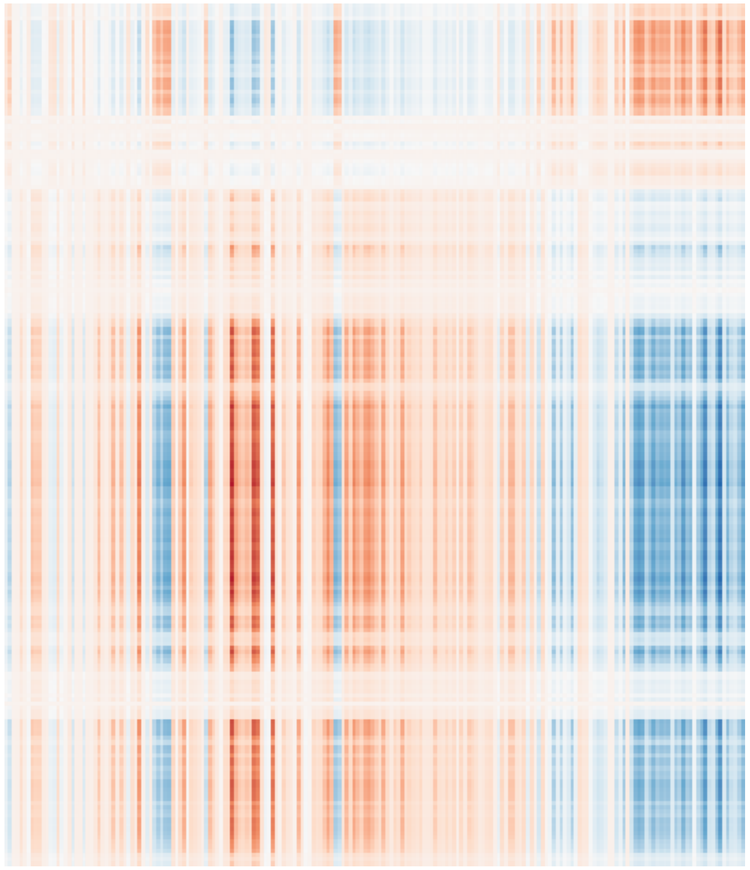


Let’s see how much the first eigenvector can explain

TOP 200 GENES AND 200 CELL

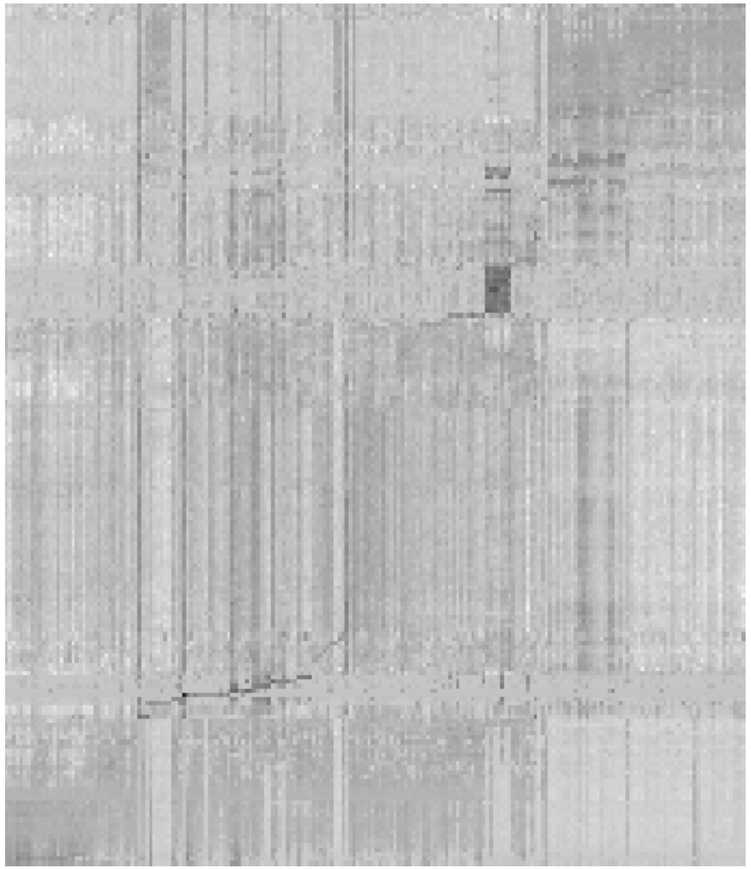


29%

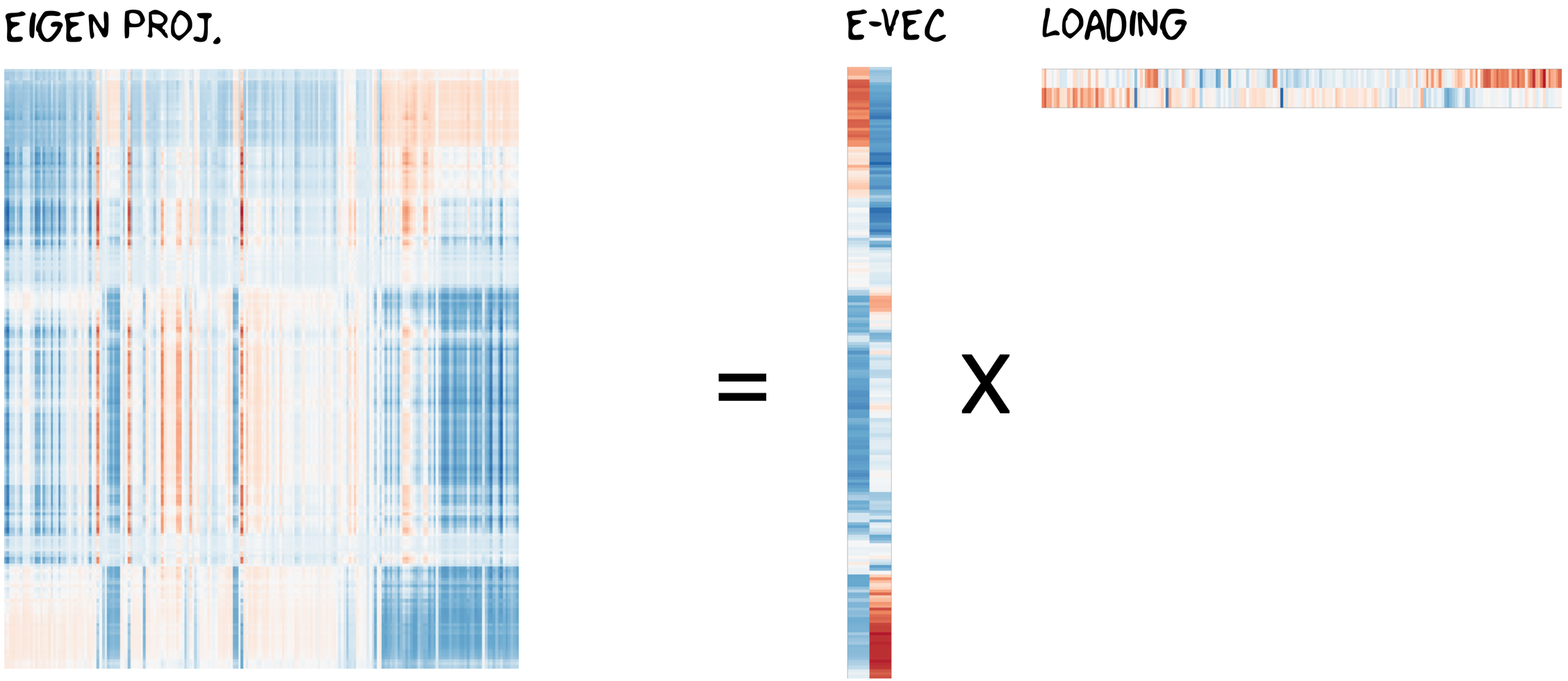


Top two eigen-vectors

TOP 200 GENES AND 200 CELL

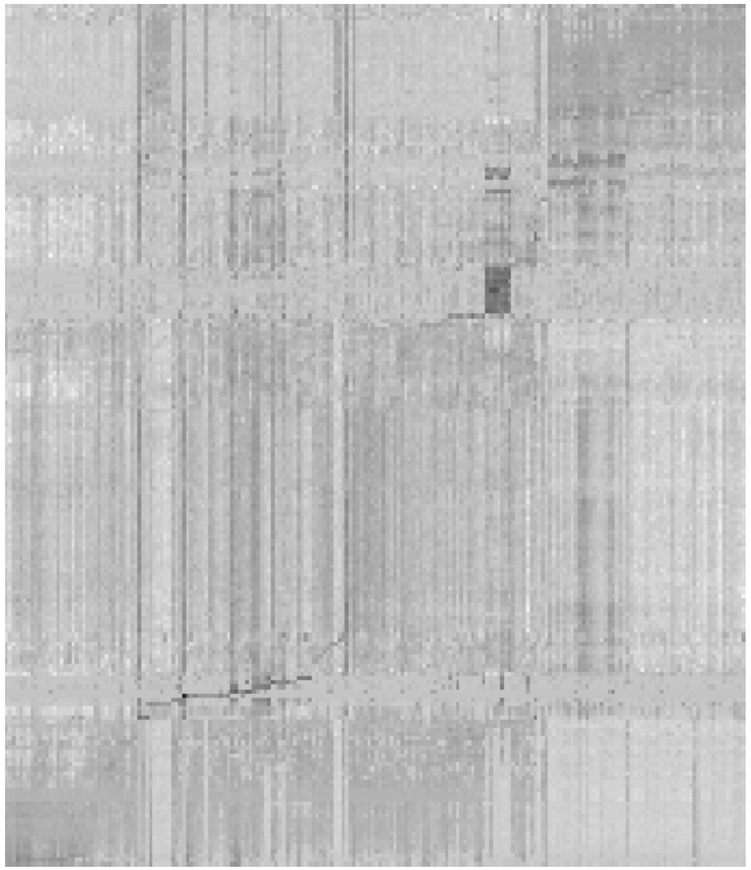


Top two eigen-vectors

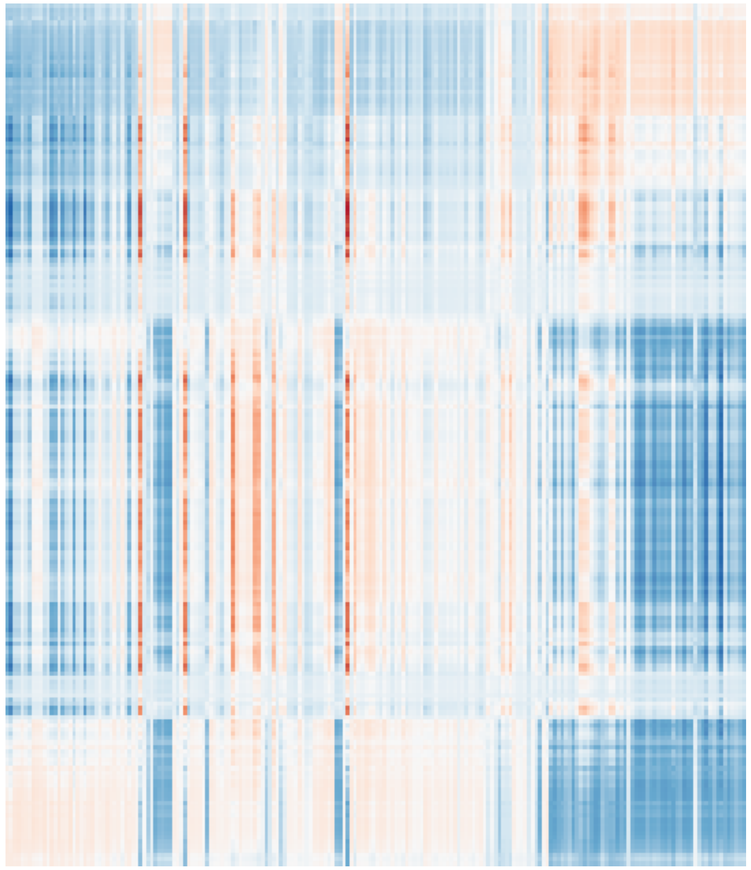


Top two eigen-vectors

TOP 200 GENES AND 200 CELL

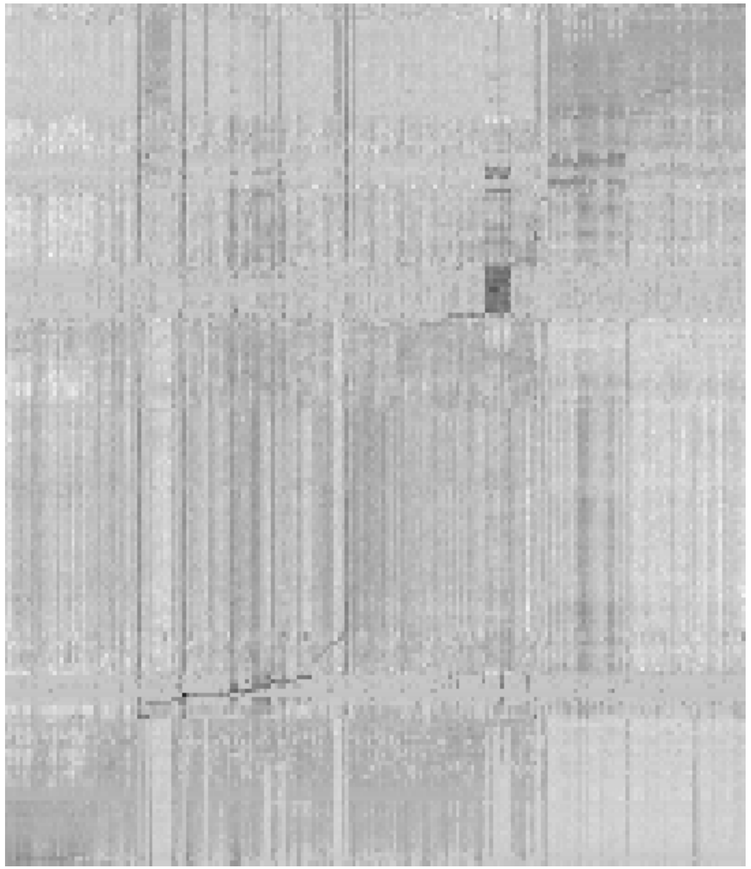


51%

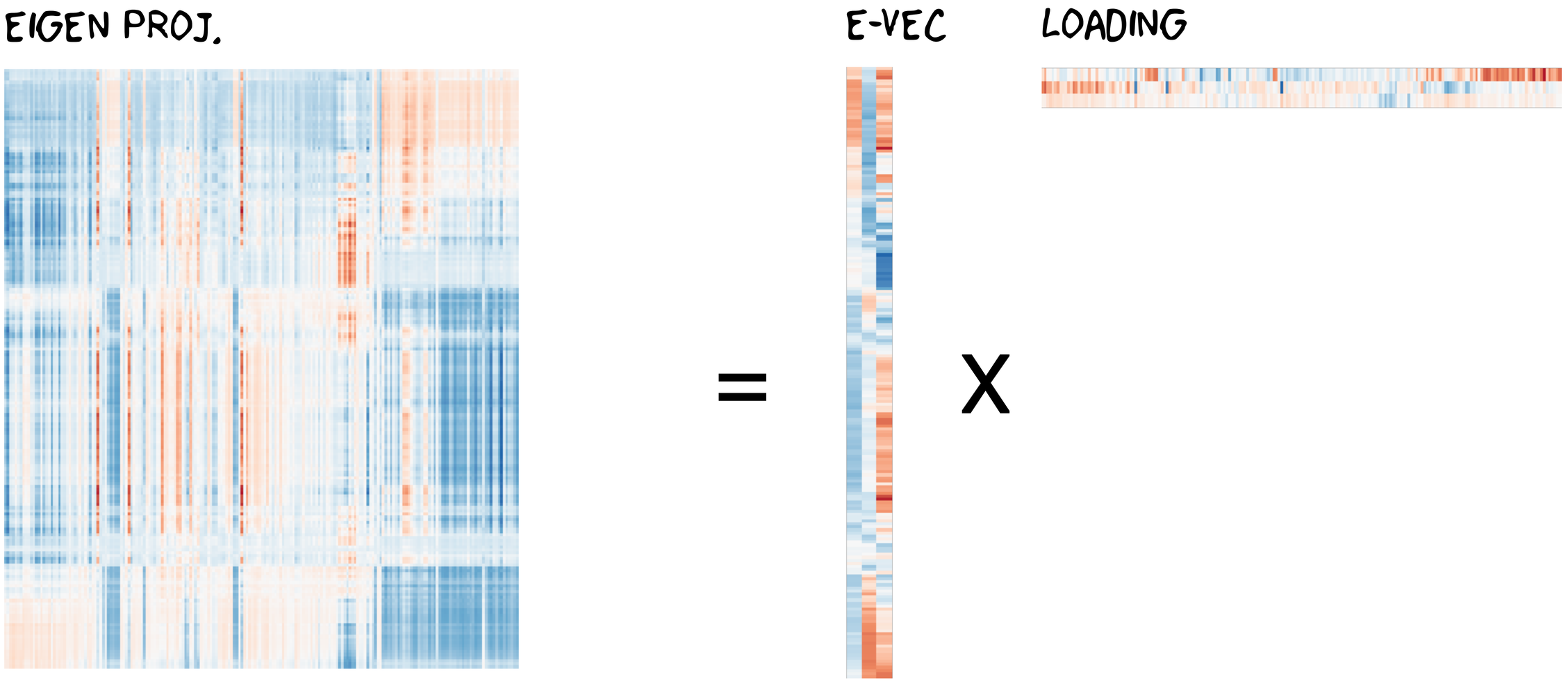


Top three eigen-vectors

TOP 200 GENES AND 200 CELL

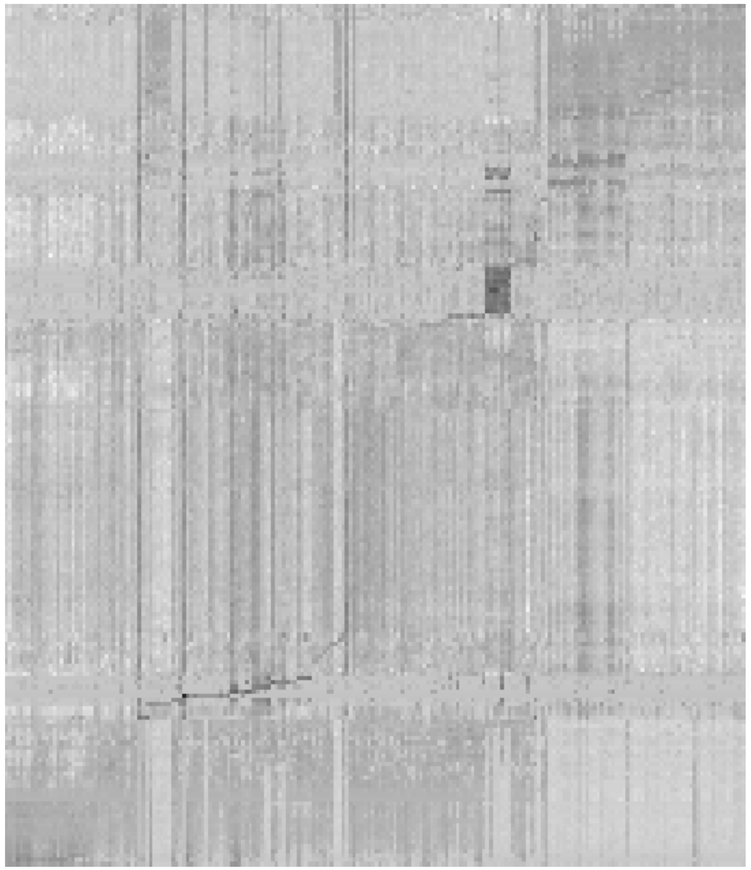


Top three eigen-vectors

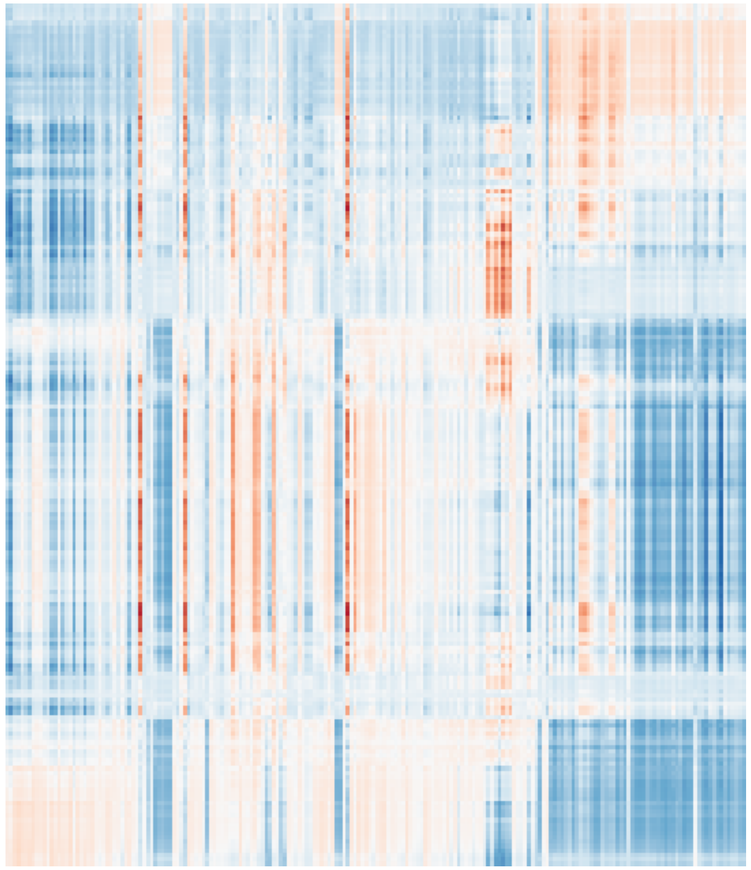


Top three eigen-vectors

TOP 200 GENES AND 200 CELL



58%



Singular Value Decomposition can find a PCA solution

Singular Value Decomposition

SVD identifies three matrices of X:

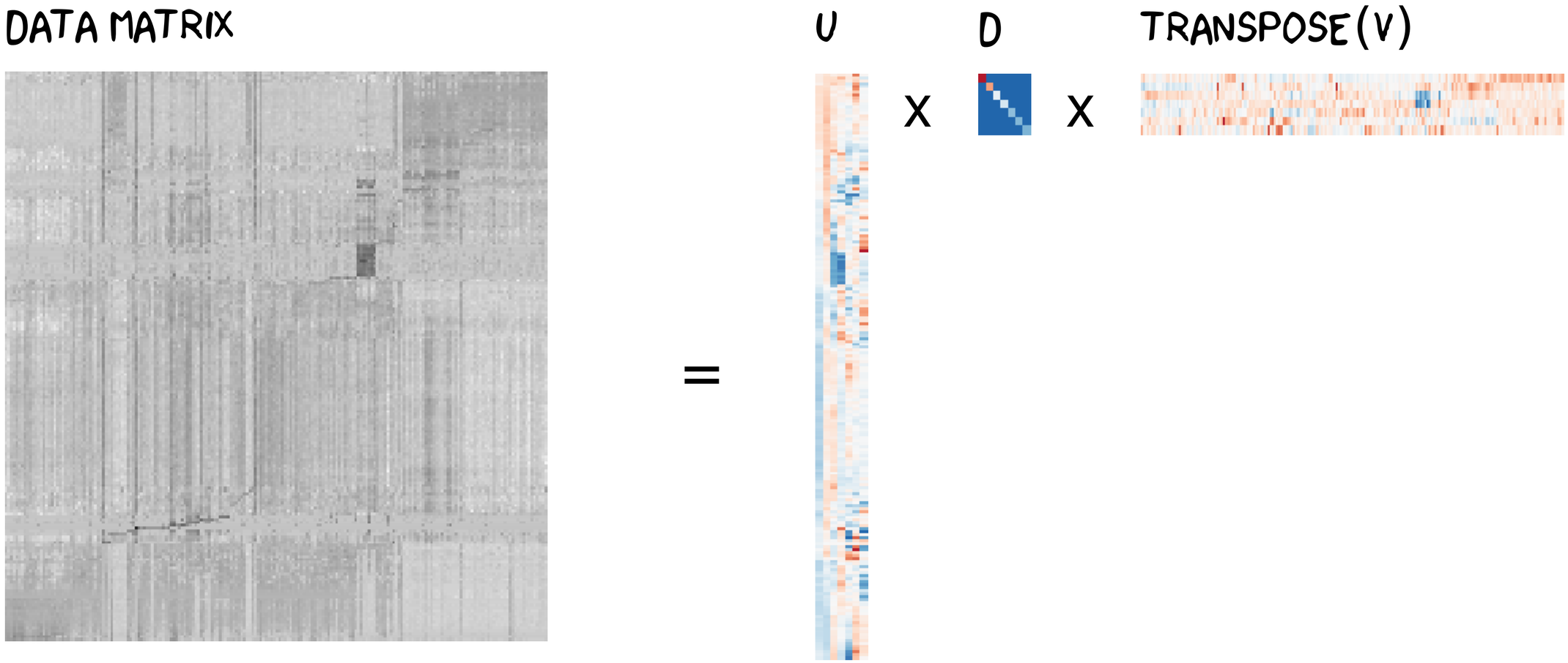
$$X = UDV^\top$$

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
```



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)

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

Covariance across the rows (genes)

$$XX^\top/n = UD^2U^\top/n$$

Remark: standardized matrix

Review: SVD finds eigenvectors of covariance

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

Review: SVD finds eigenvectors of covariance

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

$$= \frac{1}{m}(\mathbf{v}_1, \mathbf{v}_2, \dots, \mathbf{v}_k)\begin{pmatrix} D_1^2 & 0 & \dots & \dots \\ 0 & D_2^2 & 0 & \dots \\ 0 & \dots & \ddots & 0 \\ 0 & \dots & 0 & D_k^2 \end{pmatrix}\begin{pmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix}$$

Review: SVD finds eigenvectors of covariance

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

$$= \frac{1}{m}(\mathbf{v}_1, \mathbf{v}_2, \dots, \mathbf{v}_k)\begin{pmatrix} D_1^2 & 0 & \dots & \dots \\ 0 & D_2^2 & 0 & \dots \\ 0 & \dots & \ddots & 0 \\ 0 & \dots & 0 & D_k^2 \end{pmatrix}\begin{pmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix}$$

$$= \frac{1}{m}(\mathbf{v}_1, \mathbf{v}_2, \dots, \mathbf{v}_k)\begin{pmatrix} D_1^2 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

Review: SVD finds eigenvectors of covariance

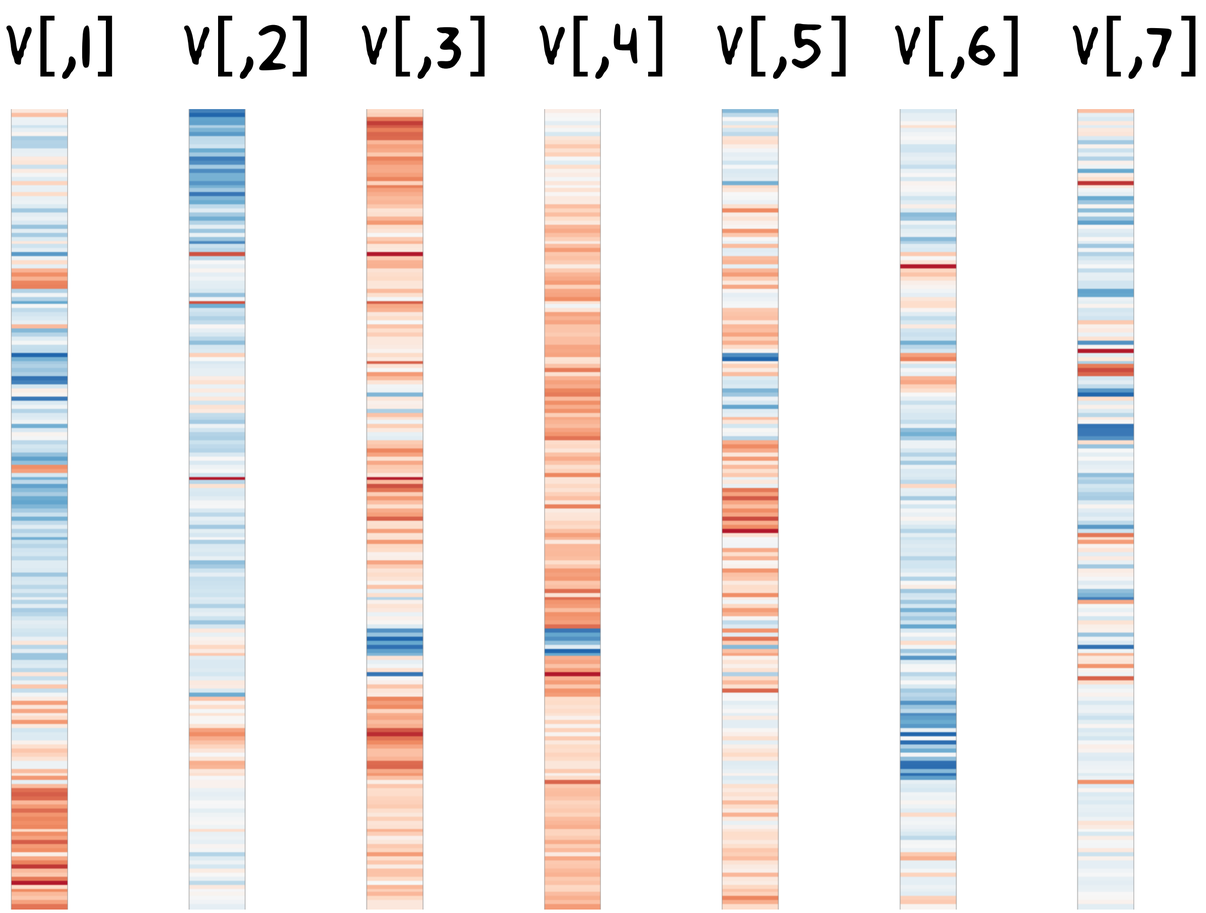
$$\underbrace{\left(\frac{1}{m}X^\top X\right)}_{\text{sample covariance}}\mathbf{v}_1 = \frac{1}{m}(\mathbf{v}_1, \mathbf{v}_2, \dots, \mathbf{v}_k)\begin{pmatrix} D_1^2 & 0 & \dots & \dots \\ 0 & D_2^2 & 0 & \dots \\ 0 & \dots & \ddots & 0 \\ 0 & \dots & 0 & D_k^2 \end{pmatrix}\begin{pmatrix} \mathbf{v}_1^\top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

$$= \frac{1}{m}(\mathbf{v}_1, \mathbf{v}_2, \dots, \mathbf{v}_k)\begin{pmatrix} D_1^2 & 0 & \dots & \dots \\ 0 & D_2^2 & 0 & \dots \\ 0 & \dots & \ddots & 0 \\ 0 & \dots & 0 & D_k^2 \end{pmatrix}\begin{pmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix}$$

$$\hat{\Sigma}\mathbf{v}_1 = \frac{1}{m}(\mathbf{v}_1, \mathbf{v}_2, \dots, \mathbf{v}_k)\begin{pmatrix} D_1^2 \\ 0 \\ 0 \\ 0 \end{pmatrix} = \underbrace{\frac{D_1^2}{m}}_{\text{eigenvalue}}\underbrace{\mathbf{v}_1}_{\text{eigenvector}}$$

Eigen values λ's quantify the proportion of variance

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



V is orthonormal (orthogonal and normalized):

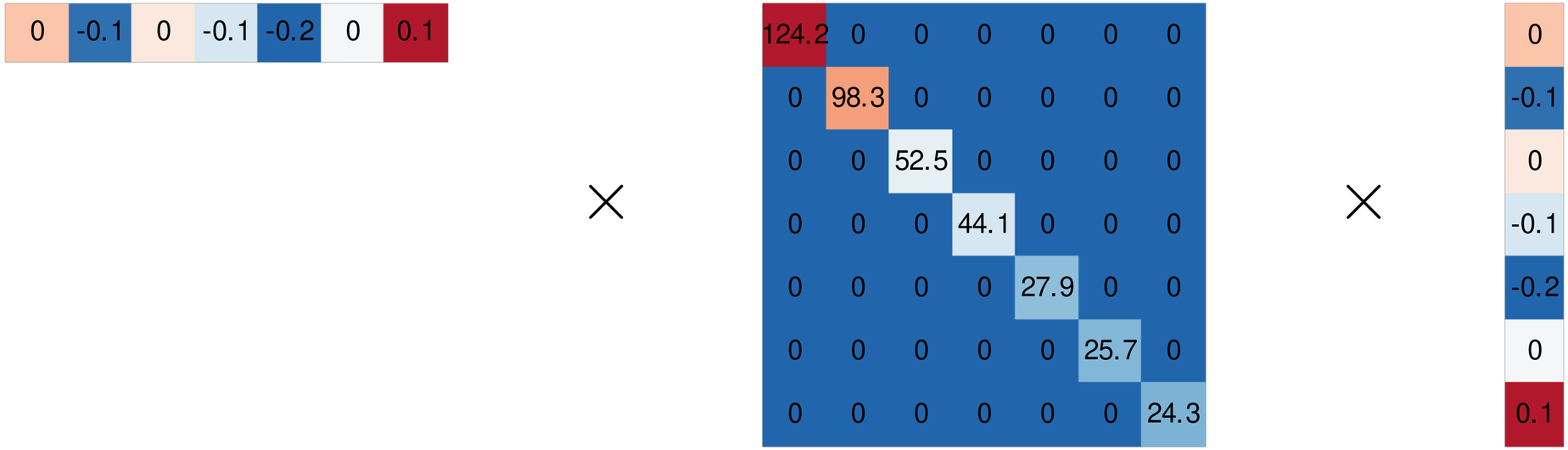
TRANSPOSE(V) * V

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

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



Eigen values λ's quantify the proportion of variance

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

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

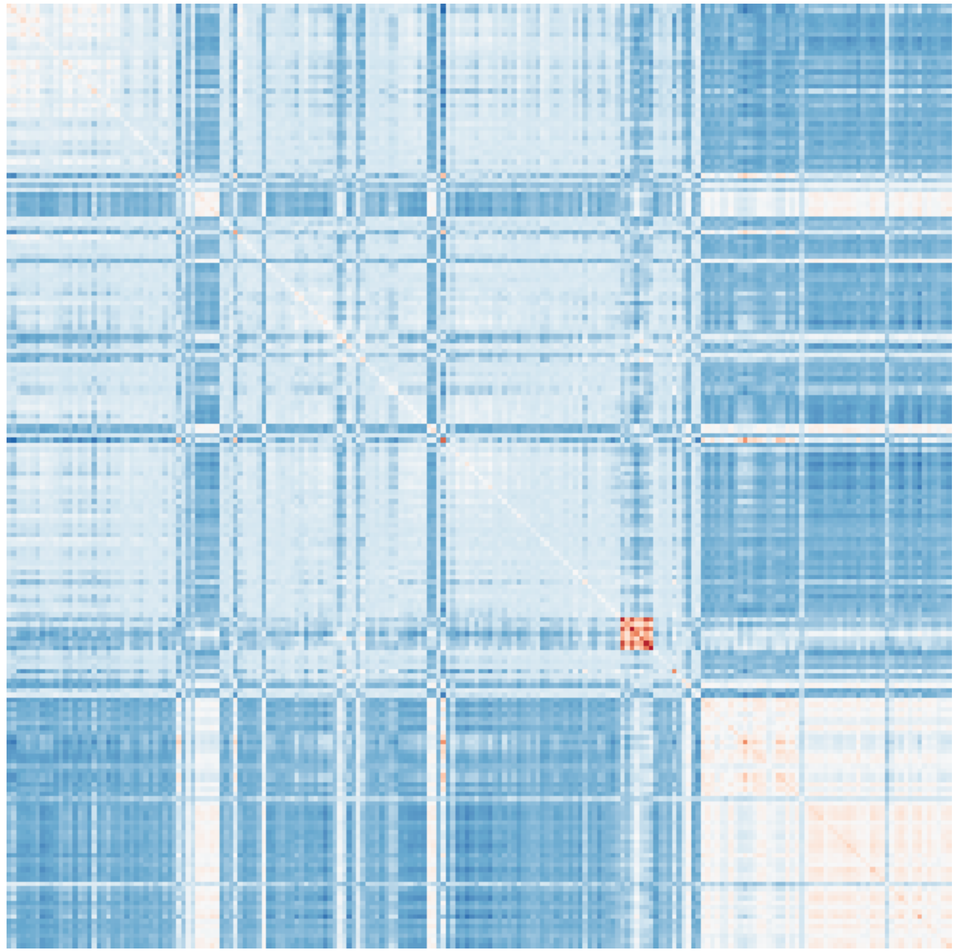
Therefore,

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

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

Sample covariance matrix

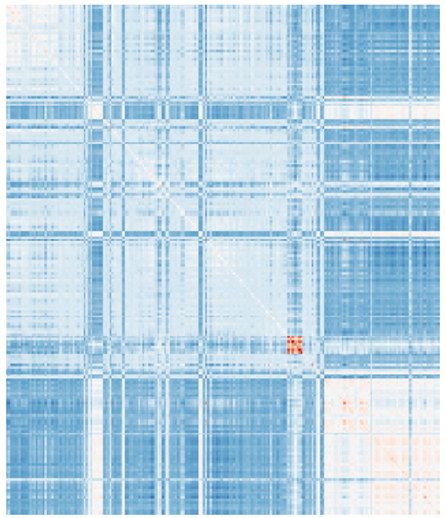
$$\hat{\Sigma} = X^\top X / m$$



PCs (SVs) decompose the covariance matrix

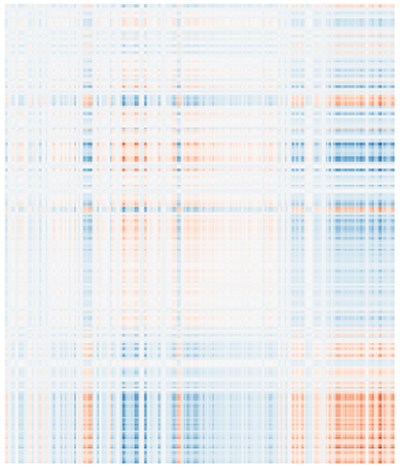
$$\hat{\Sigma} = X^\top X / m$$

N X N



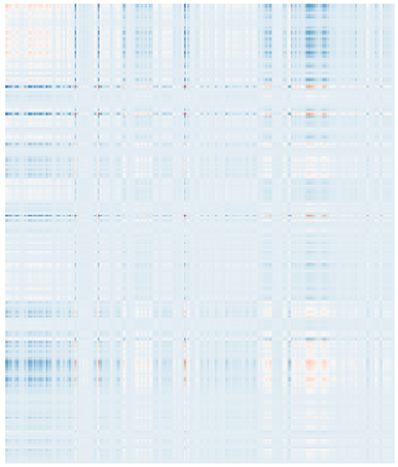
=

77.47%



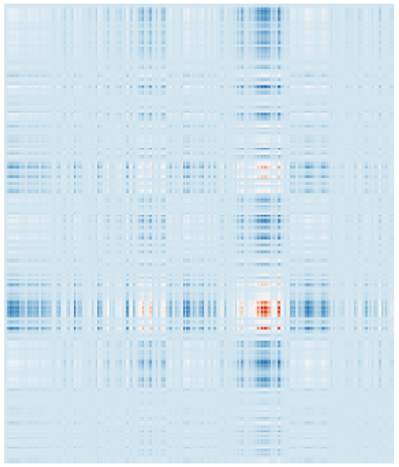
+

48.53%



+

13.86%

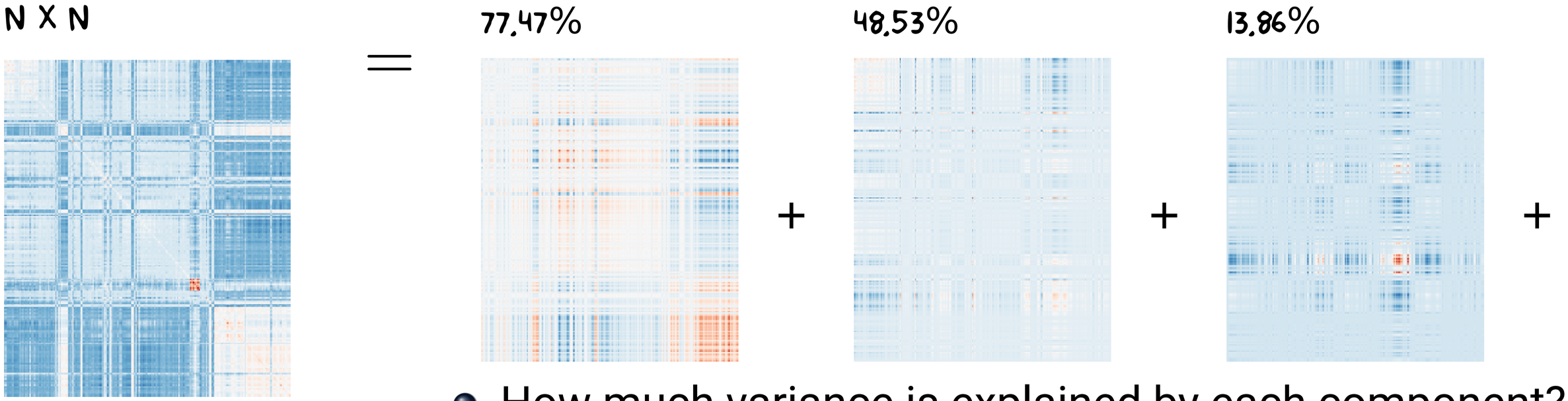


+

- How much variance is explained by each component?

PCs (SVs) decompose the covariance matrix

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

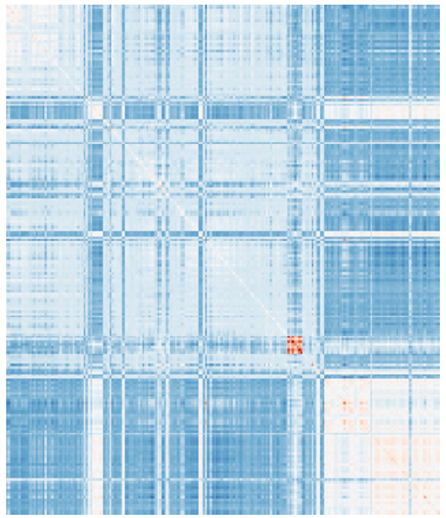


- How much variance is explained by each component?

PCs (SVs) decompose the covariance matrix

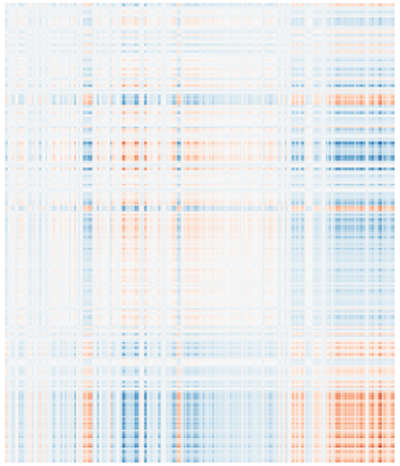
$$\hat{\Sigma} = X^\top X/m = V D^2 V^\top/m = \sum_{k=1} \lambda_k \mathbf{v}_k \mathbf{v}_k^\top, \quad \lambda_k = D_k^2/m$$

N X N



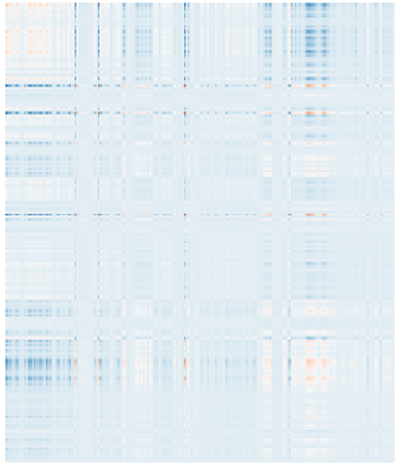
=

77.47%



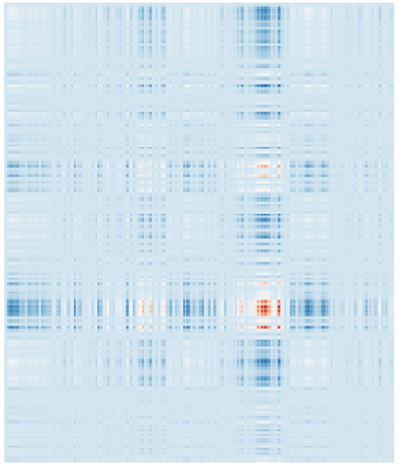
+

48.53%



+

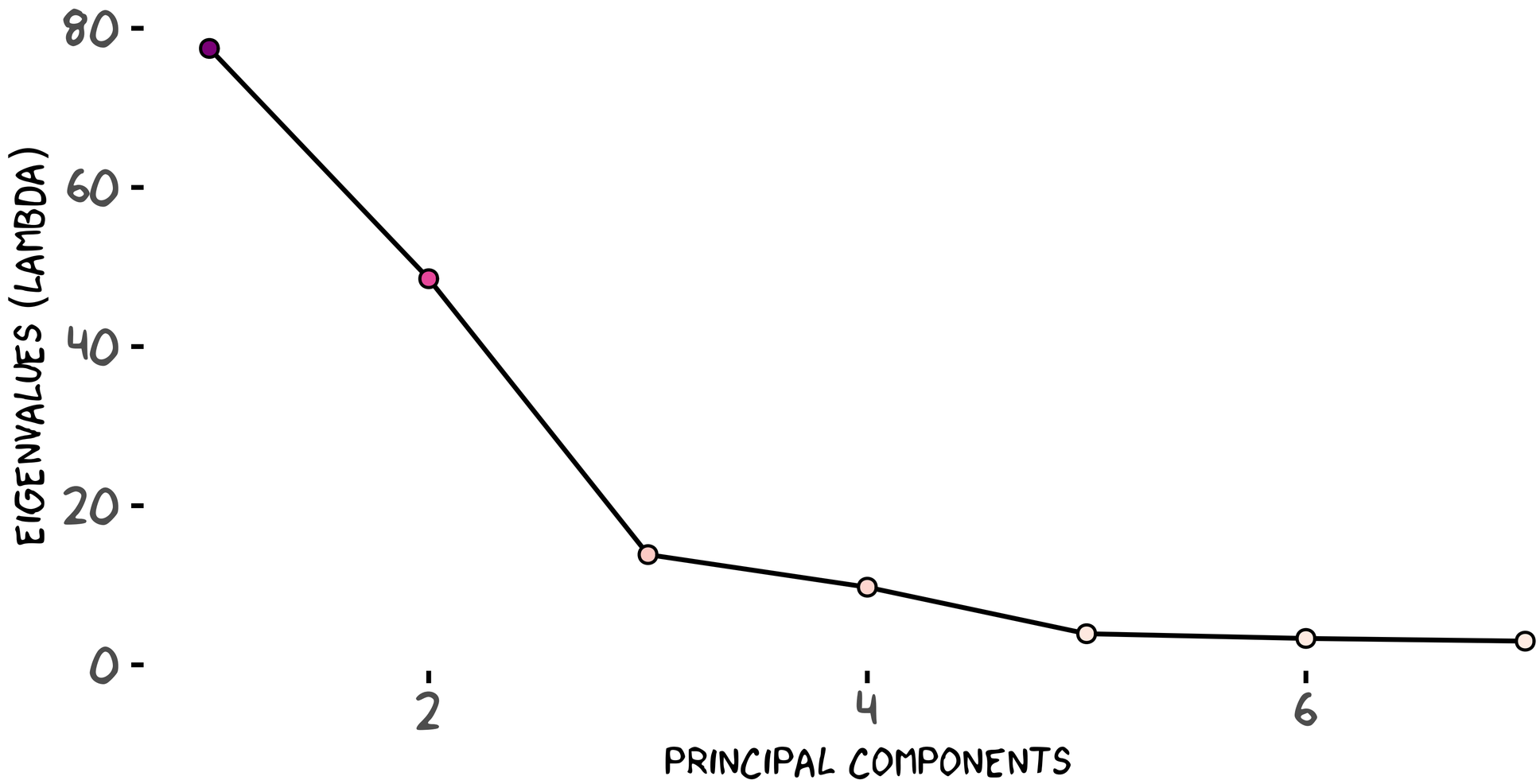
13.86%



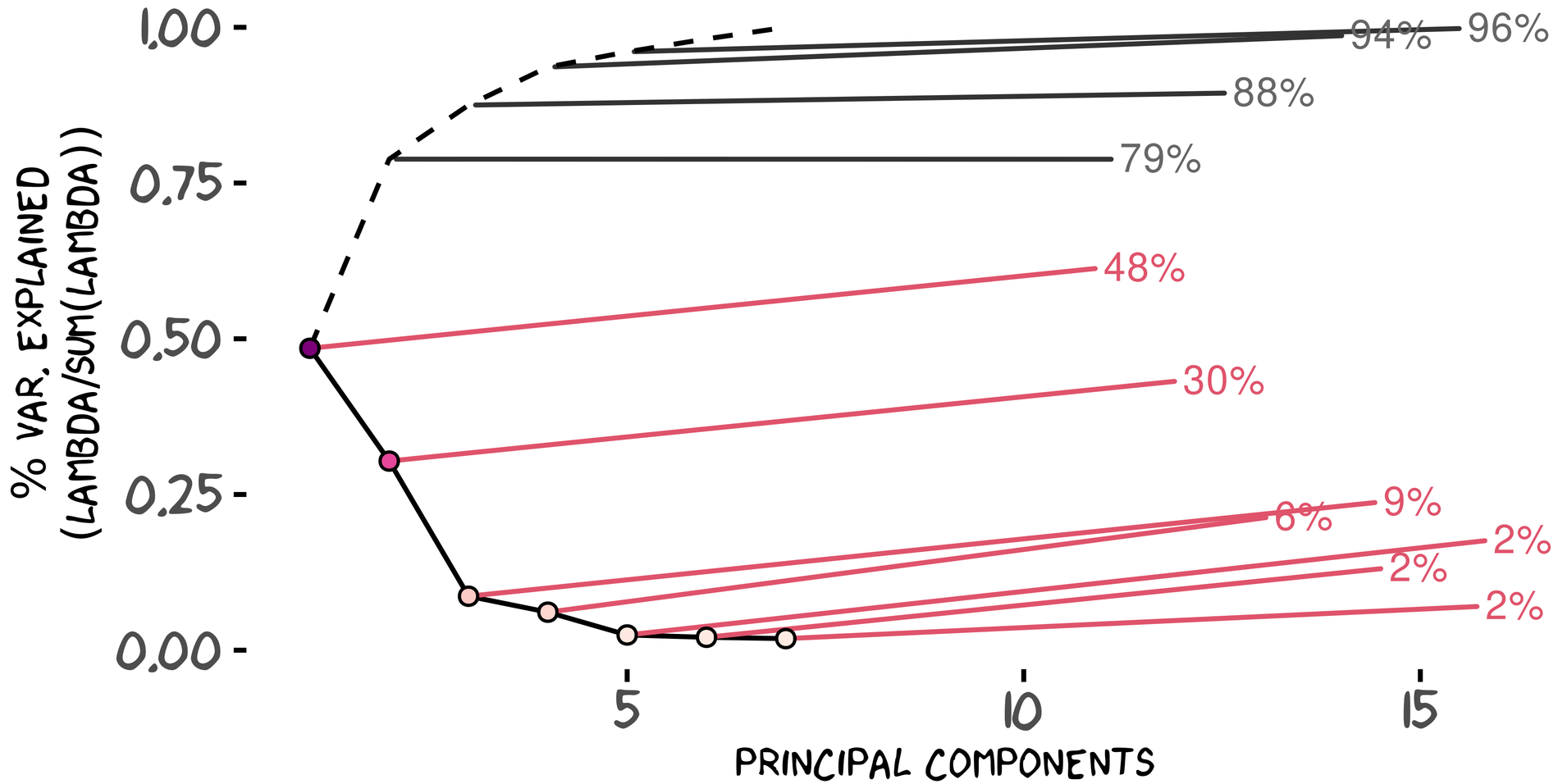
+

- How much variance is explained by each component?

How much variance is explained by each principal component?



How much variance is explained by each principal component?

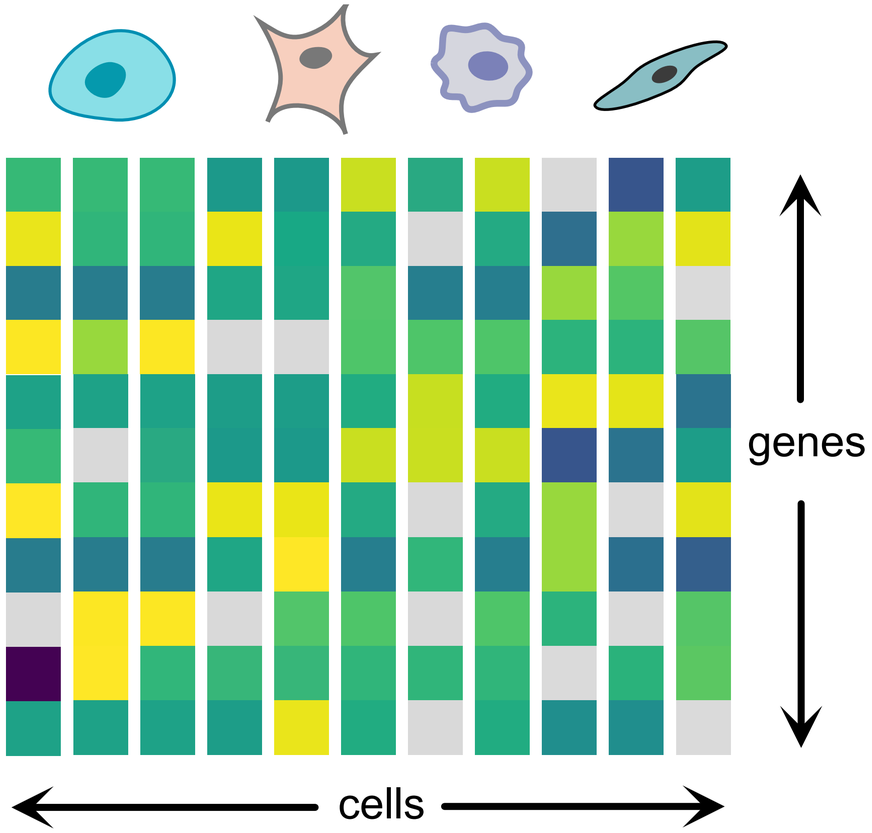


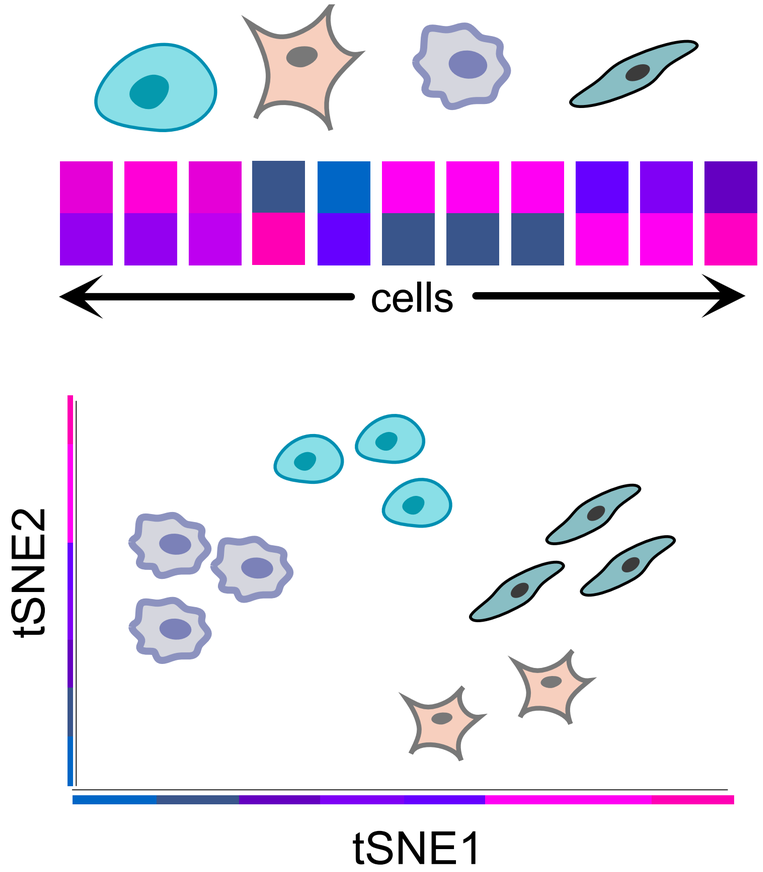
Today's lecture

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?

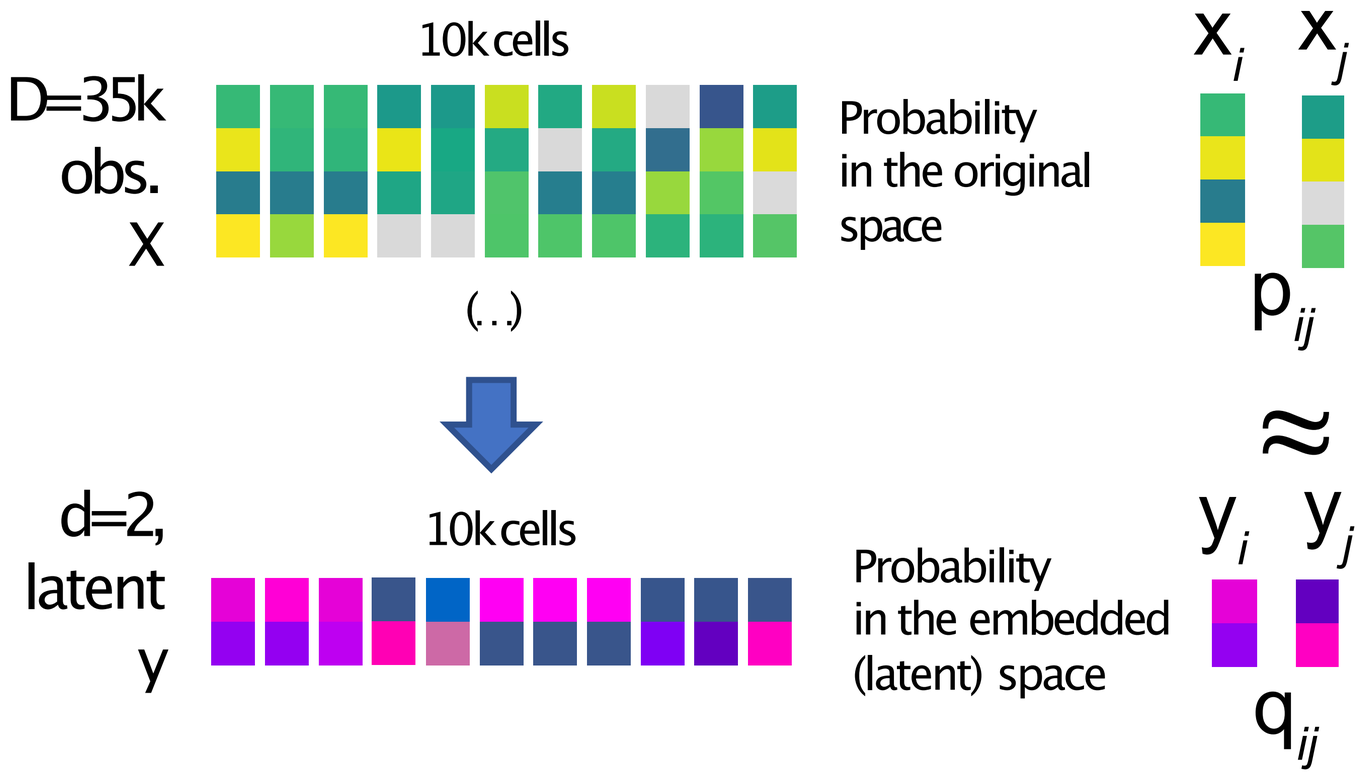
tSNE (and other similar variants) are fundamentally different from matrix factorization





`tSNE`: t-distributed Stochastic Neighbourhood Embedding (Van der Maaten & Hinton, 2008).

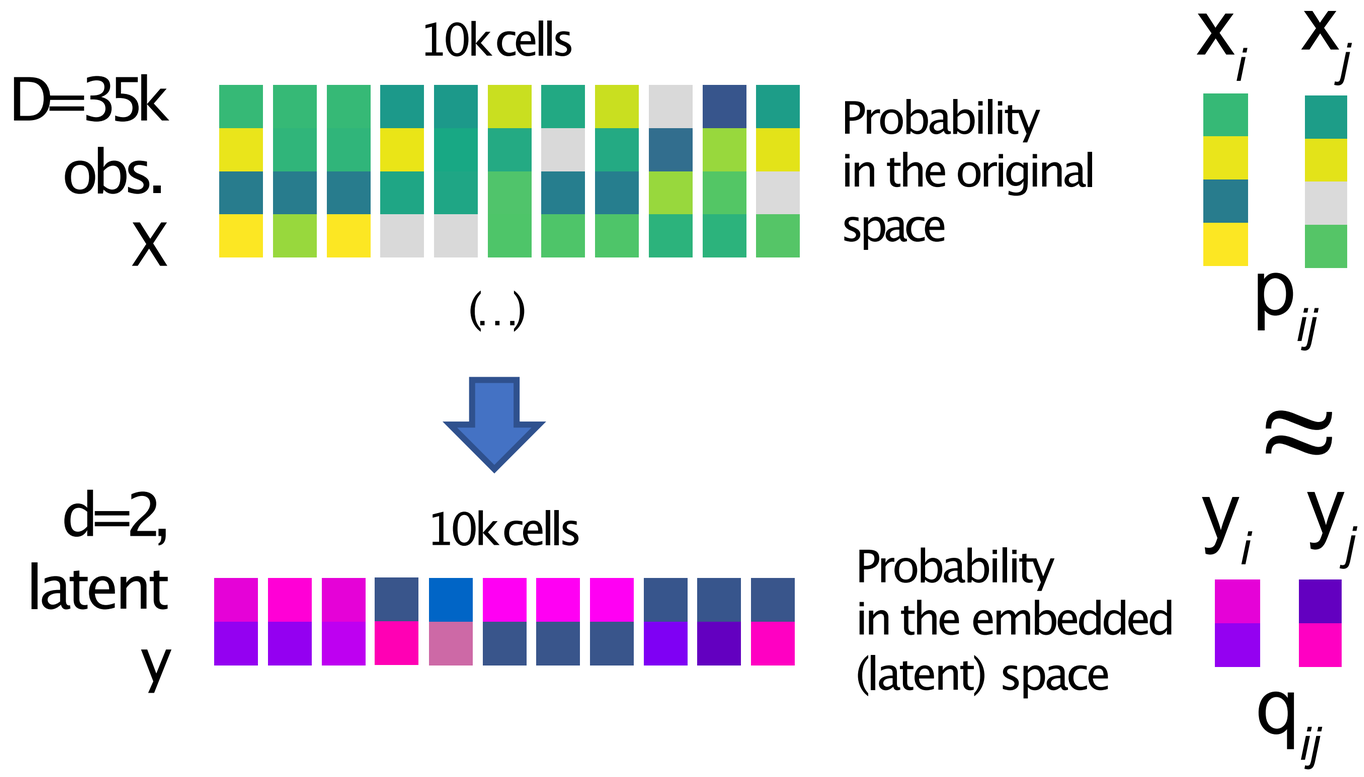
SNE: What is “stochastic neighbourhood embedding?”



- p_{ij}: probability between cells i and j in the original high-dimensional space

$$p_{ij} \propto \exp(-\|\mathbf{x}_i - \mathbf{x}_j\|^2 / 2\sigma^2)$$

SNE: What is “stochastic neighbourhood embedding?”



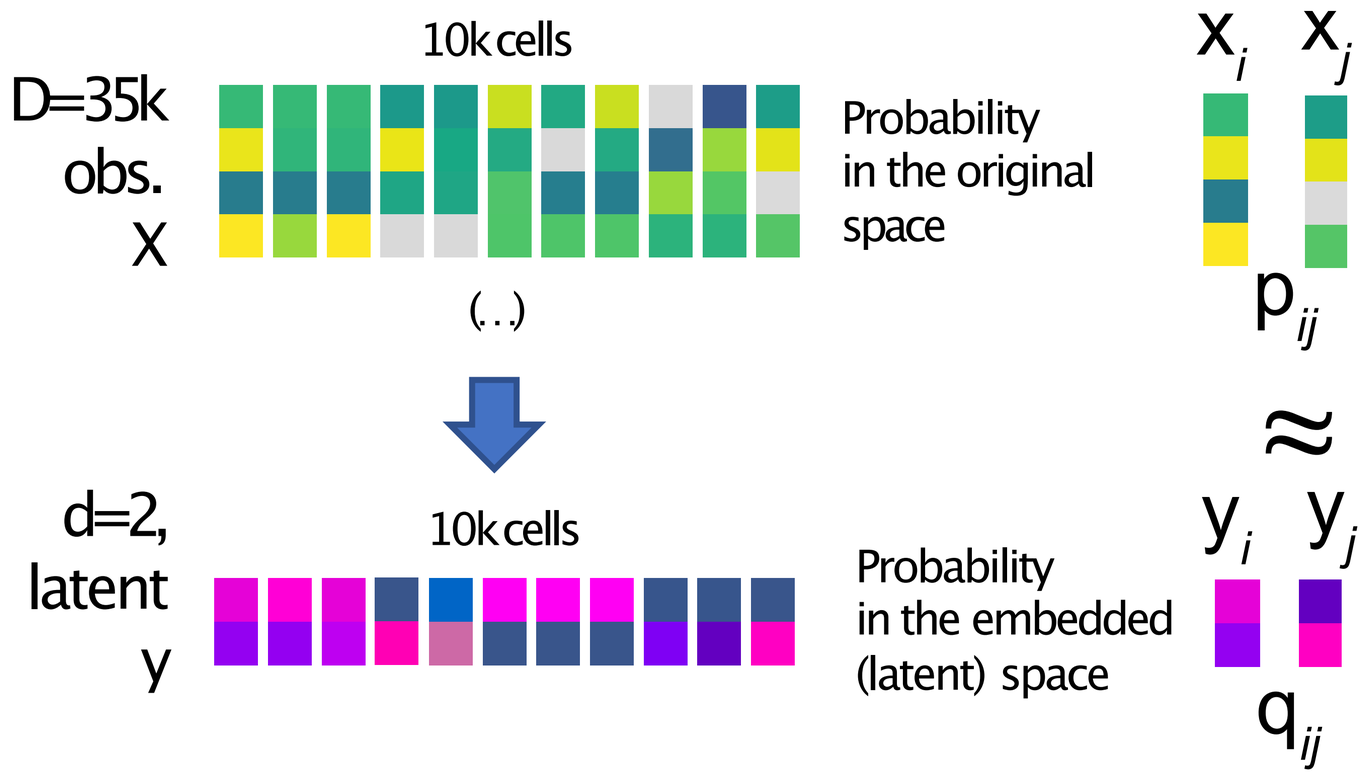
- p_{ij}: probability between cells i and j in the original high-dimensional space

$$p_{ij} \propto \exp(-\|\mathbf{x}_i - \mathbf{x}_j\|^2/2\sigma^2)$$

- q_{ij}: probability between cells i and j in the embedded low-dimensional space

$$q_{ij} \propto \exp(-\|\mathbf{y}_i - \mathbf{y}_j\|^2/2\sigma^2)$$

SNE: What is "stochastic neighbourhood embedding?"



- p_{ij}: probability between cells i and j in the original high-dimensional space

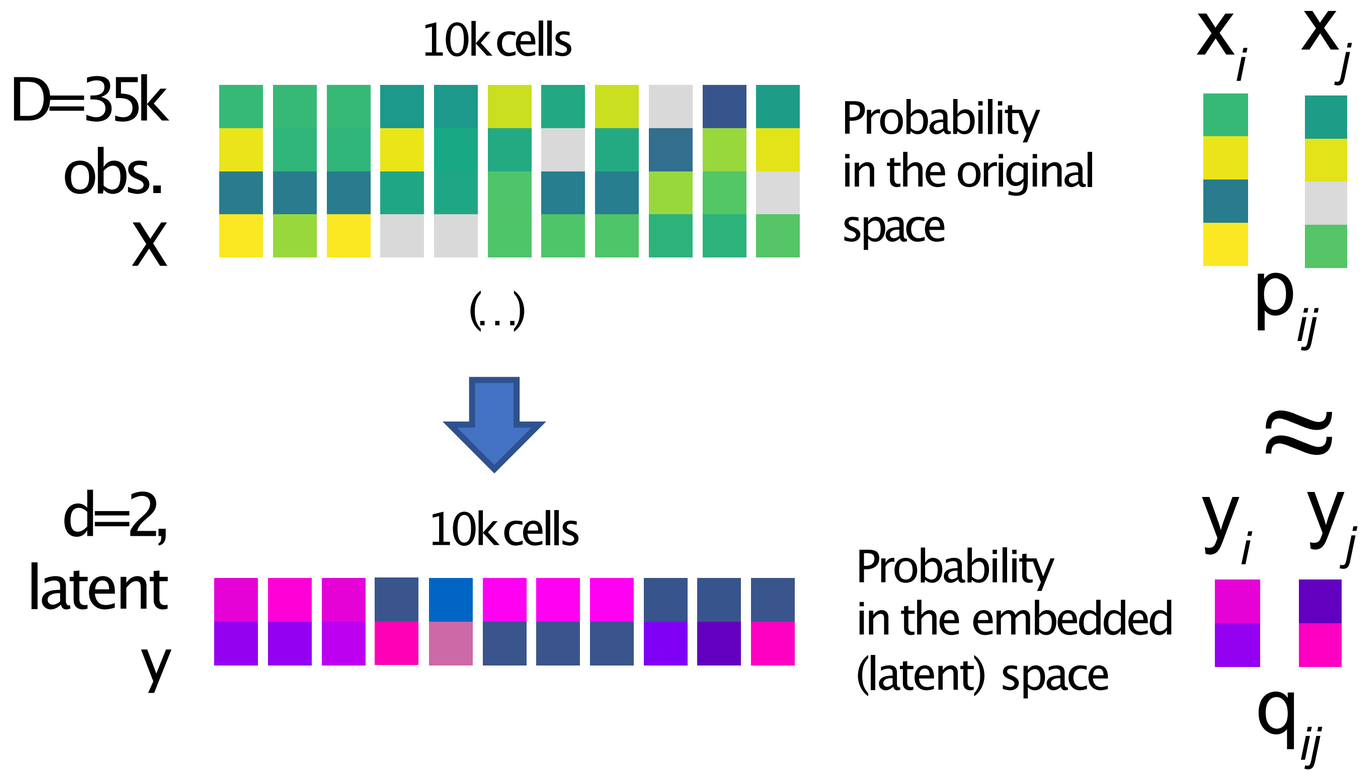
$$p_{ij} \propto \exp(-\|\mathbf{x}_i - \mathbf{x}_j\|^2/2\sigma^2)$$

- q_{ij}: probability between cells i and j in the embedded low-dimensional space

$$q_{ij} \propto \exp(-\|\mathbf{y}_i - \mathbf{y}_j\|^2/2\sigma^2)$$

Goal: make pairwise probabilities between cells in the observed and latent space as close as possible.

SNE: What is “stochastic neighbourhood embedding?”



- p_{ij}: probability between cells i and j in the original high-dimensional space

$$p_{ij} \propto \exp(-\|\mathbf{x}_i - \mathbf{x}_j\|^2/2\sigma^2)$$

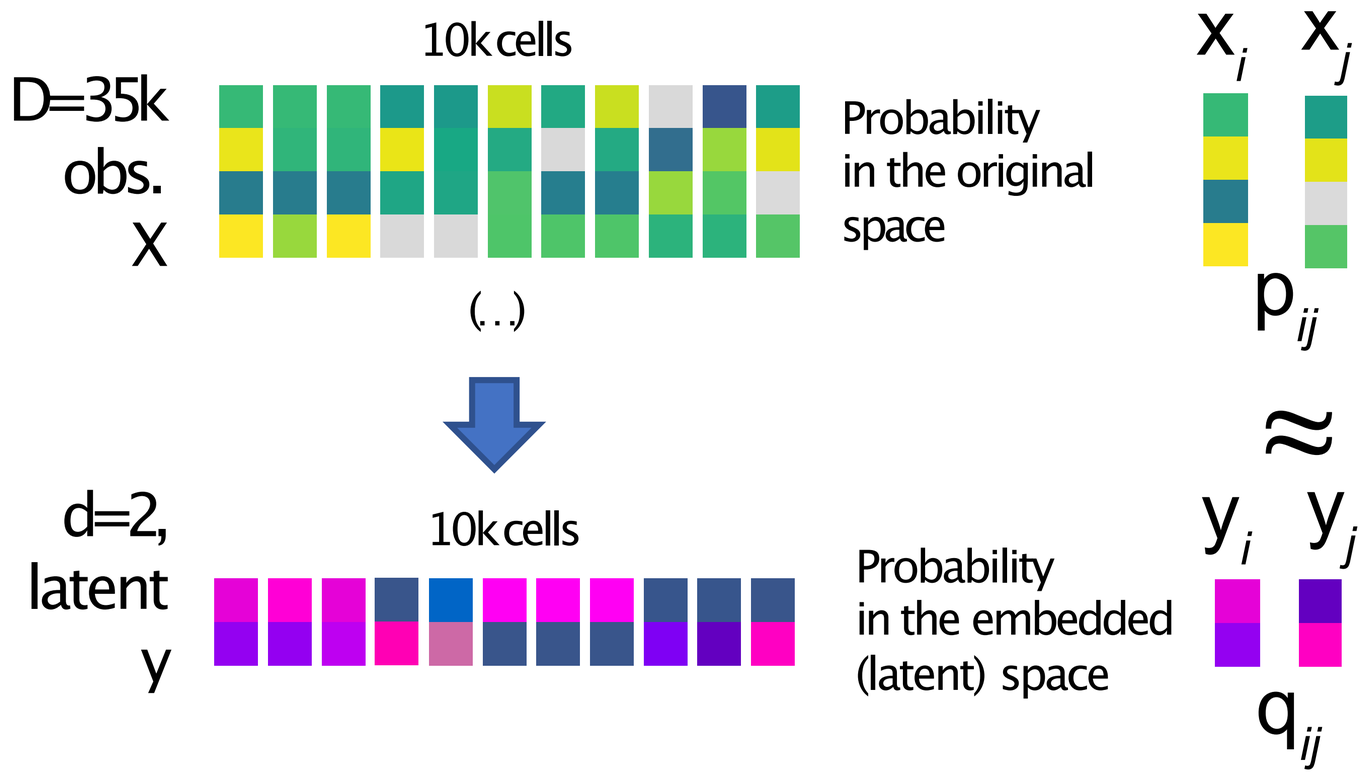
- q_{ij}: probability between cells i and j in the embedded low-dimensional space

$$q_{ij} \propto \exp(-\|\mathbf{y}_i - \mathbf{y}_j\|^2/2\sigma^2)$$

Goal: make pairwise probabilities between cells in the observed and latent space as close as possible.

$$\min D_{\mathsf{KL}}\left(p_{ij}\|q_{ij}\right) = \sum_{ij} p_{ij} \log \frac{p_{ij}}{q_{ij}}$$

tSNE: What is t-distributed "stochastic neighbourhood embedding?"



- p_{ij}: probability between cells i and j in the original high-dimensional space

$$p_{ij} \propto \exp(-\|\mathbf{x}_i - \mathbf{x}_j\|^2/2\sigma^2)$$

- q_{ij}: probability between cells i and j in the embedded low-dimensional space

$$q_{ij} \propto \left(1 + \|\mathbf{y}_i - \mathbf{y}_j\|^2\right)^{-1}$$

Goal: make pairwise probabilities between cells in the observed and latent space as close as possible.

$$\min D_{\mathsf{KL}}\left(p_{ij} \| q_{ij}\right) = \sum_{ij} p_{ij} \log \frac{p_{ij}}{q_{ij}}$$

Today's lecture

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

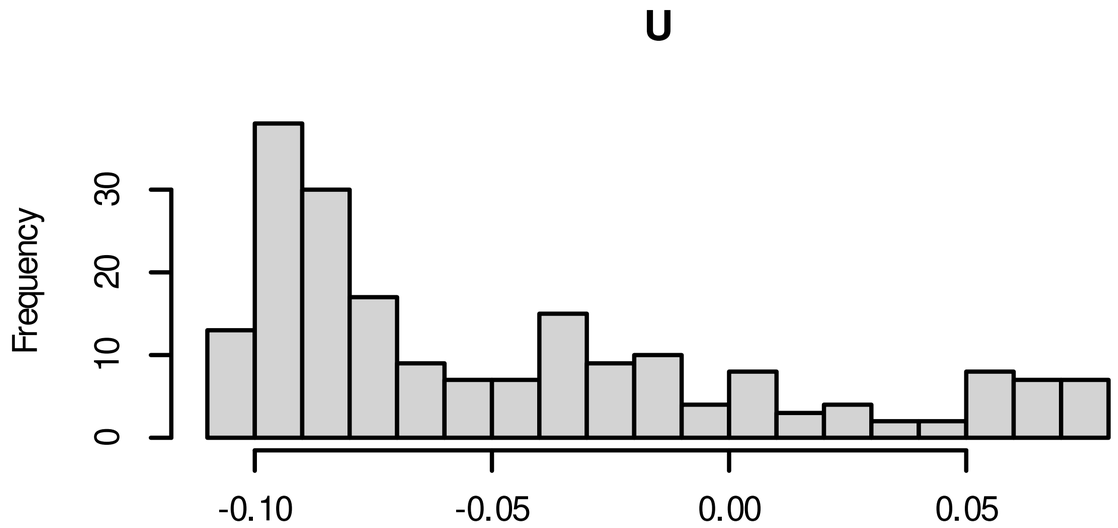
Why another factorization method?

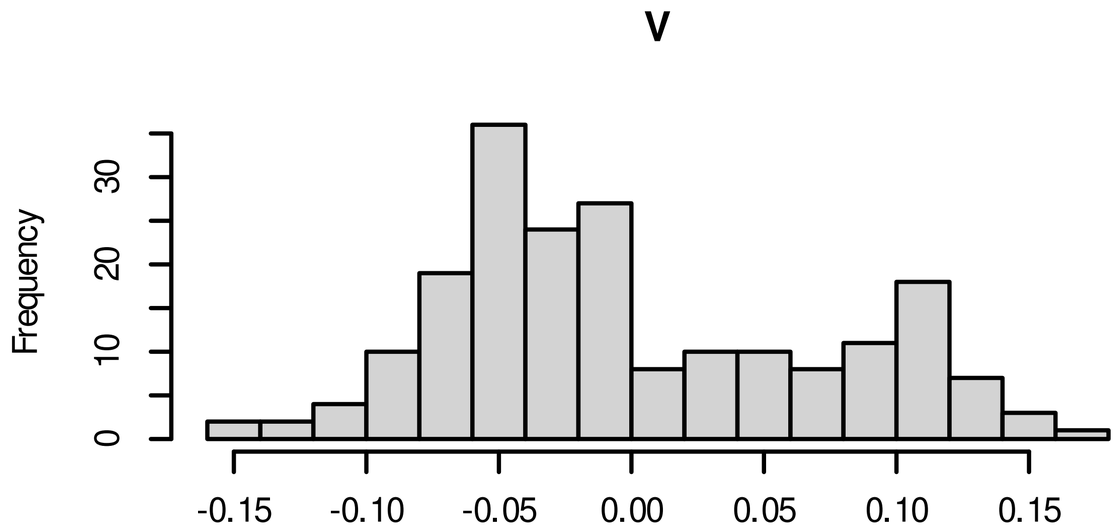
- The count matrix X is non-negative.

```
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.

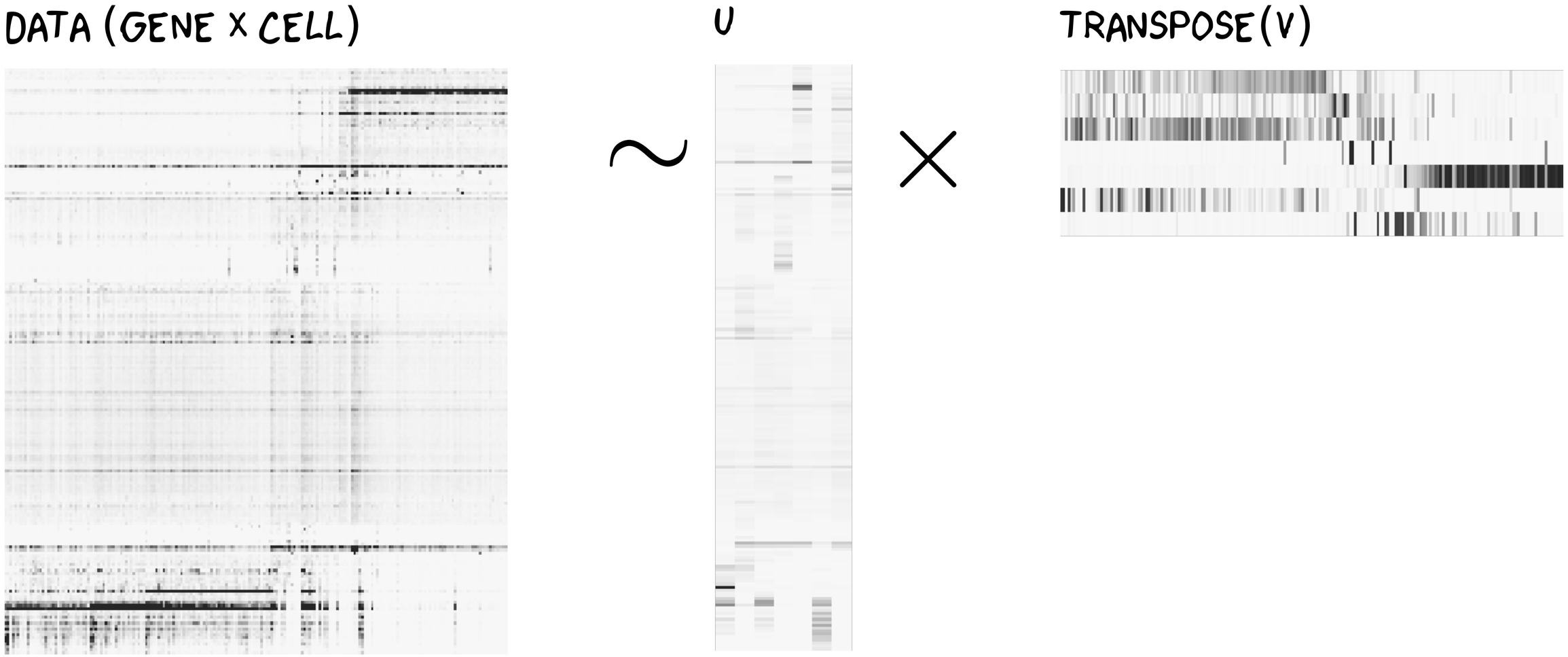




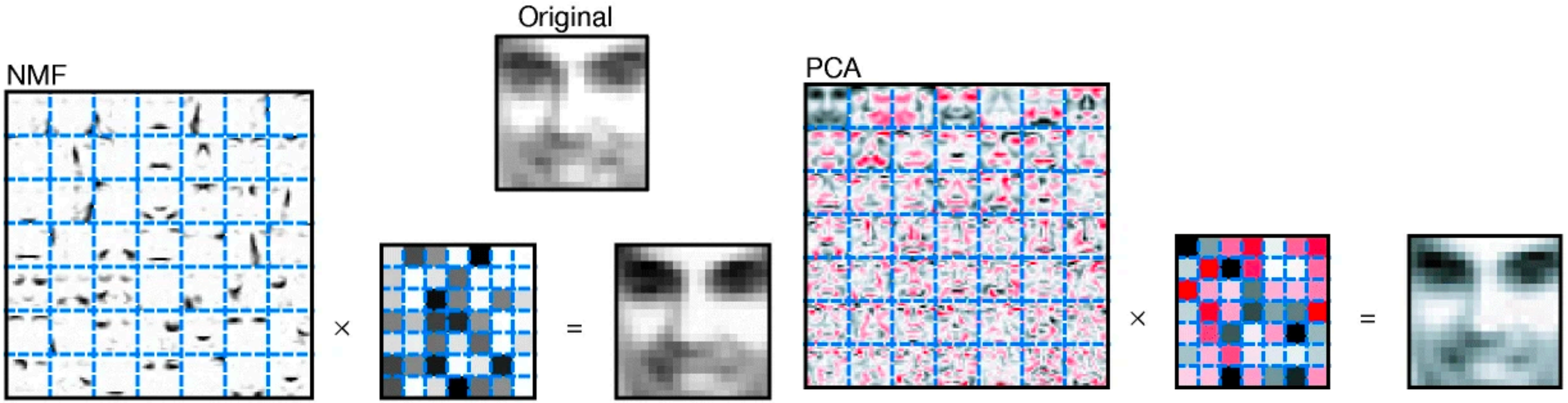
Non-negative factorization

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

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



Why another factorization method?

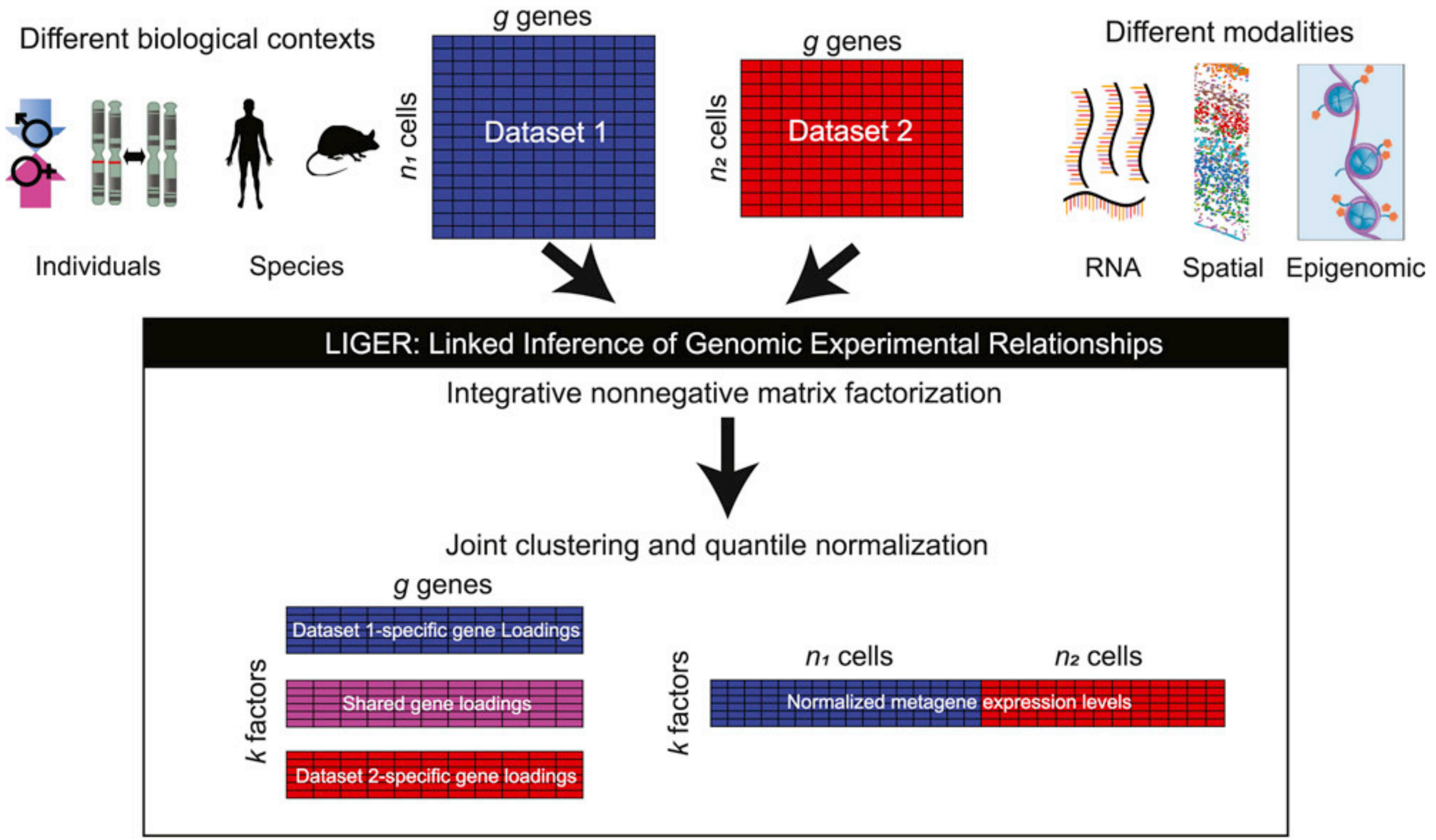


Decomposition by "parts"

Positive + negative can cancel each other

Lee and Seung, *Nature* (1999)

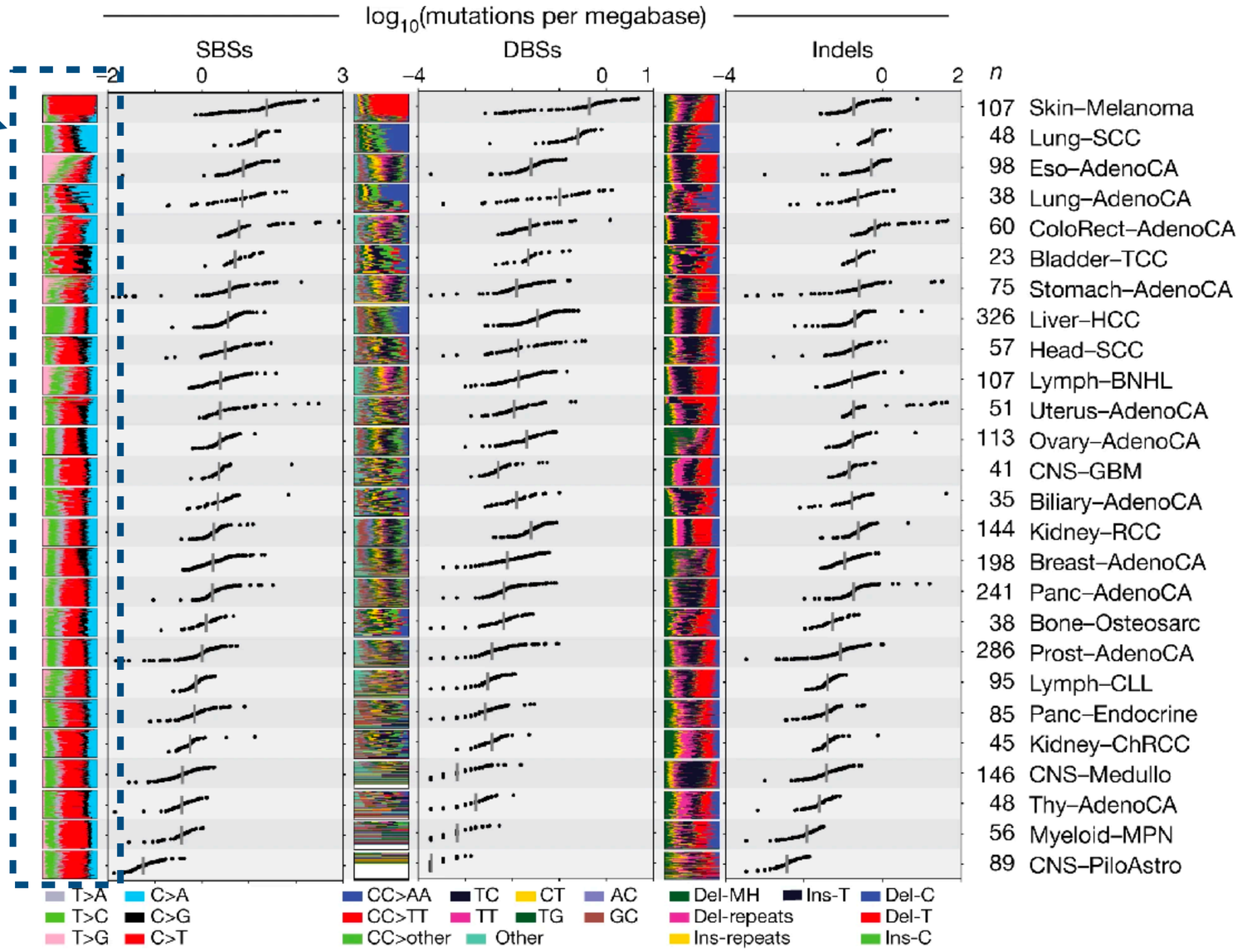
NMF-based method can facilitate data integration



Welch .. Macosko, *Cell* (2019)

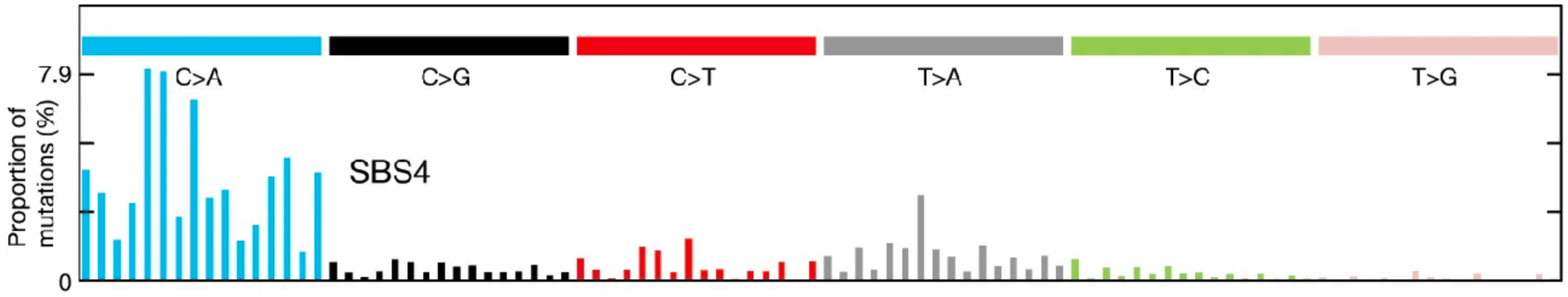
NMF can identify hidden patterns in high-dimensional count data

Proportion of mutation "topic" or "signatures" per tumour sample



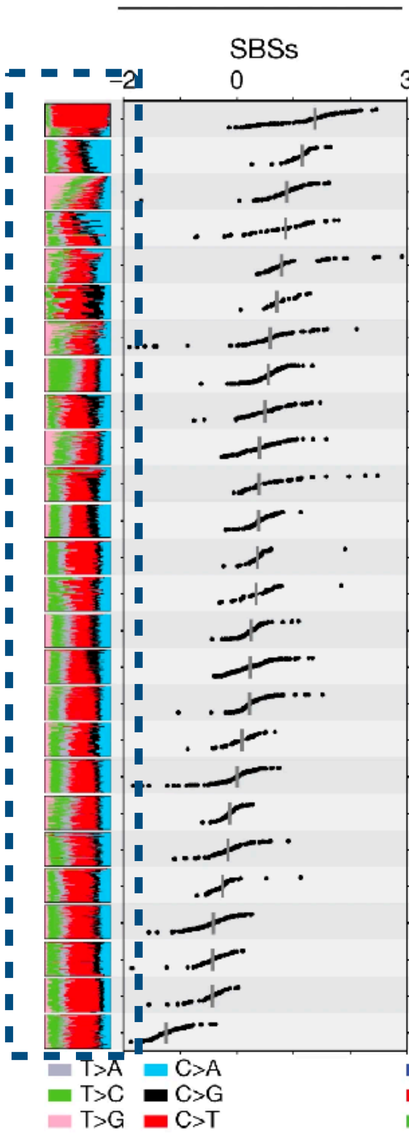
Alexandrov *et al.*, *Nature* (2020)

NMF can identify hidden patterns in high-dimensional count data



One of the single-base substitution signatures

Proportion of mutation "topic" or "signatures" per tumour sample



Alexandrov *et al.*, *Nature* (2020)

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

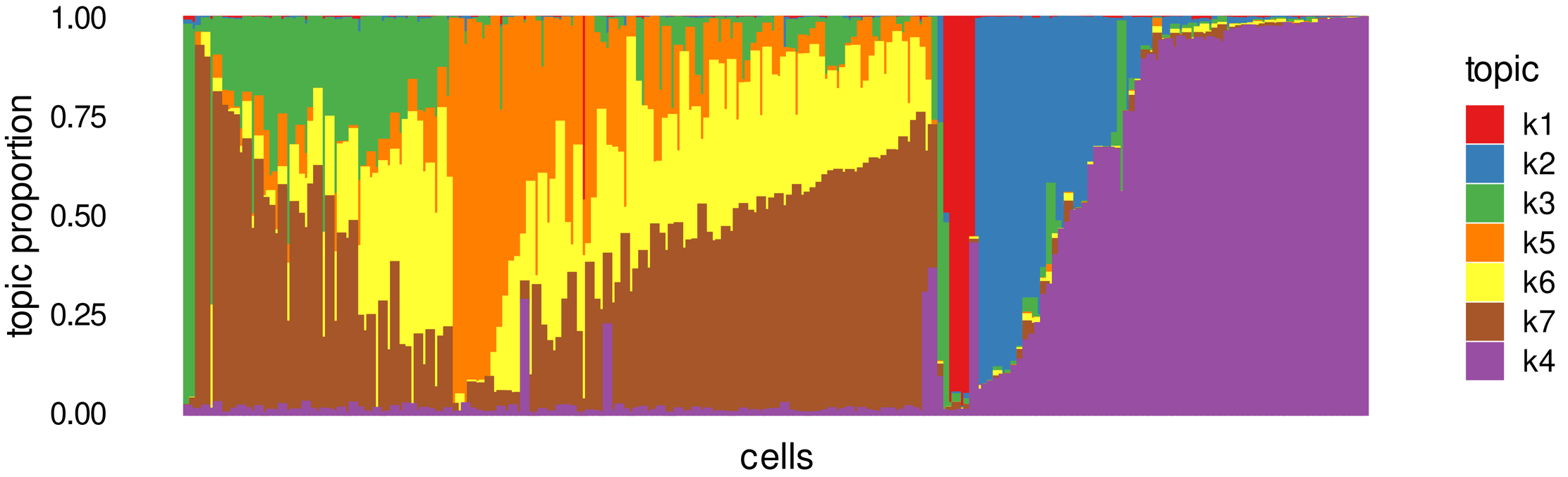
```
library(fastTopics)
out <- fit_topic_model(t(xx), k = 7)
```

```
## 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

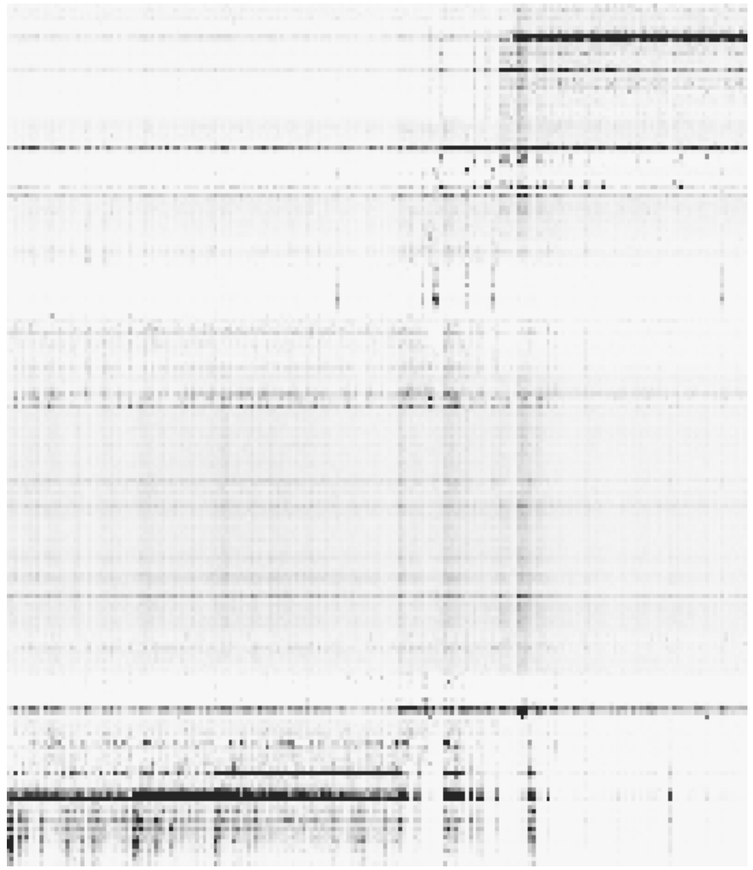
```
structure_plot(out, verbose = F) + xlab("cells")
```



Carbonetto .. Stephens, Genome Biology (2023)

Poisson NMF by `fastTopics`

DATA (GENE x CELL)

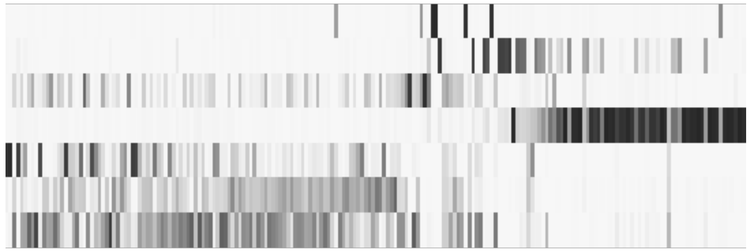


$\sim$

GENE x F

$\times$

FACTOR x CELL



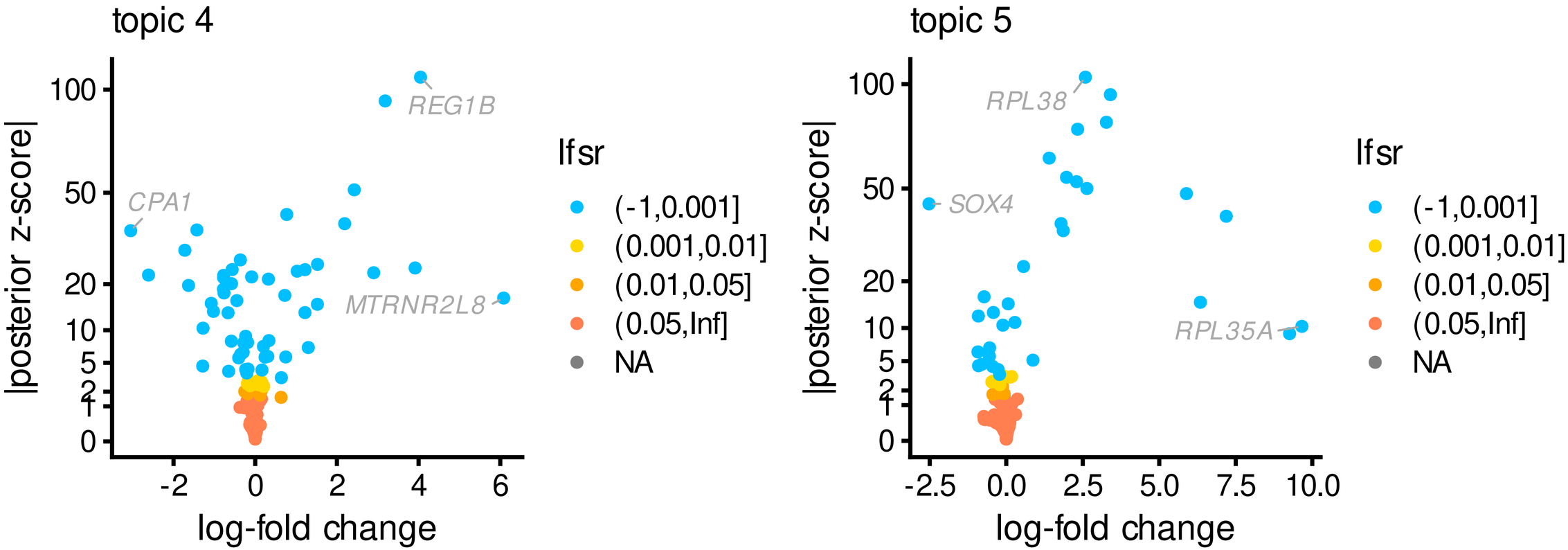
`fastTopics`' differential expression analysis to find topic-specific genes

```
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



`rliger` provides a more scalable option for NMF estimation

```
library(rliger)
.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)
```

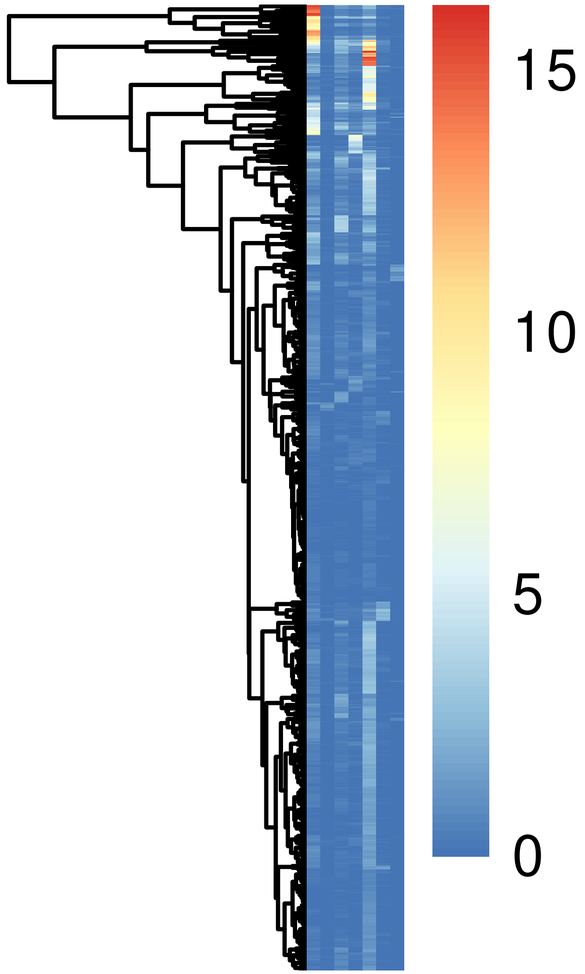
```
## Starting Online iNMF...
##   |                                                                      |
```

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

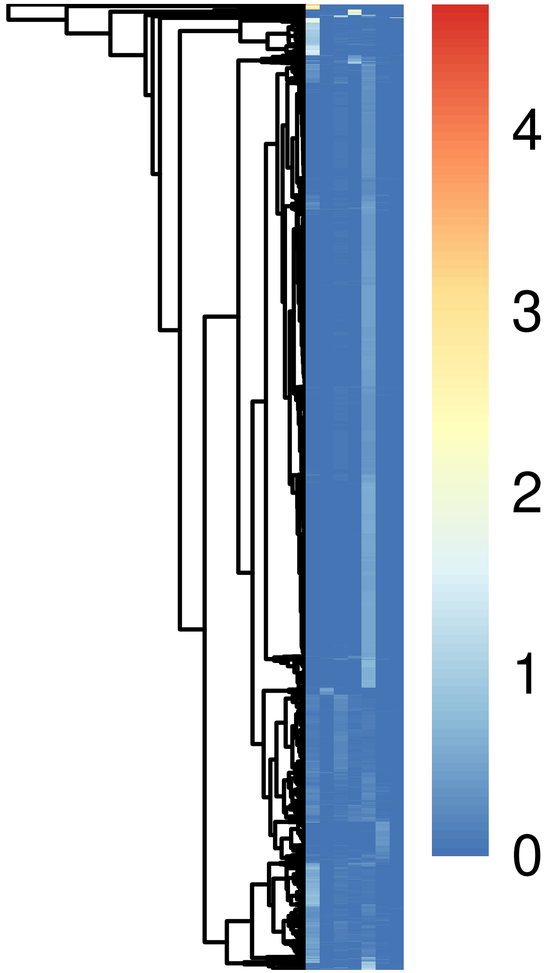
Gao .. Welch, Nature Biotech. (2021)

`rliger`'s NMF result

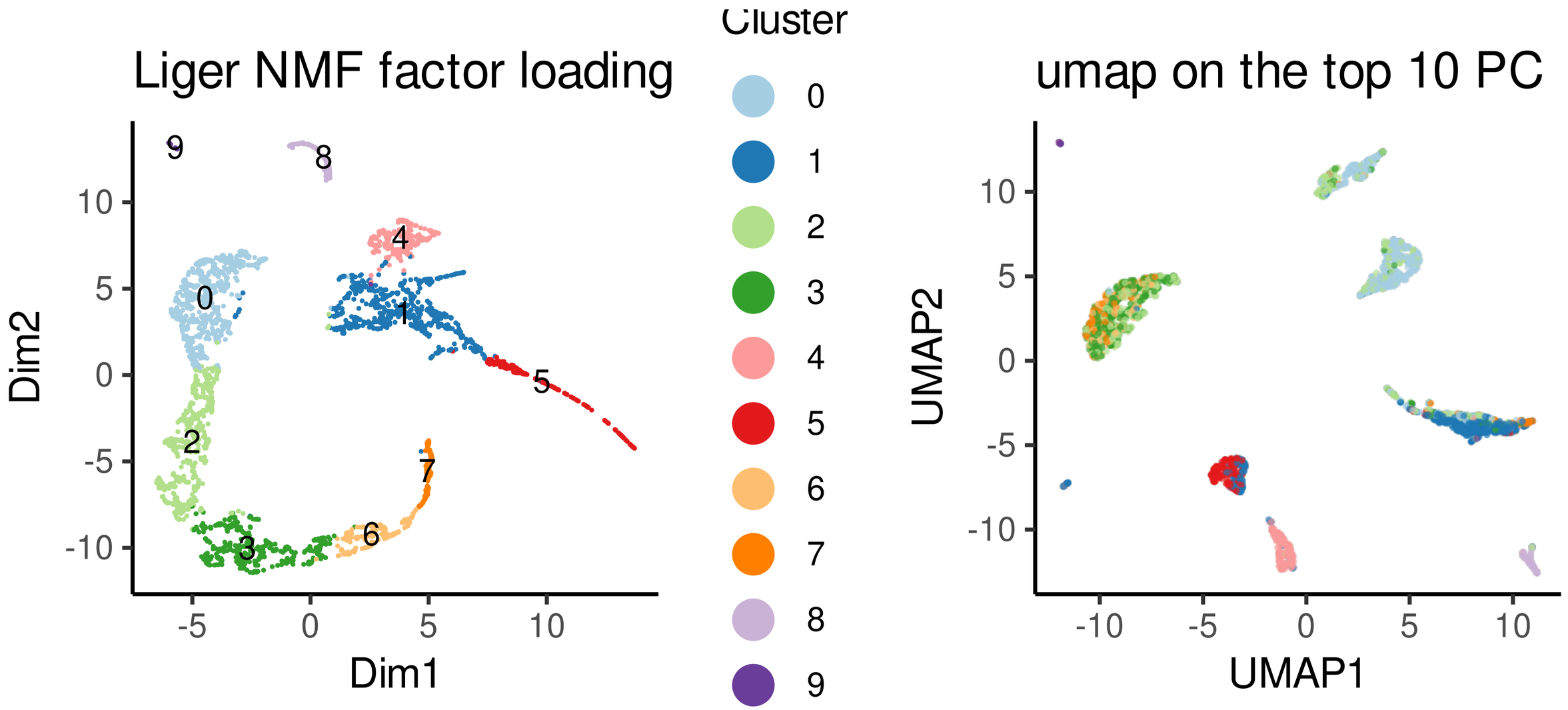
gene × factor matrix



cell × factor matrix

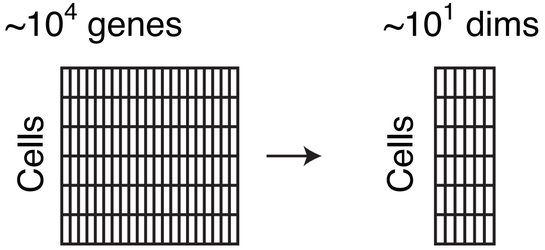


`rliger`: clustering cells by the topic/factor loading

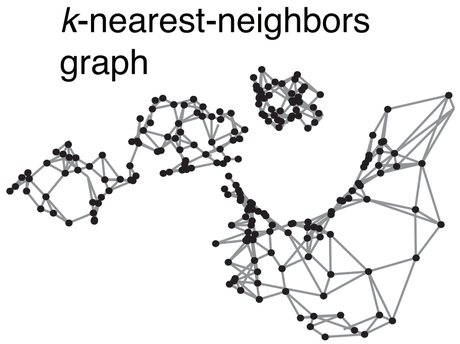


What's next after clustering?

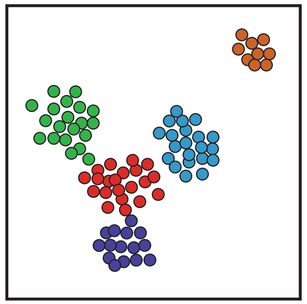
Reduction to a medium-dimensional space



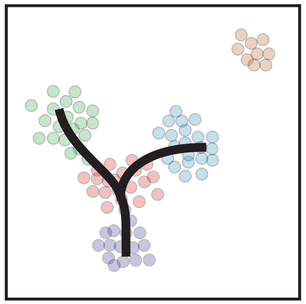
Manifold representation



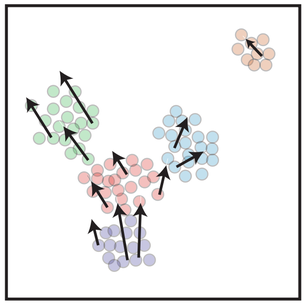
Clustering and differential expression



Trajectories

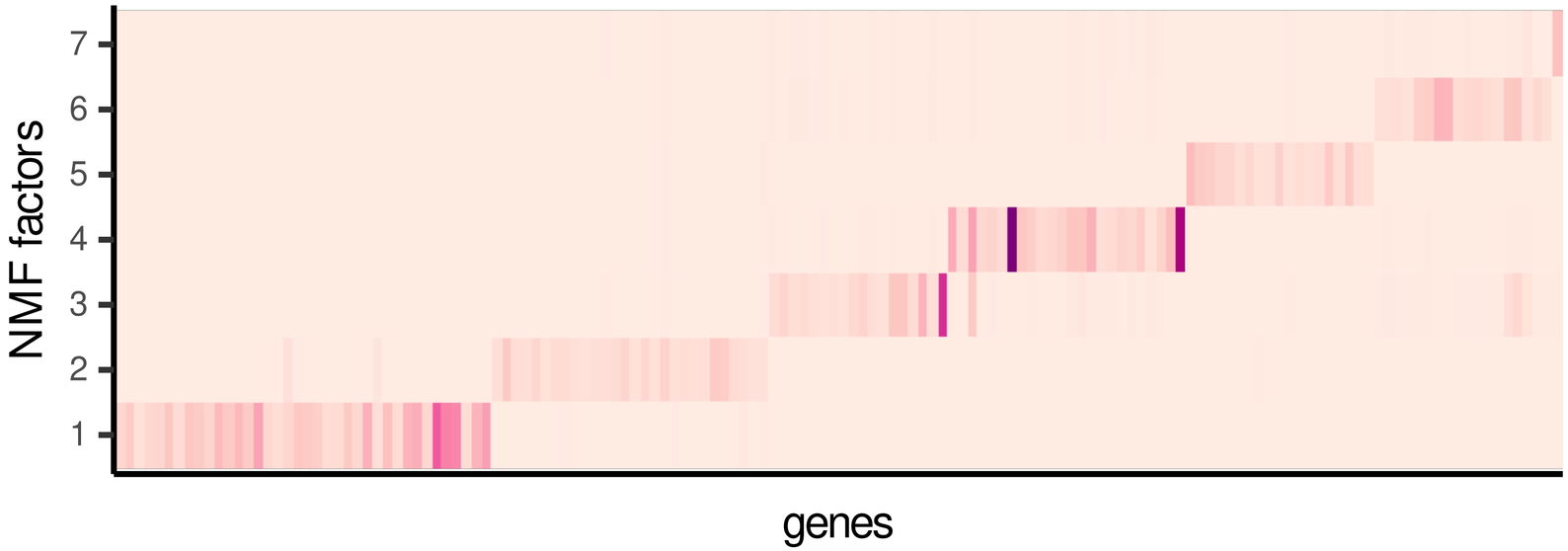


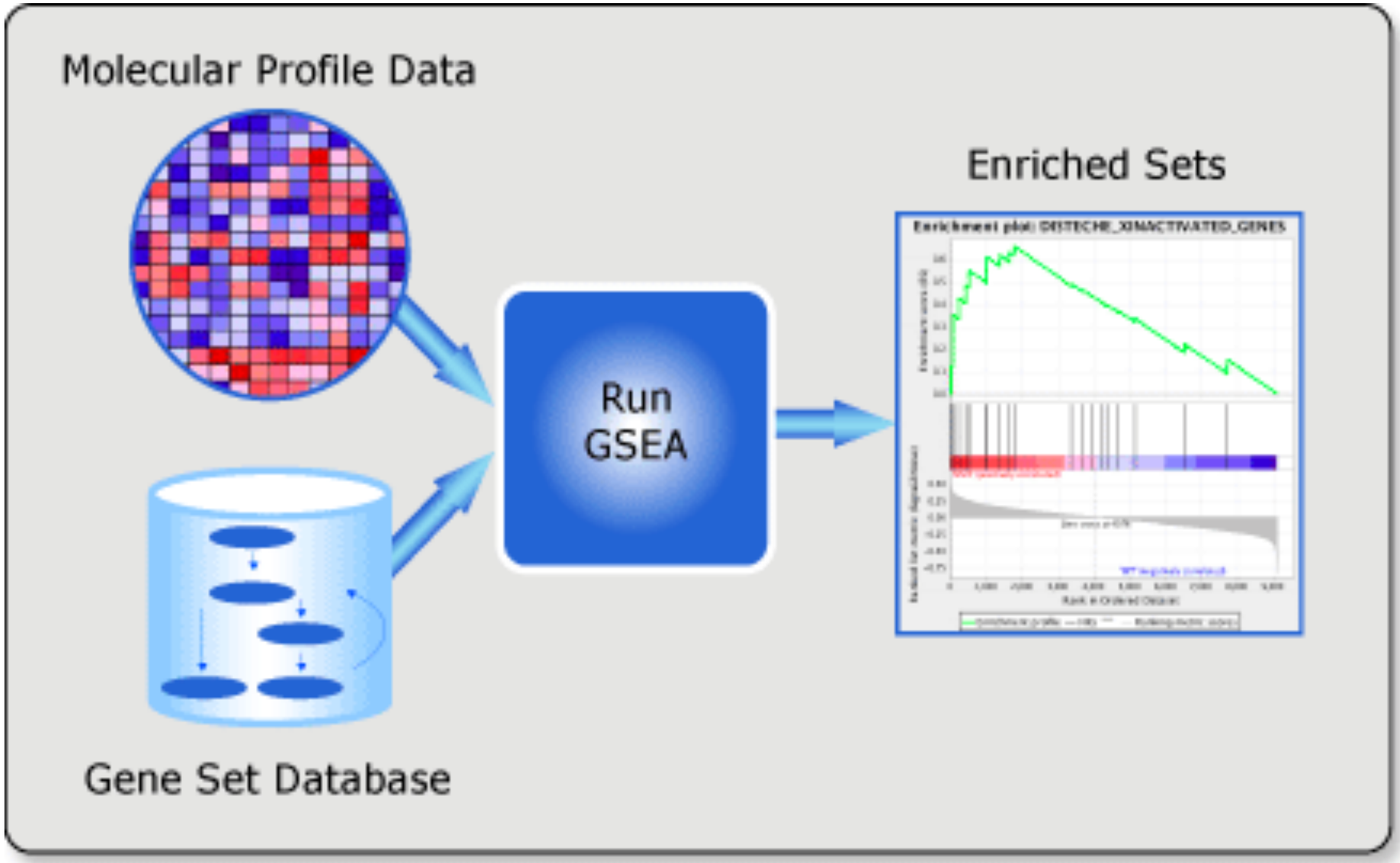
Velocity estimation



Karchenko, *Nature Methods* (2021)

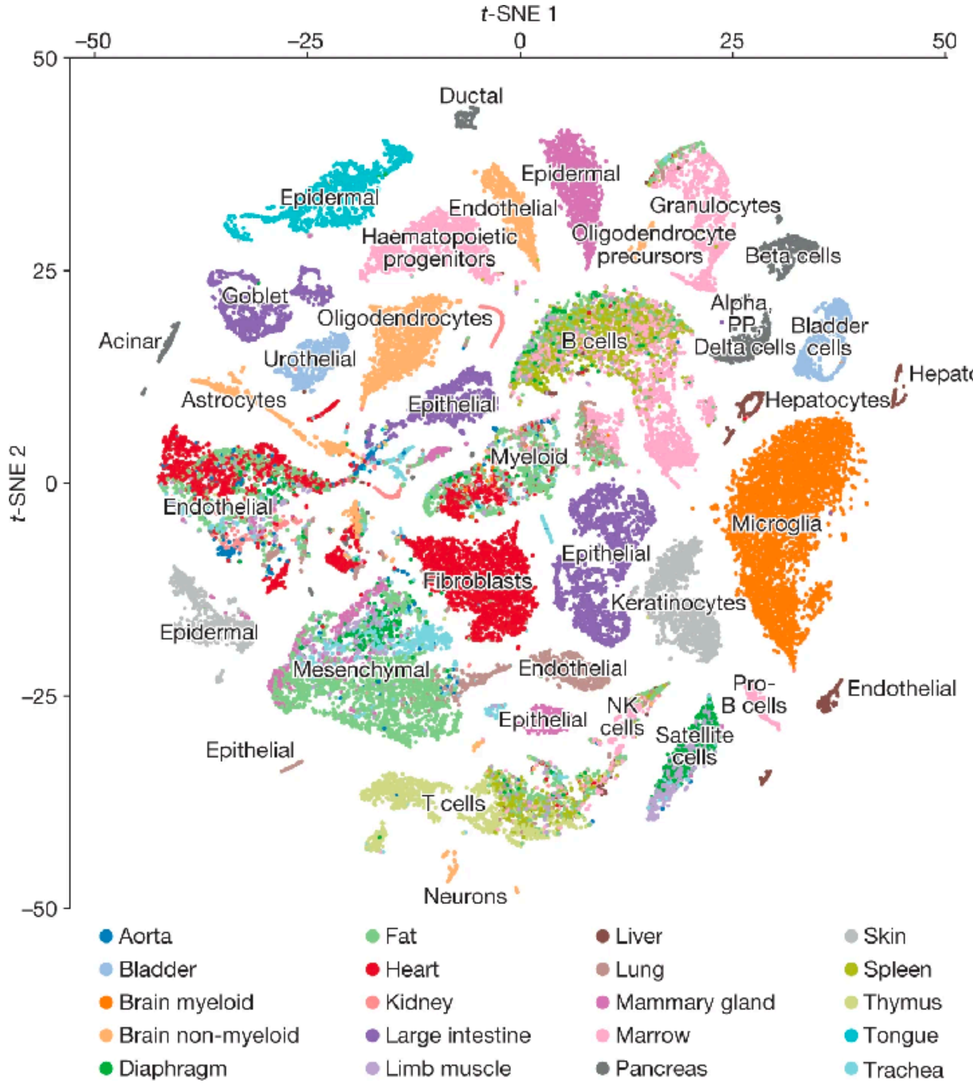
Once we have gene × factor dictionary

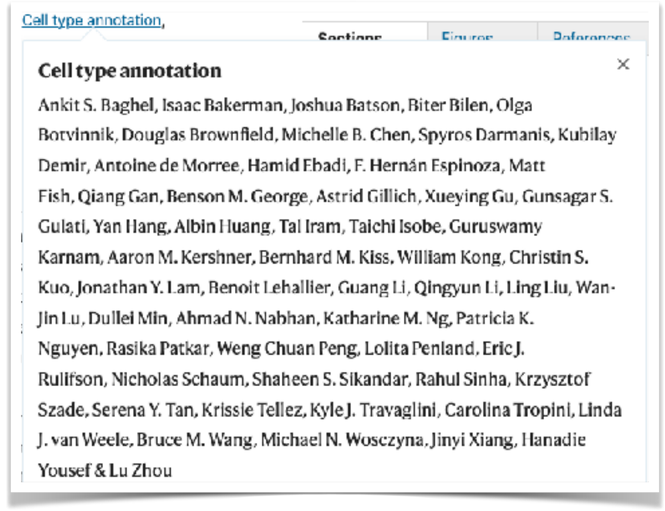




https://www.gsea-msigdb.org/gsea/index.jsp

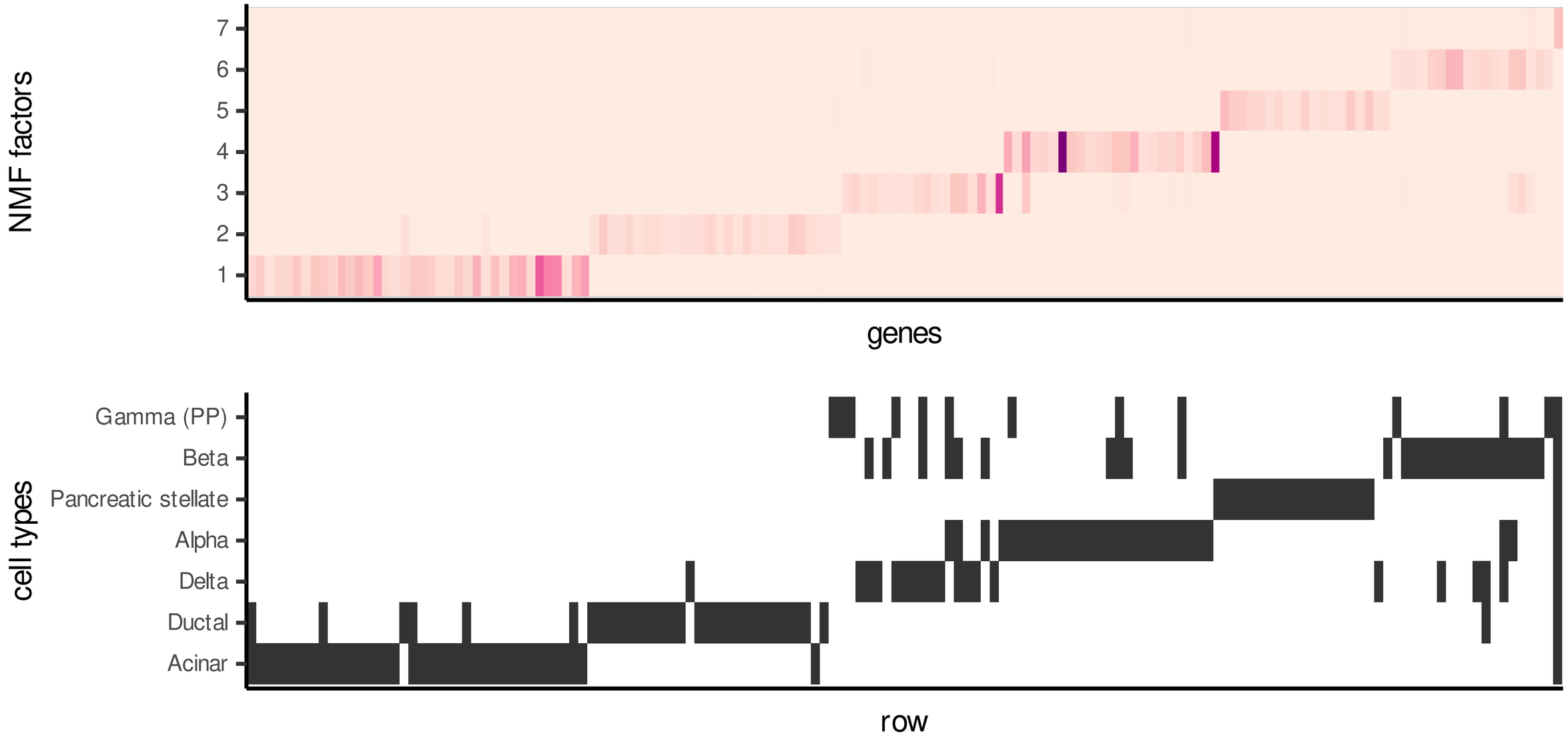
How do we know clusters of cells correspond to cell types?



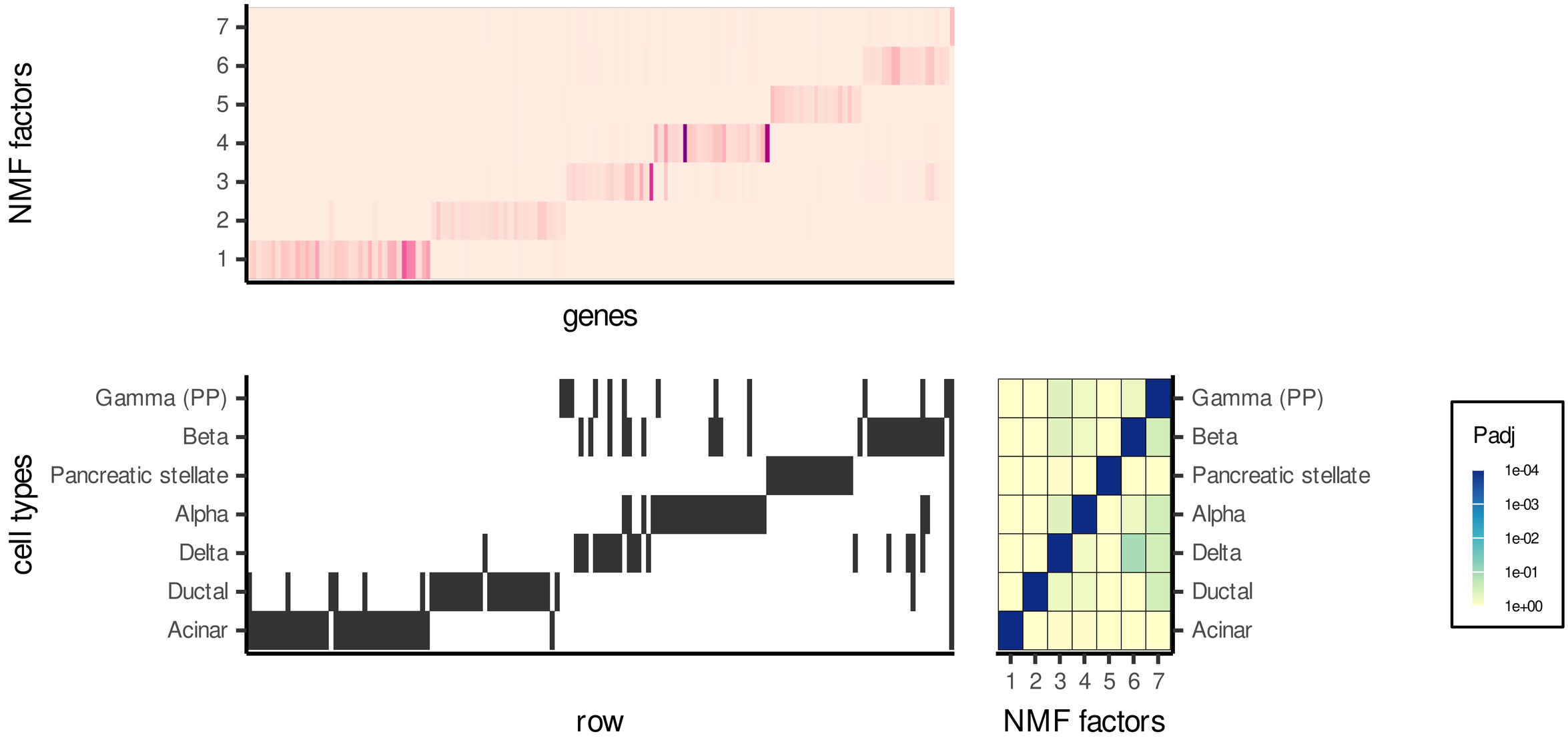


Tabular Muris, *Nature* (2018)

Annotate topics to cell types by enrichment (`fgsea`)



Annotate topics to cell types by enrichment (`fgsea`)

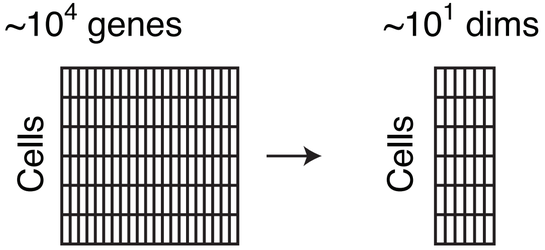


Today's lecture

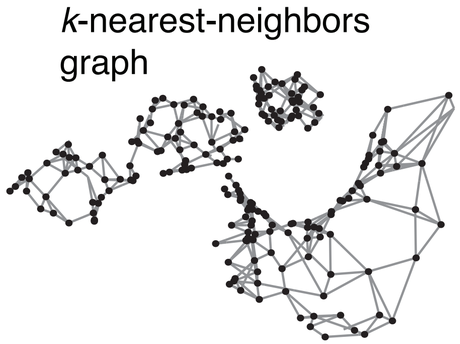
1. (review) Principal Component Analysis
2. Non-negative matrix factorization
3. Differential gene expression analysis

What's next after clustering?

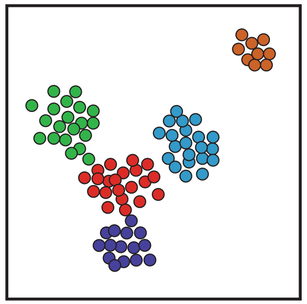
Reduction to a medium-dimensional space



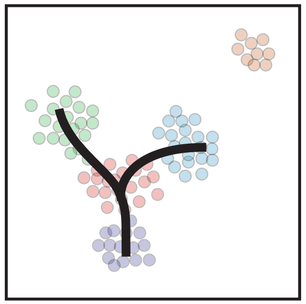
Manifold representation



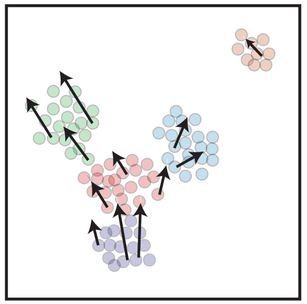
Clustering and differential expression



Trajectories



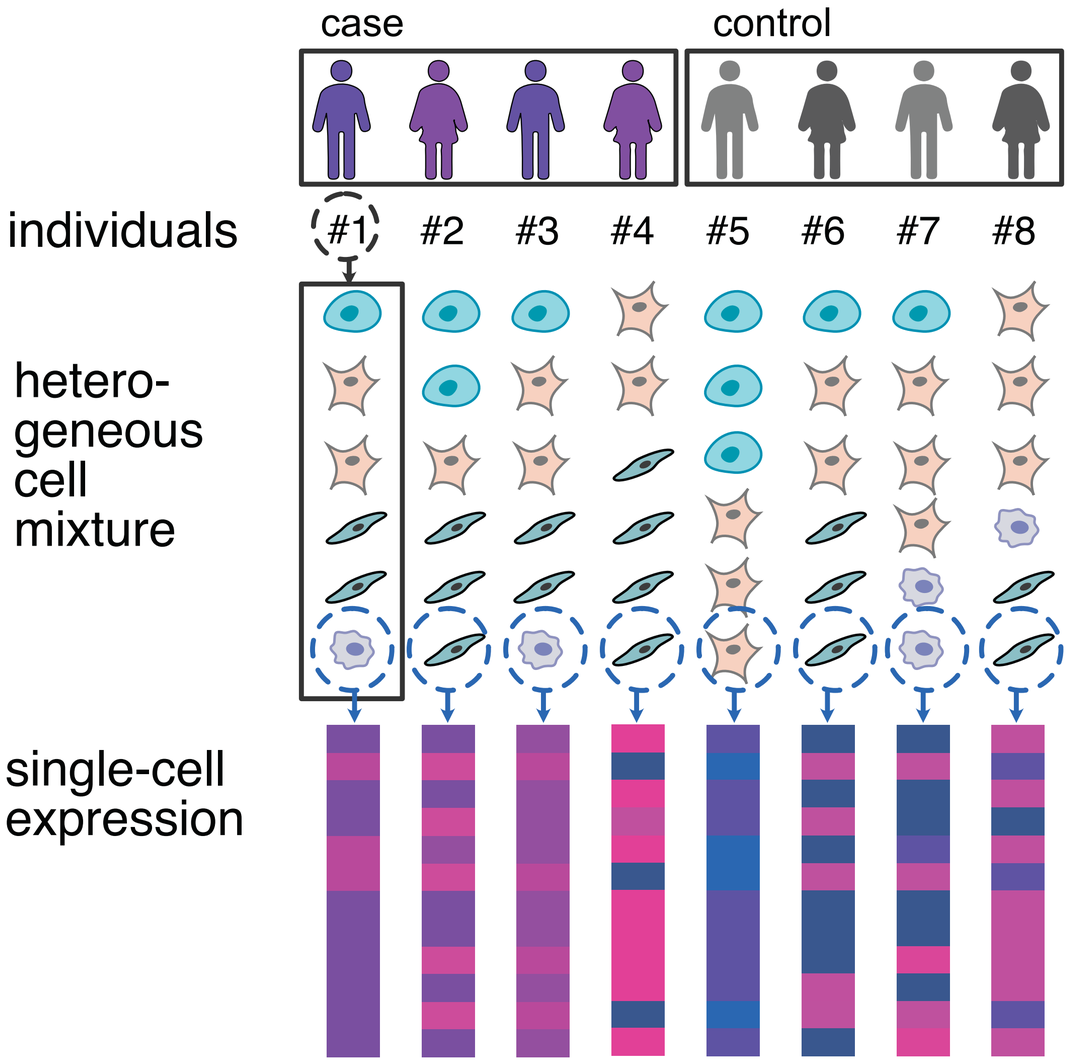
Velocity estimation



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

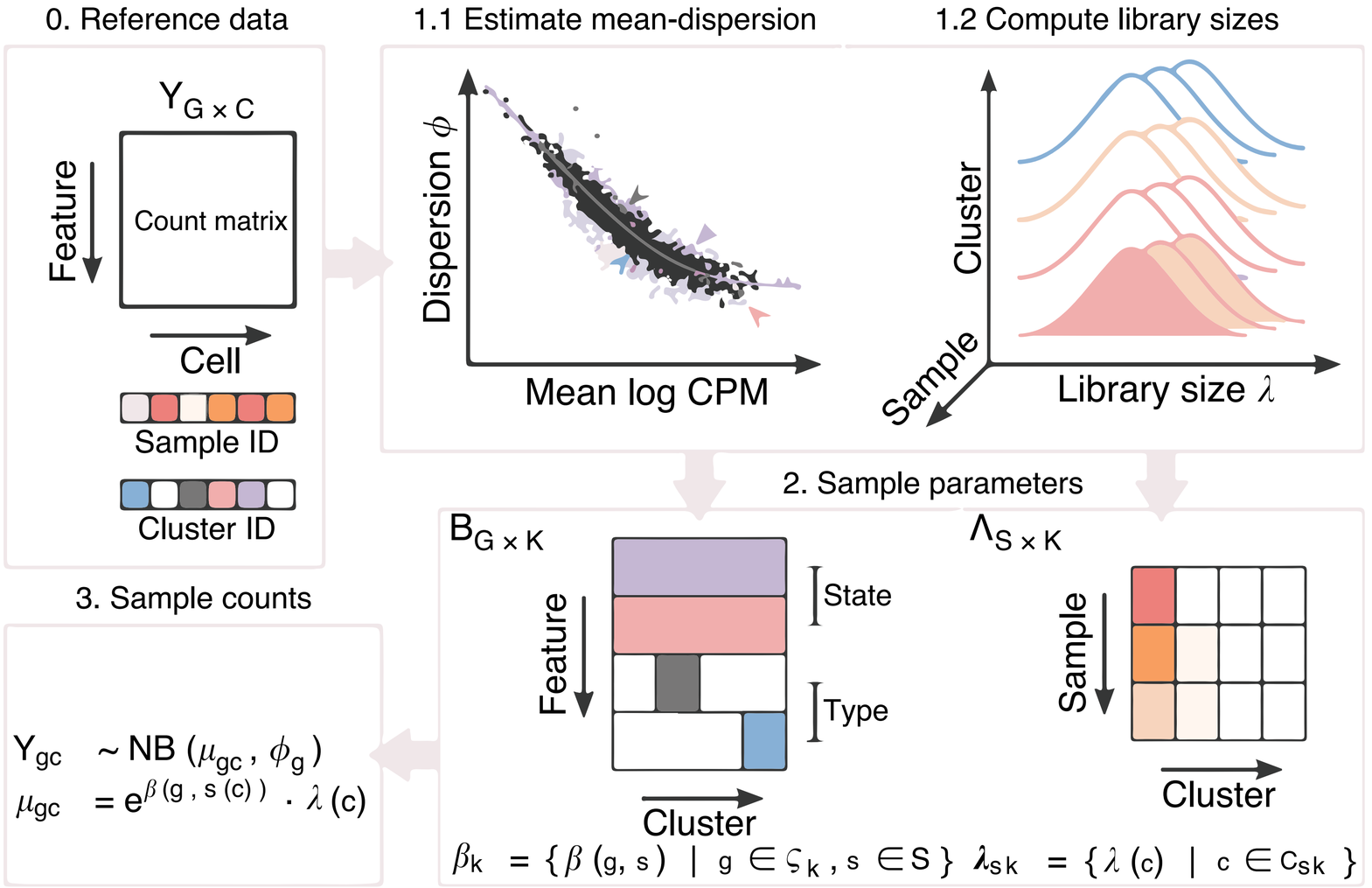
Karchenko, *Nature Methods* (2021)

Multi-individual single-cell data analysis



Park & Kellis Genome Biology (2021)

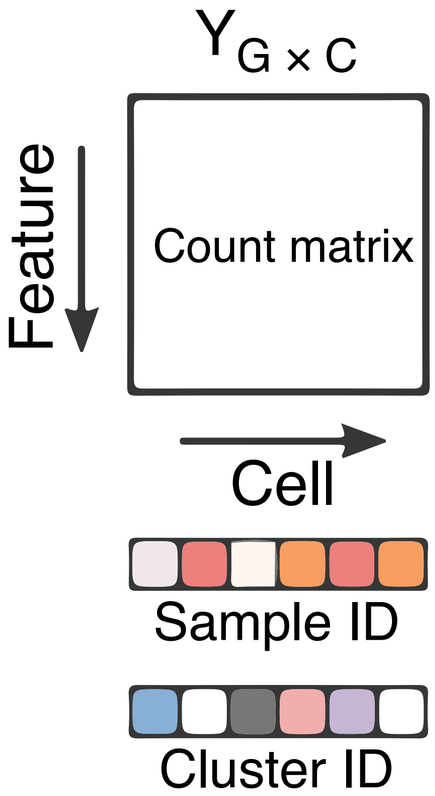
What is the best strategy to identify disease-specific genes from multi-individual studies?



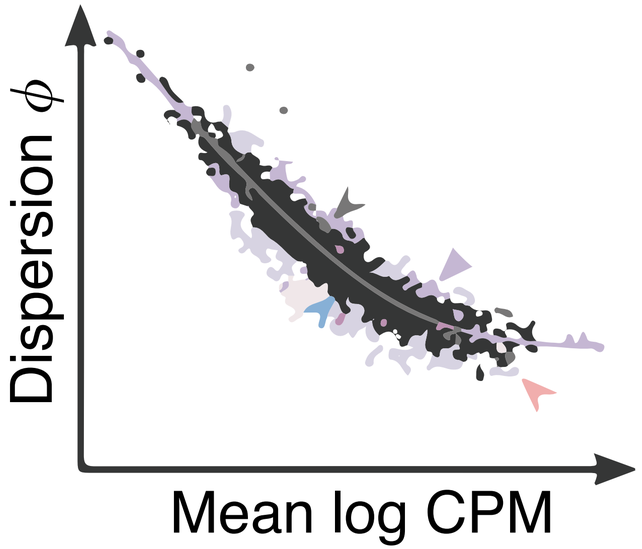
Crowell .. Robinson, Nature Comm. (2020)

Learn paramters from reference data

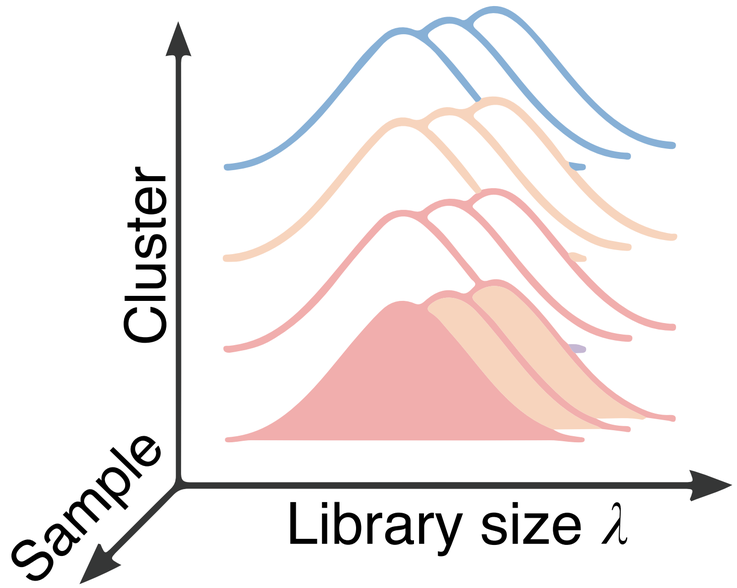
0. Reference data



1.1 Estimate mean-dispersion



1.2 Compute library sizes



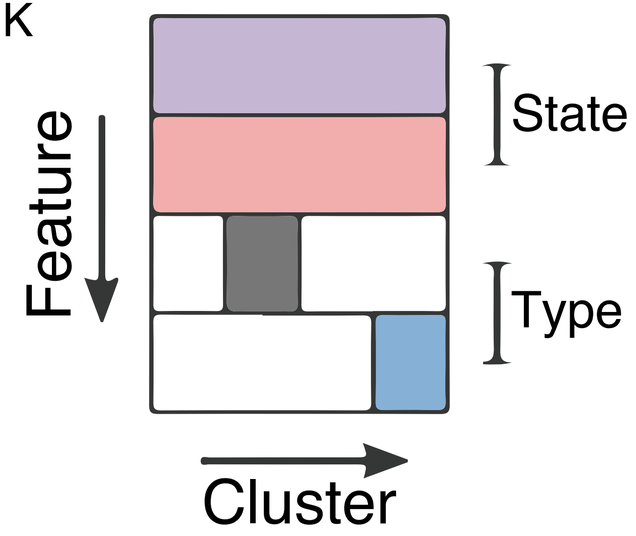
Crowell .. Robinson, Nature Comm. (2020)

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

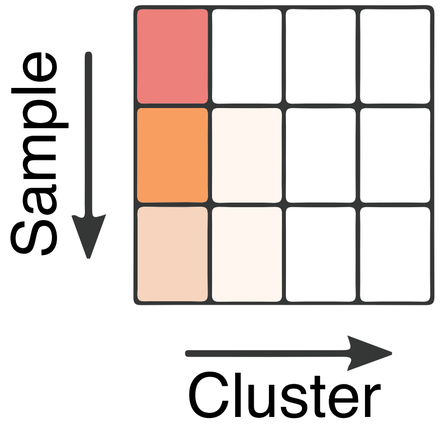
2. Sample parameters

3. Sample counts

$B_{G \times K}$



$\Lambda_{S \times K}$



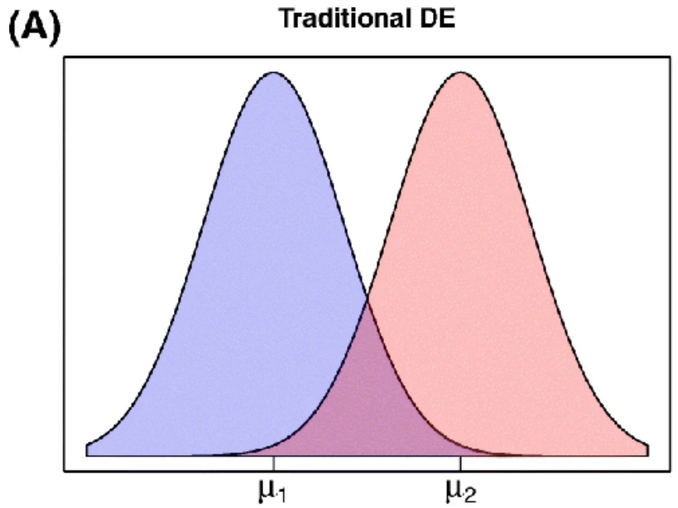
$$Y_{gc} \sim NB(\mu_{gc}, \phi_g)$$
$$\mu_{gc} = e^{\beta(g, s(c))} \cdot \lambda(c)$$

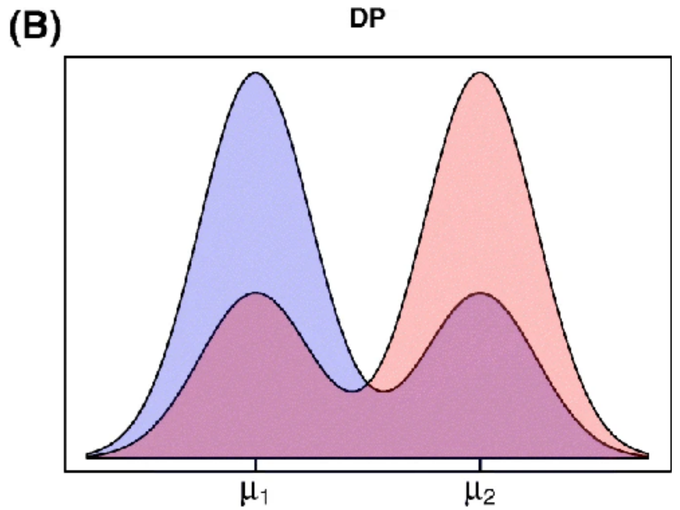
$$\beta_k = \{\beta(g, s) \mid g \in \varsigma_k, s \in S\} \quad \lambda_{sk} = \{\lambda(c) \mid c \in C_{sk}\}$$

Crowell .. Robinson, Nature Comm. (2020)

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

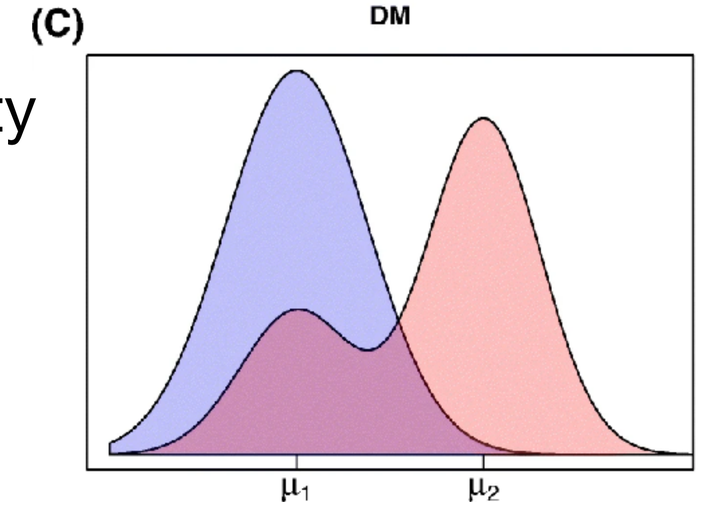
Diff. expr.

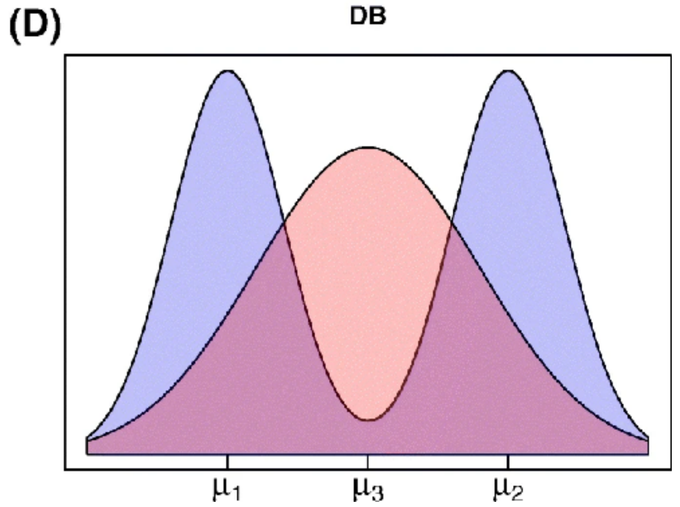




Diff. Proportion

Diff. modality

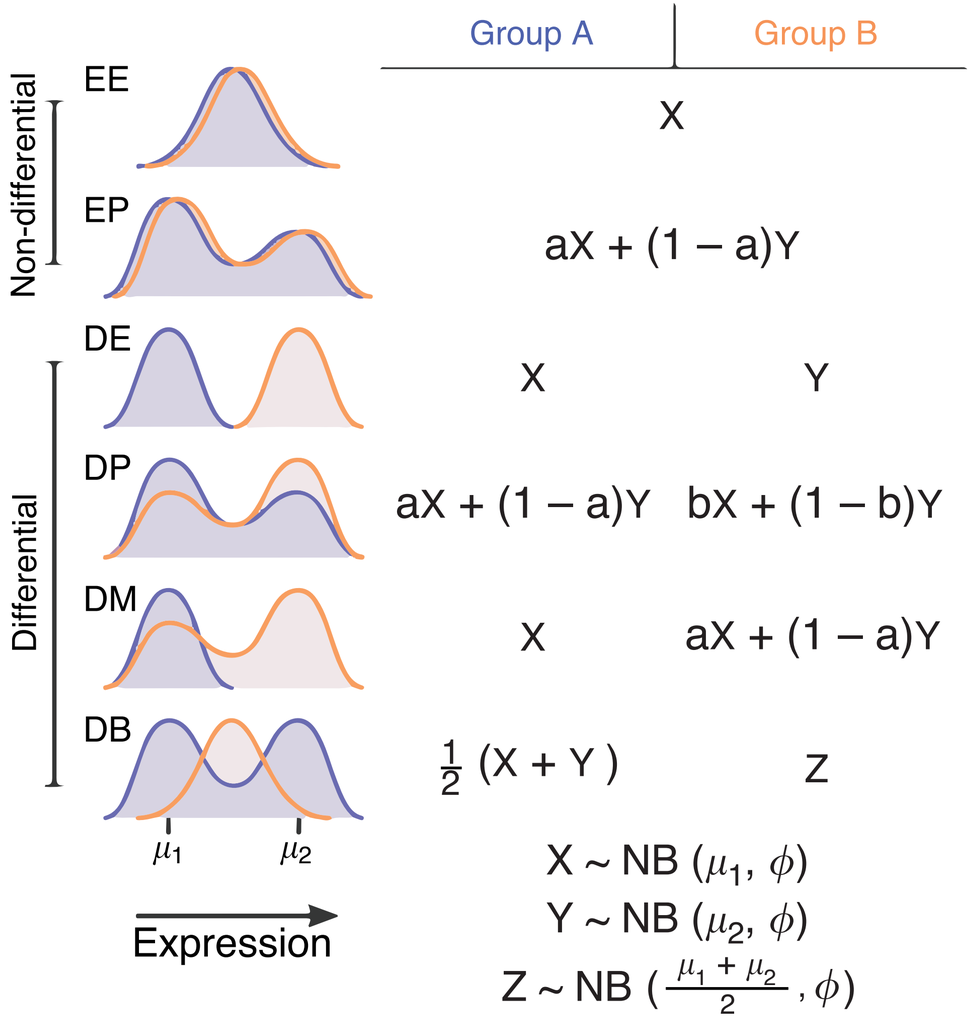




Diff. both

Korthauer *et al*. Genome Biology (2016)

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



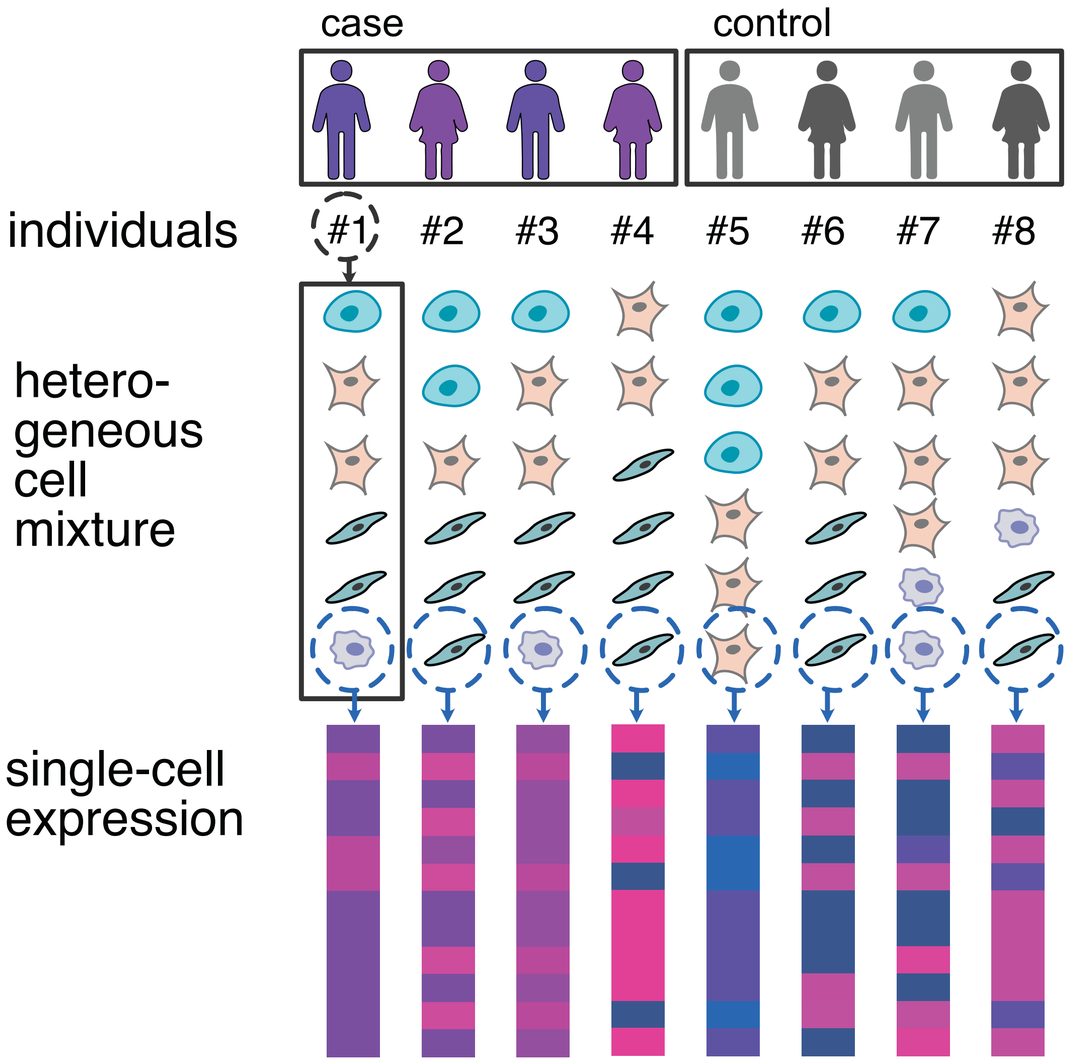
Diff. expr.

Diff. Proportion

Diff. modality

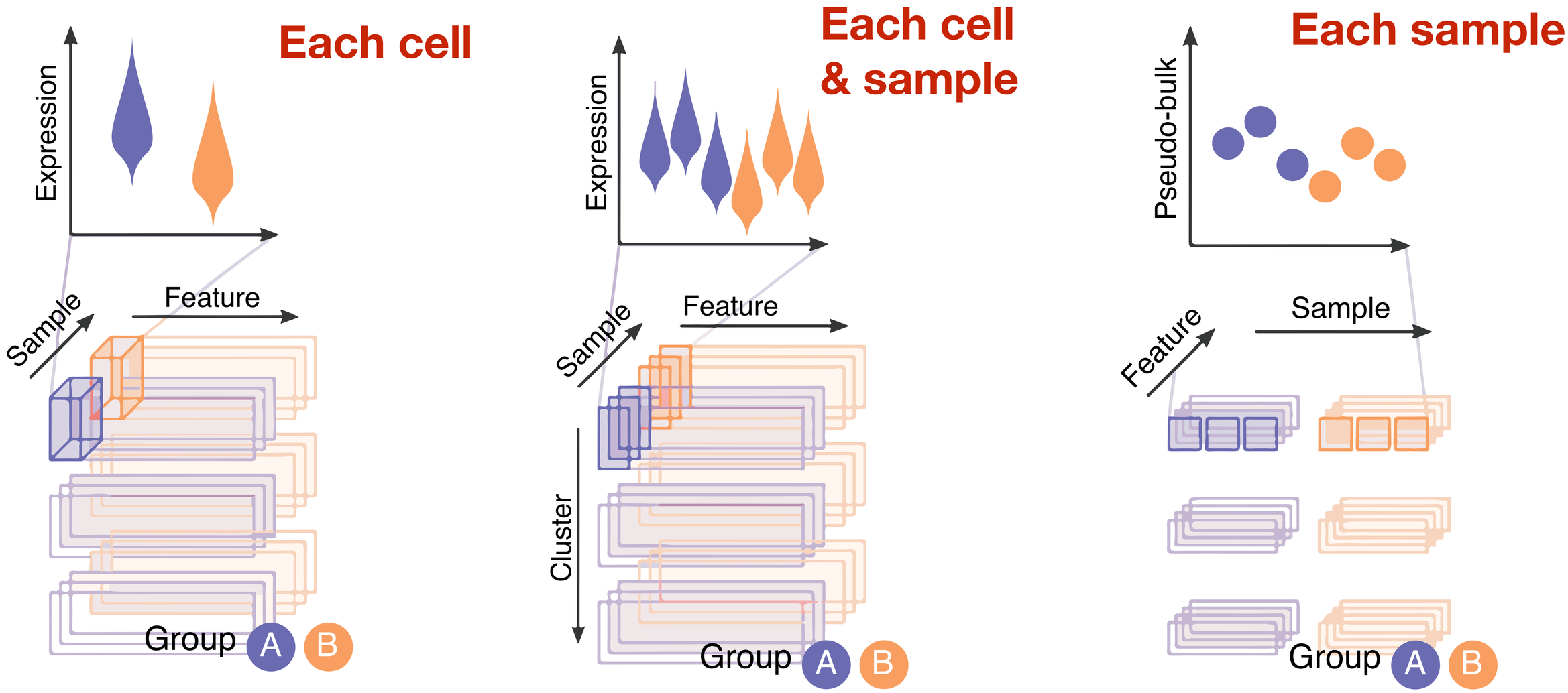
Crowell .. Robinson, Nature Comm. (2020)

Multi-individual single-cell data analysis



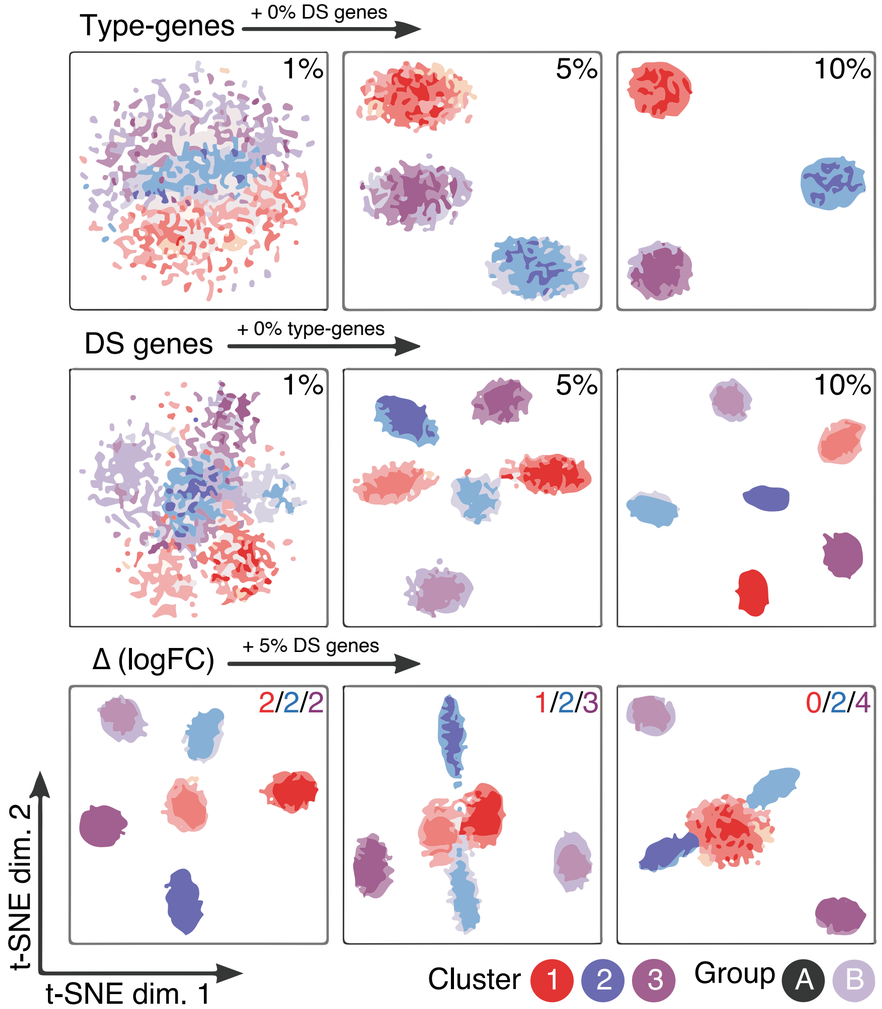
Park & Kellis Genome Biology (2021)

Multiple ways to define a single data point



Crowell .. Robinson, Nature Comm. (2020)

Genes may implicate cell type identity & disease



Cell type specific

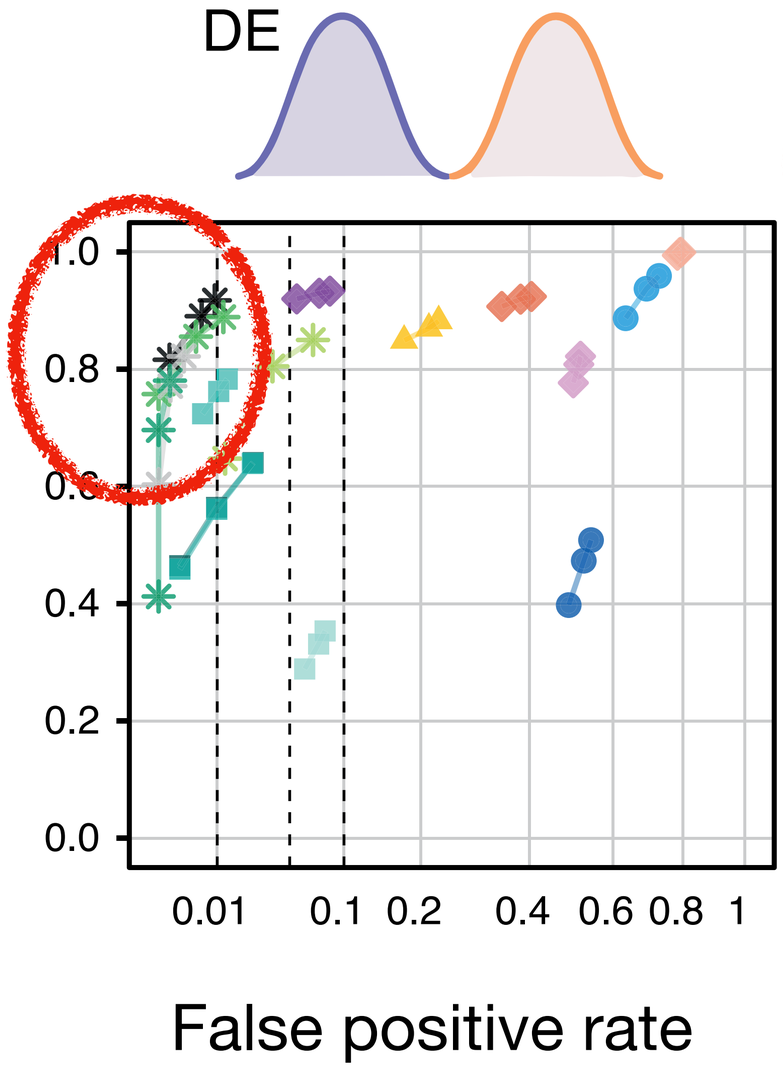
Disease specific

Both cell type & disease variation

Crowell .. Robinson, Nature Comm. (2020)

Benchmark: Pseudo-bulk approaches work best

Diff. expr.



edgeR.sum.counts
edgeR.sum.scalecpm

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

MAST.logcounts

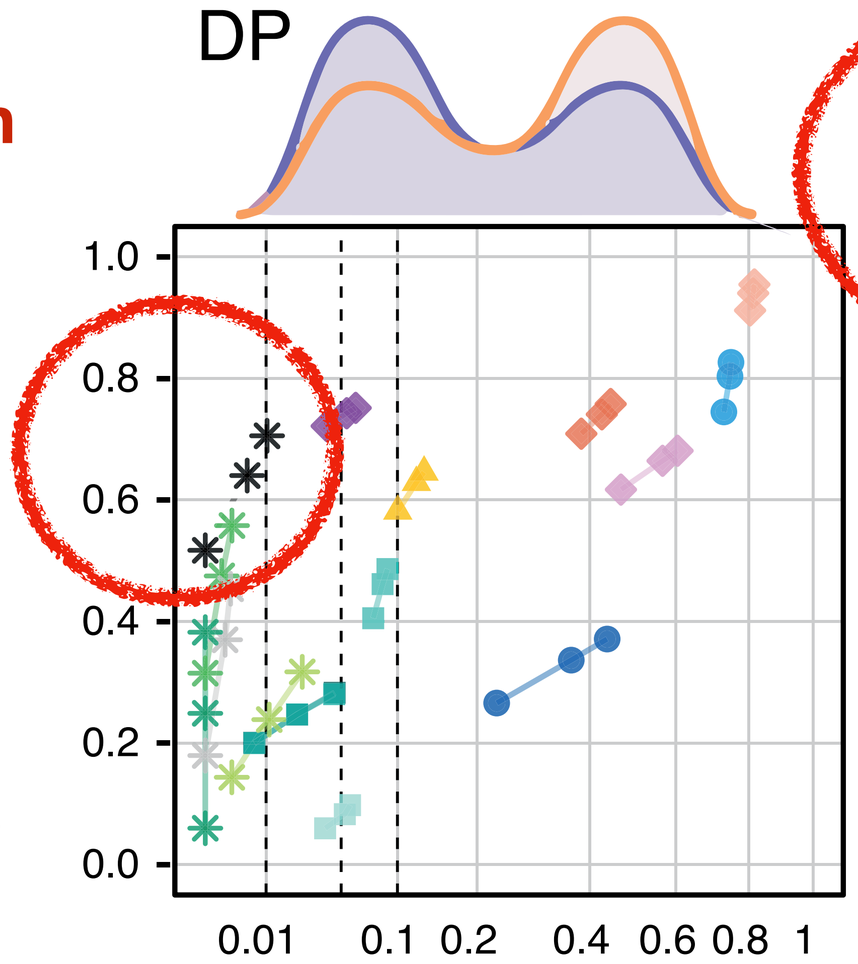
scDD.logcounts
scDD.vstresiduals

MM–dream
MM–dream2
MM–nbinom
MM–vst

mixed membership

Pseudo-bulk approaches: high power, low FDR

Diff. Proportion



True positive rate

False positive rate

edgeR.sum.counts
edgeR.sum.scalecpm

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

MAST.logcounts

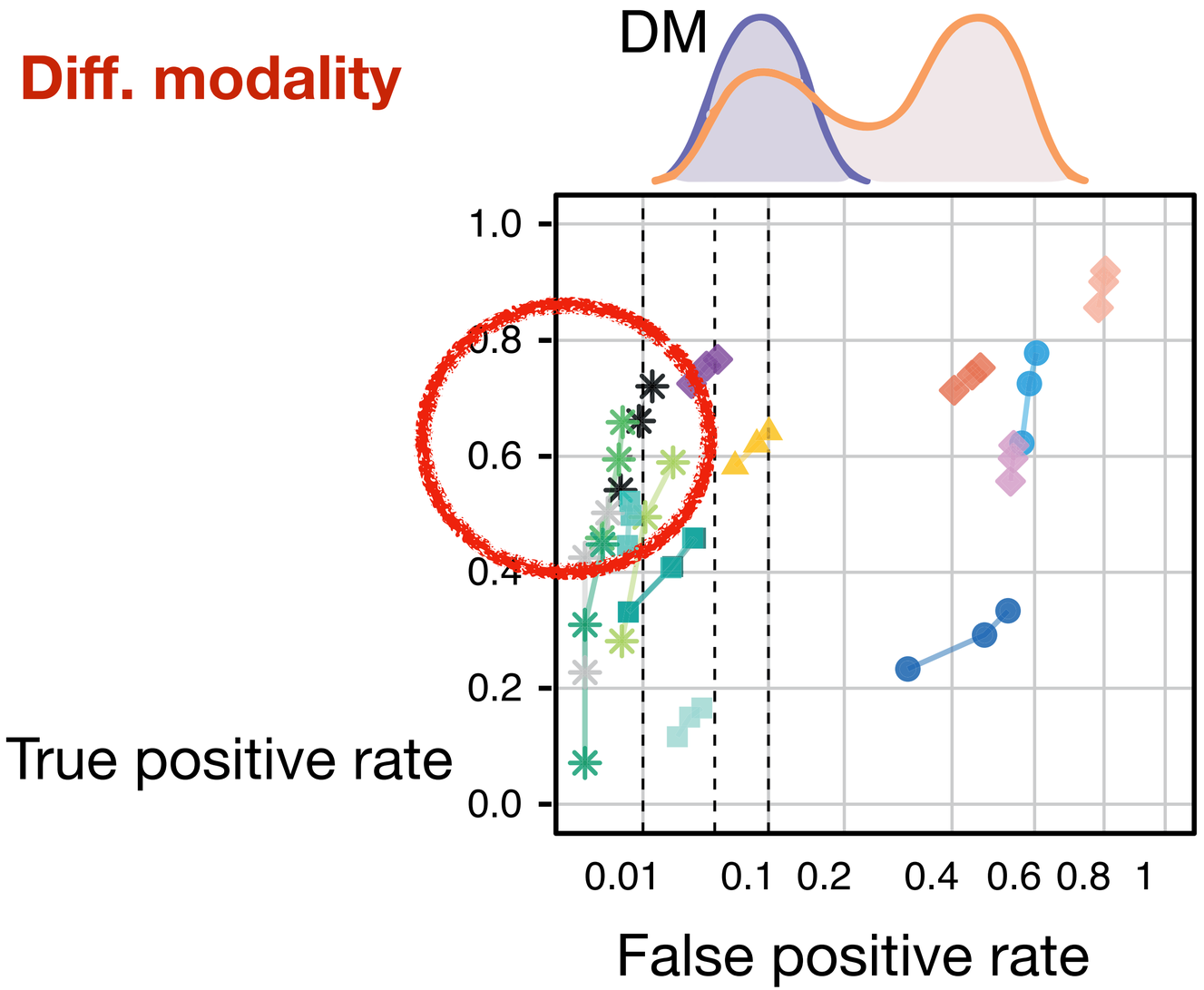
scDD.logcounts
scDD.vstresiduals

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

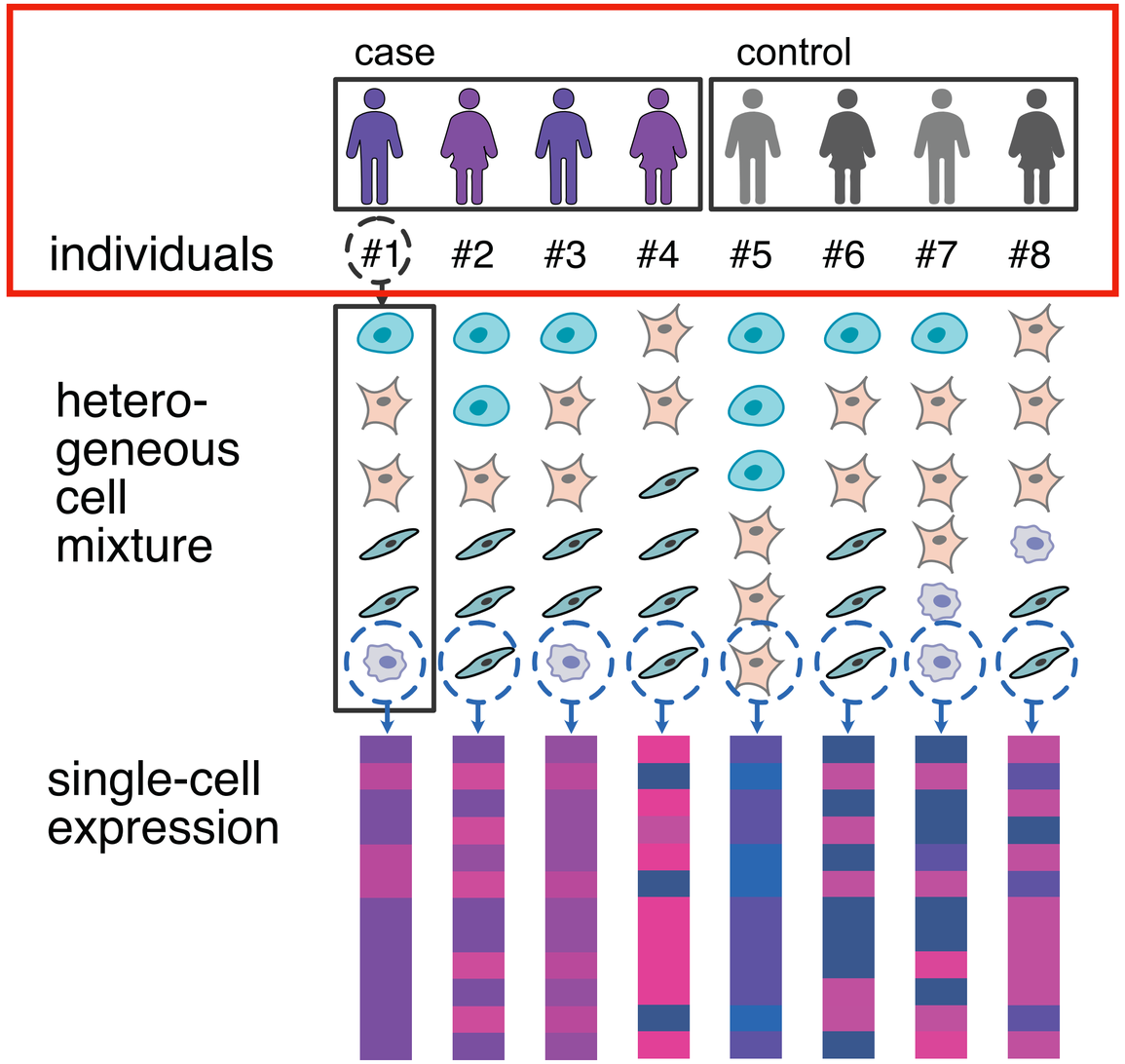
MAST.logcounts

scDD.logcounts
scDD.vstresiduals

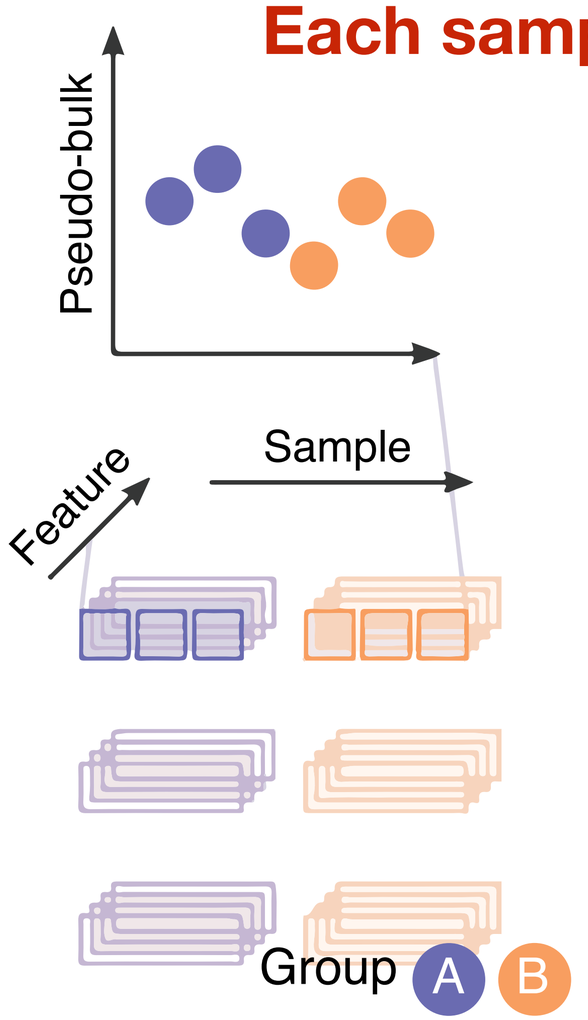
MM–dream
MM–dream2
MM–nbinom
MM–vst

mixed membership

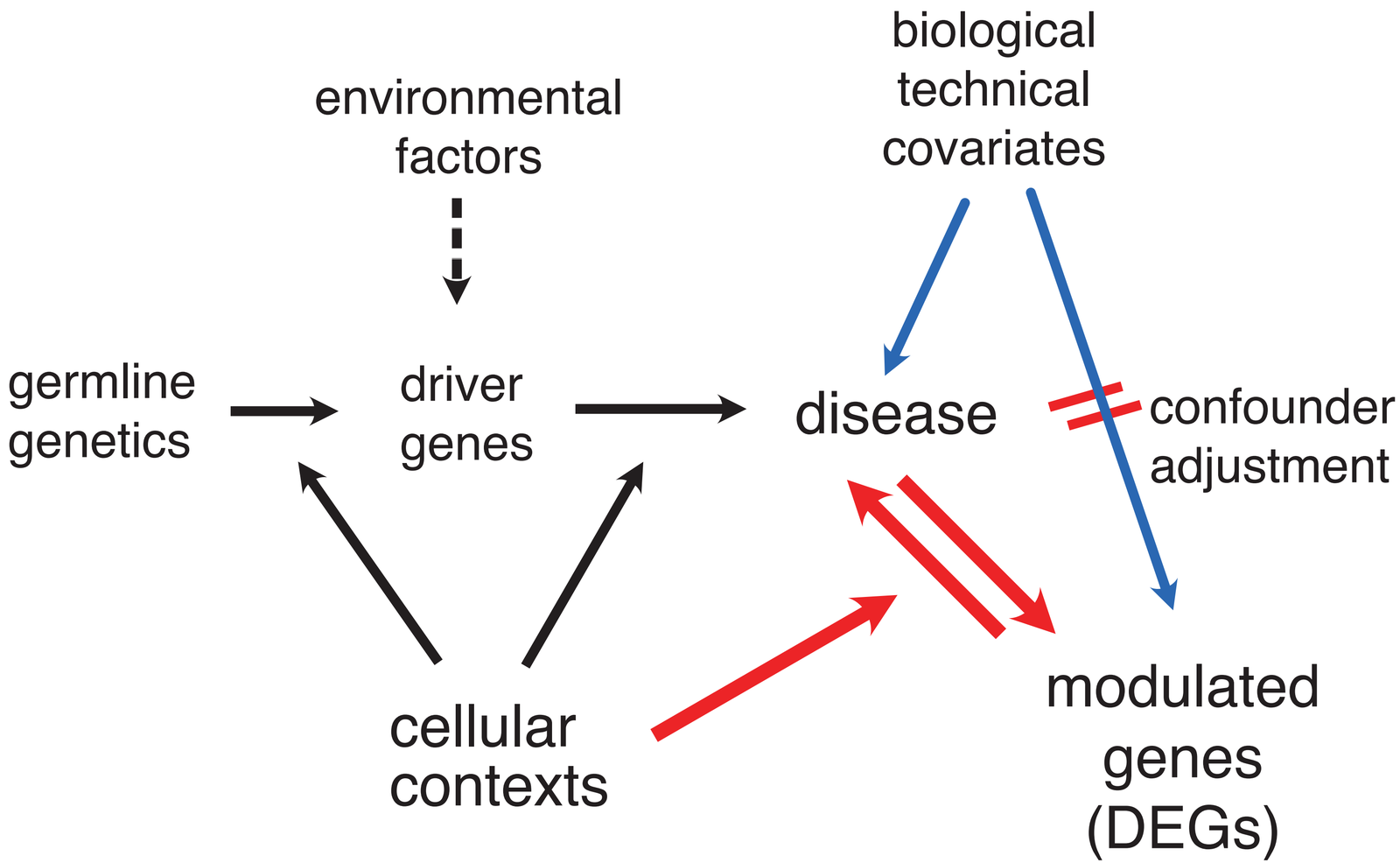
Each sample/individual within each cell type is a proper unit for DE analysis



How should we aggregate cells into pseudo-bulk samples?



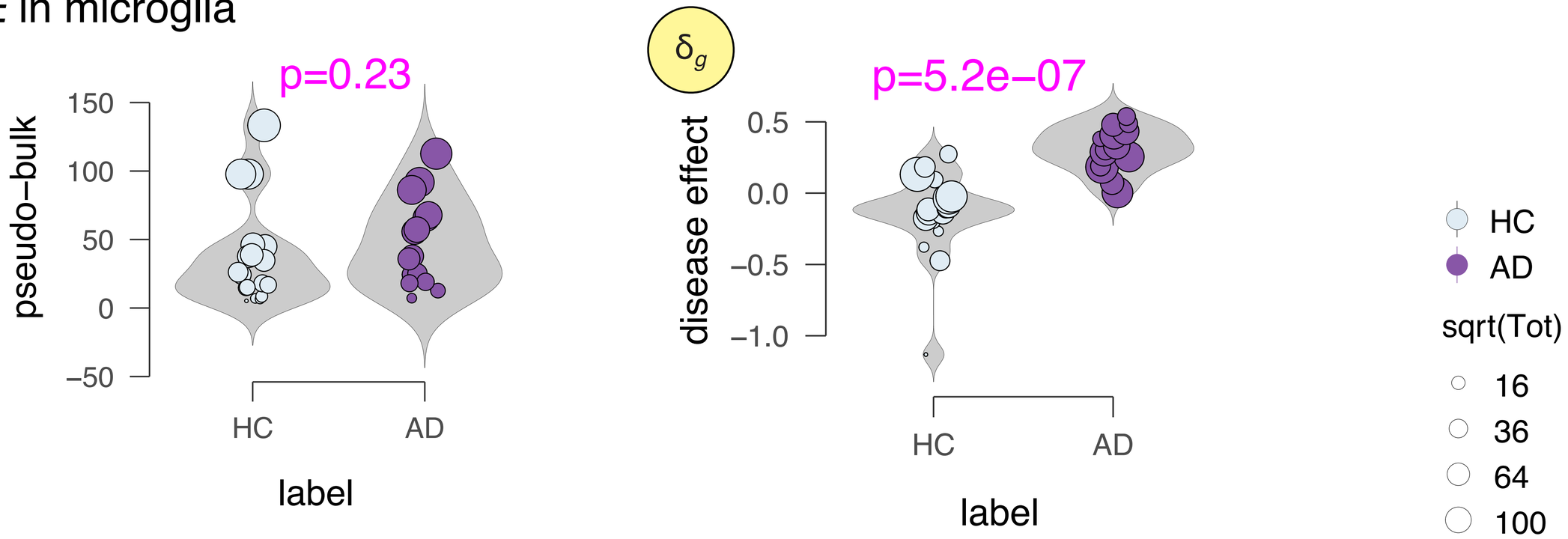
Differential expression analysis could be affected by confounding factors



Park & Kellis Genome Biology (2021)

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

APOE in microglia

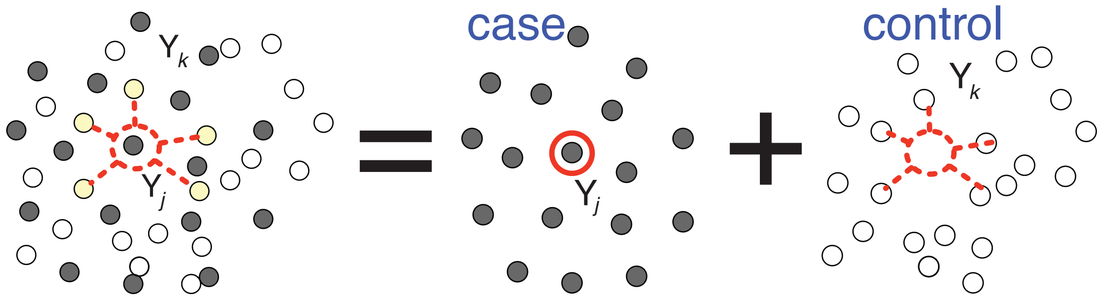


Park & Kellis Genome Biology (2021)

How can we identify confounding factors?

Step 1. Counterfactual imputation

Find nearest neighbour cells with a different label:

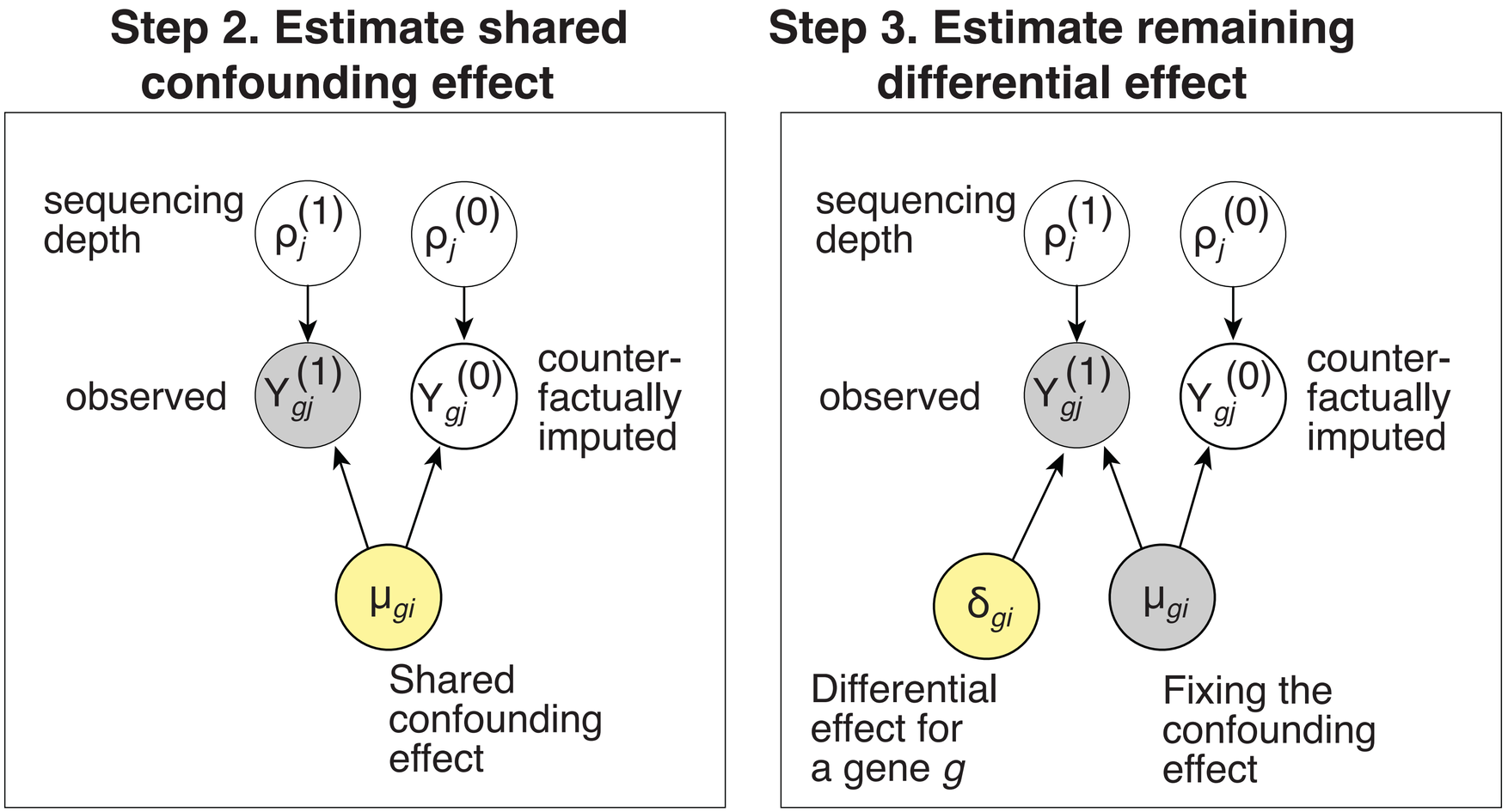


Regression on the cells with a different label: $Y_j^{(1)} \sim \Sigma_k Y_k^{(0)} \beta_k + \delta$

Counterfactual imputation: $Y_j^{(0)} \approx \Sigma_k Y_k^{(0)} \hat{\beta}_k$

Park & Kellis Genome Biology (2021)

How can we identify confounding factors?



Park & Kellis Genome Biology (2021)

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