# Module 1 Final Demo

- 10 min.
- Demo your code on a real dataset (optional task point if you apply it on a new dataset).
- Explain your design decisions.
- Show us the additional features you implemented and exploratory tasks that you did.

# Computational complexity of common operations

**Vector element wise product**

**Vector dot product (vector -> scalar)**

**Matrix multiplication (two n x n matrices):**

In practice, O(n^3) and driven by implementation efficiency

| year | algorithm | order of growth |
|---|---|---|
| ? | brute force | $N^3$ |
| 1969 | Strassen | $N^{2.808}$ |
| 1978 | Pan | $N^{2.796}$ |
| 1979 | Bini | $N^{2.780}$ |
| 1981 | Schönhage | $N^{2.522}$ |
| 1982 | Romani | $N^{2.517}$ |
| 1982 | Coppersmith-Winograd | $N^{2.496}$ |
| 1986 | Strassen | $N^{2.479}$ |
| 1989 | Coppersmith-Winograd | $N^{2.376}$ |
| 2010 | Strother | $N^{2.3737}$ |
| 2011 | Williams | $N^{2.3727}$ |
| ? | ? | $N^{2+\varepsilon}$ |

**number of floating-point operations to multiply two N-by-N matrices**

# Computational complexity of common operations

**Vector element wise product**

**Vector dot product (vector -> scalar)**

**Matrix multiplication (two n x n matrices):**

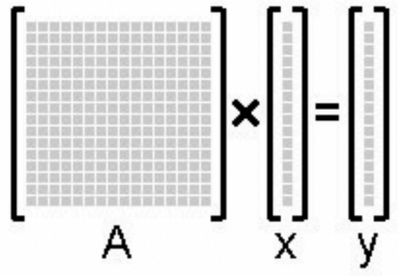
**Matrix-vector multiplication**

| year | algorithm | order of growth |
|---|---|---|
| ? | brute force | $N^3$ |
| 1969 | Strassen | $N^{2.808}$ |
| 1978 | Pan | $N^{2.796}$ |
| 1979 | Bini | $N^{2.780}$ |
| 1981 | Schönhage | $N^{2.522}$ |
| 1982 | Romani | $N^{2.517}$ |
| 1982 | Coppersmith-Winograd | $N^{2.496}$ |
| 1986 | Strassen | $N^{2.479}$ |
| 1989 | Coppersmith-Winograd | $N^{2.376}$ |
| 2010 | Strother | $N^{2.3737}$ |
| 2011 | Williams | $N^{2.3727}$ |
| ? | ? | $N^{2+\varepsilon}$ |

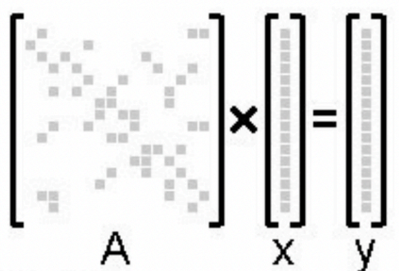
**number of floating-point operations to multiply two N-by-N matrices**

# Structure in the data can be exploited for faster computation

**Dense Matrix-vector Multiplication : O(n^2)**




**Sparse Matrix-vector Multiplication: O(m)**

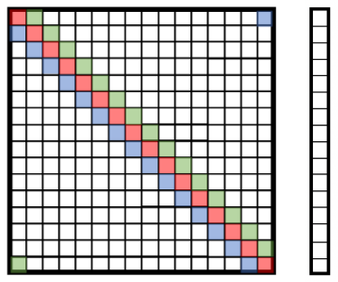



(Structured sparse patterns can be exploited, e.g. block diagonal, banded)

**Circulant Matrix-vector Multiplication: O(n log n)**

$$\begin{bmatrix} a_0 & a_{-1} & a_{-2} & a_{-3} \\ a_1 & a_0 & a_{-1} & a_{-2} \\ a_2 & a_1 & a_0 & a_{-1} \\ a_3 & a_2 & a_1 & a_0 \end{bmatrix} \quad \begin{bmatrix} 7 & 11 & 5 & 6 \\ 3 & 7 & 11 & 5 \\ 8 & 3 & 7 & 11 \\ 1 & 8 & 3 & 7 \end{bmatrix}$$

Convolution can be viewed as circulant matrix vector multiplication



# Computational complexity of common operations

**Vector element wise product**

**Vector dot product (vector -> scalar)**

**Matrix multiplication (two n x n matrices):**

**Matrix-vector multiplication**

**Matrix inversion**

**Linear solve Ax = b**

**Sparse linear solve Ax = b, where A is sparse**

**SVD**

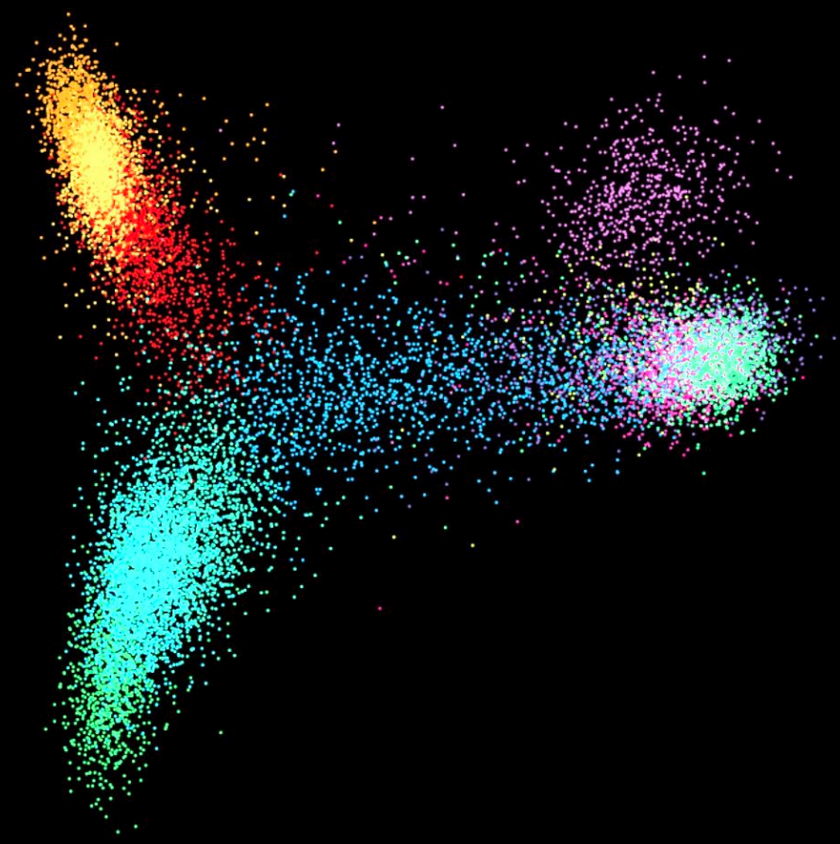
**Nearest neighbor search**

Practical note:

1. Utilize structure in the data  (e.g. sparsity, Toeplitz) whenever possible
2. Avoid O(n^3) operations whenever possible (e.g. with randomized SVD)

## Linear representations
## (PCA, ICA, …)

Easy comparison & integration across datasets



## Nonlinear representations
## (t-SNE, UMAP…)

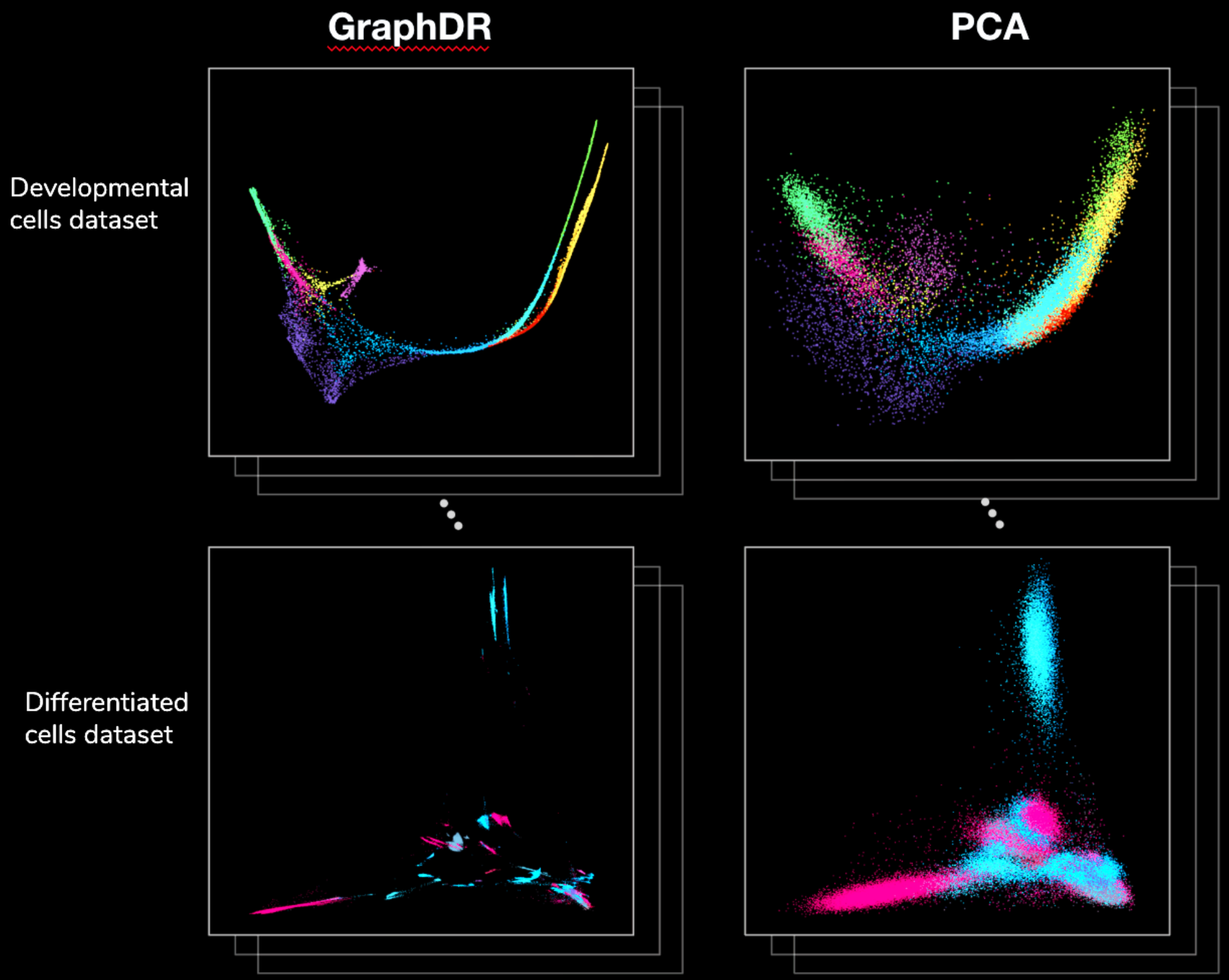
Good representation of cell types / states

**Can we combine advantages of linear and nonlinear representations?**

**Our solution: f**rom linear to **quasilinear** representation

“Quasilinear” approaches:

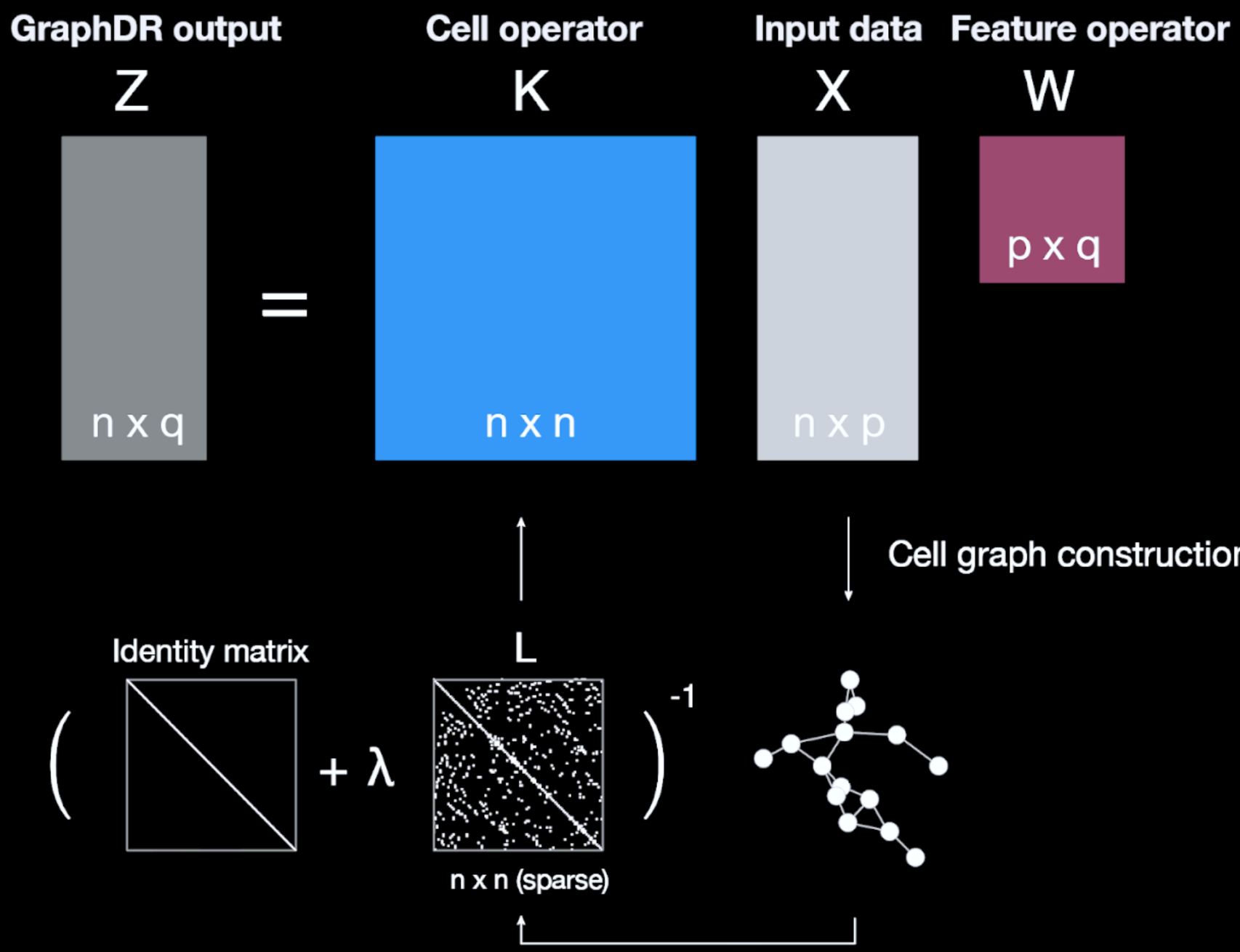
**GraphDR - quasilinear visualization and general representation**




Data from Hochgerner et al. 2018, Zeisel et al,. 2018

“Quasilinear” approaches:

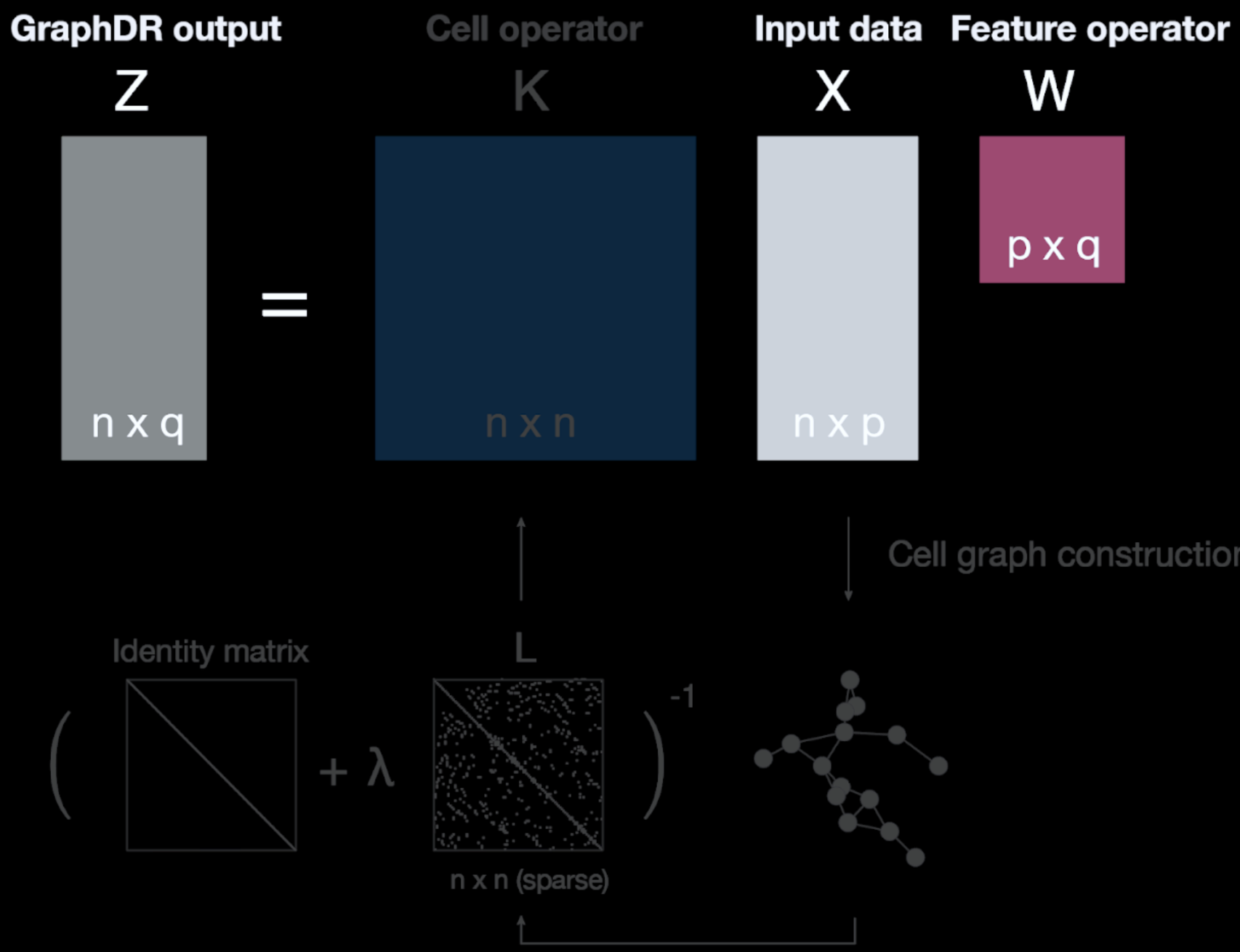
**GraphDR - quasilinear visualization and general representation**




**GraphDR objective:** $\underset{W,Z}{minimize} \quad \|X - Z\mathrm{W}^{\mathrm{T}}\|_2^2 + \lambda \sum_{\{i,j\}\in G} G_{ij}\|Z_i - Z_j\|_2^2, \qquad s.t.\ W^TW = I$

“Quasilinear” approaches:

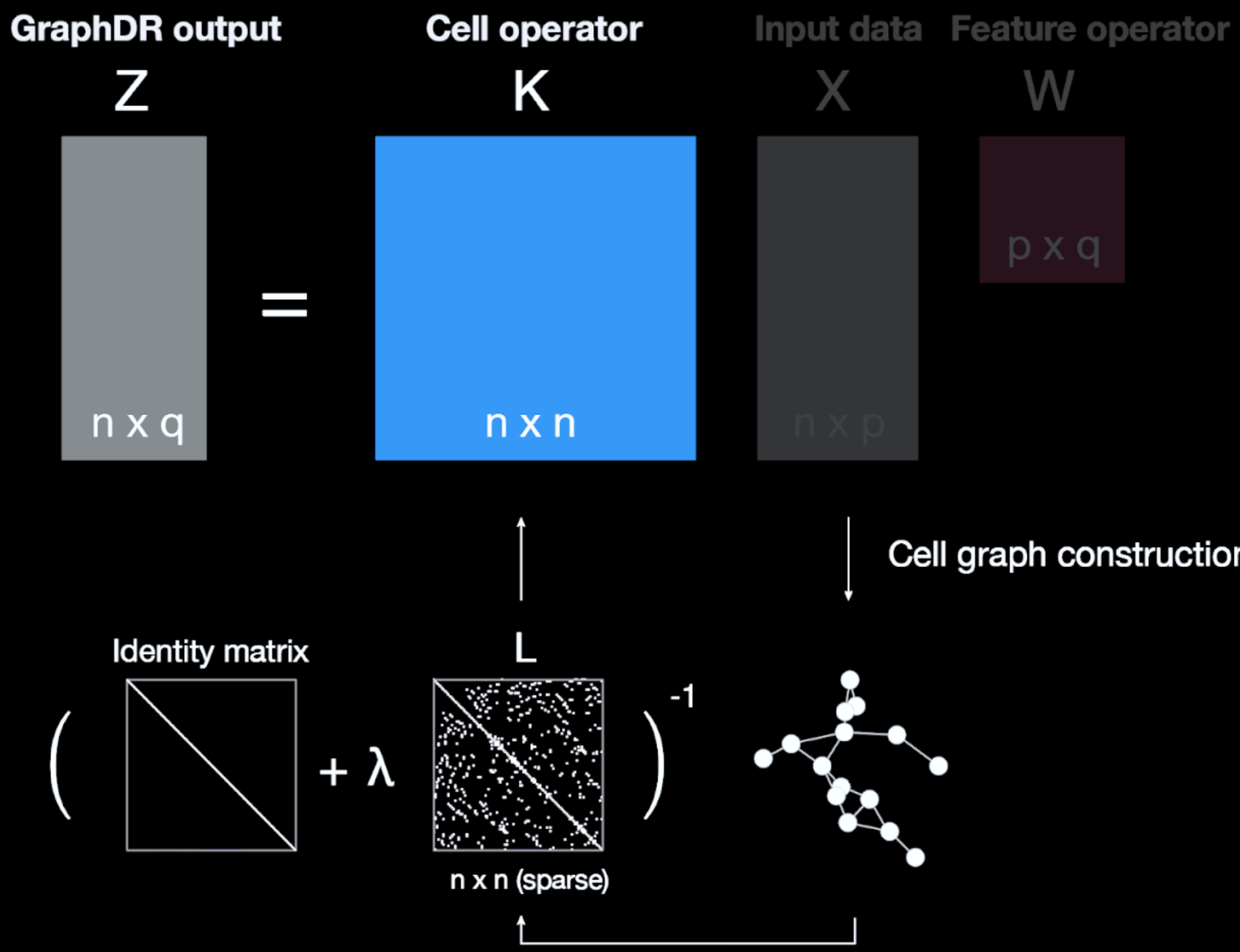
**GraphDR - quasilinear visualization and general representation**




GraphDR objective function: $\underset{W,Z}{minimize} \quad \|XW - Z\|_2^2 + \lambda \sum_{\{i,j\} \in G} G_{ij} \|Z_i - Z_j\|_2^2, \qquad s.t.\ W^T W = I$

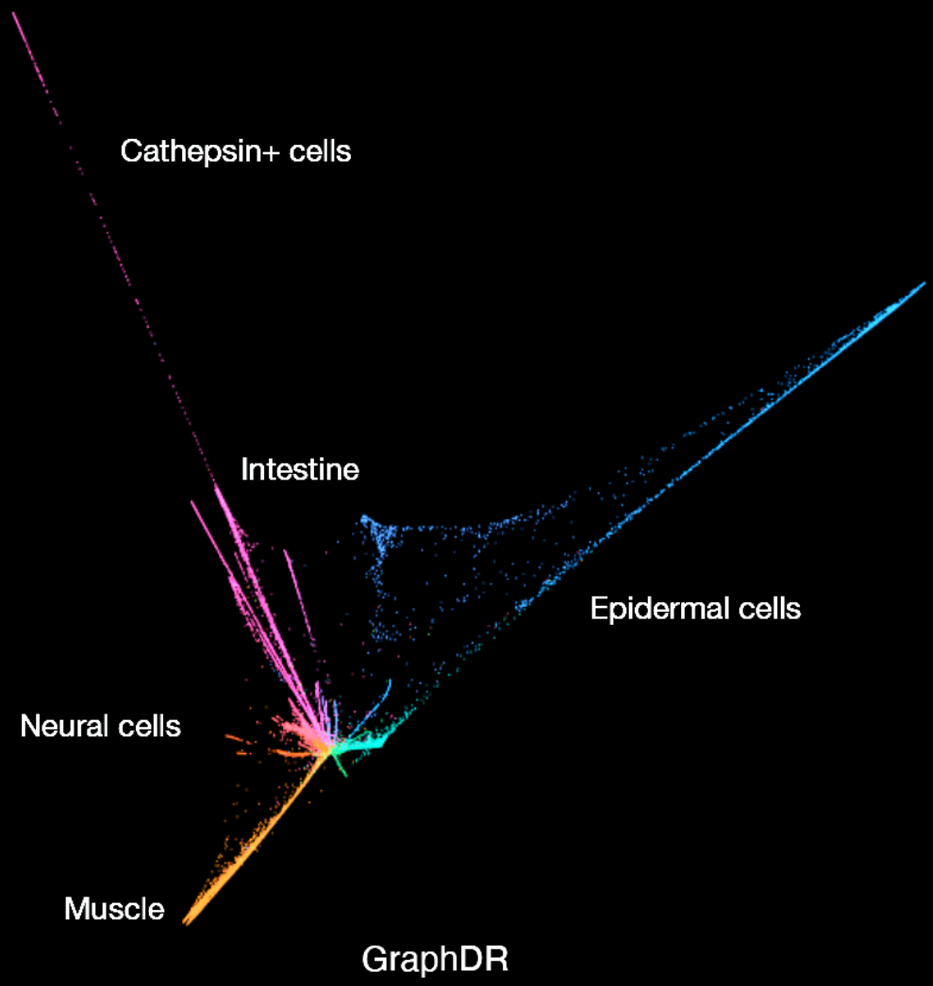
“Quasilinear” approaches:

**GraphDR - quasilinear visualization and general representation**

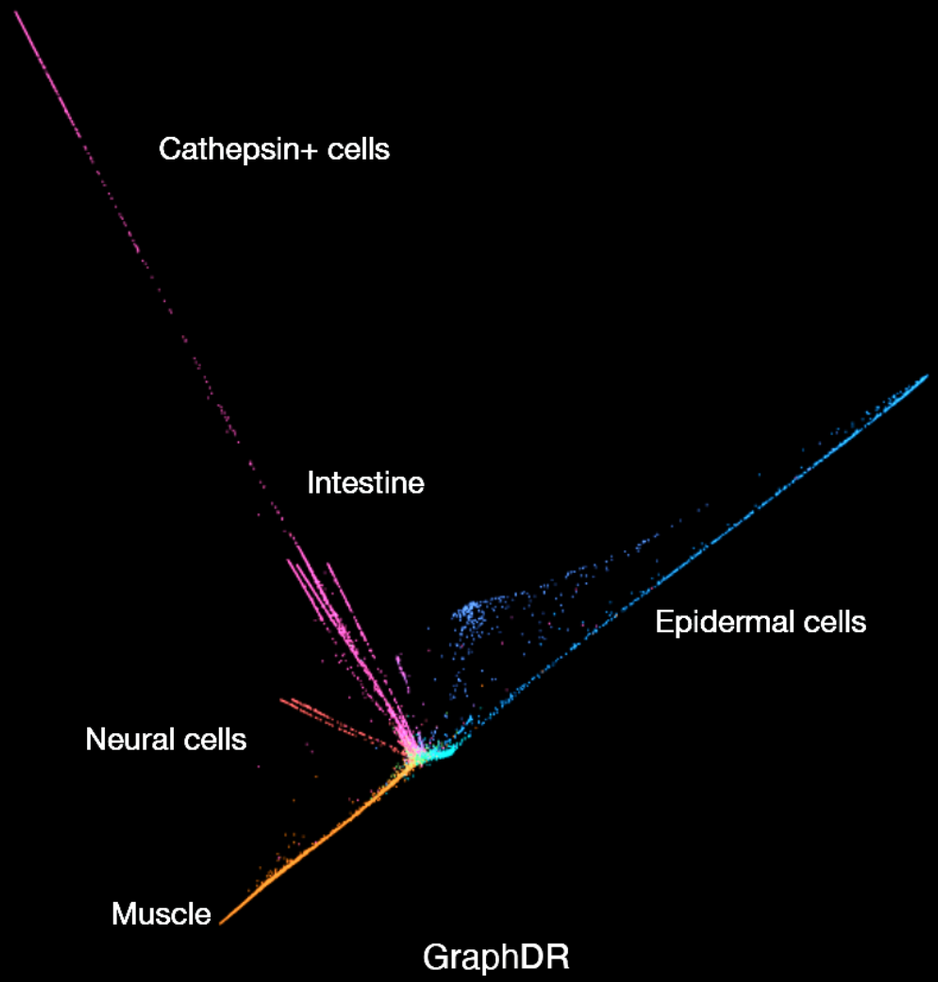



**GraphDR objective:** $\underset{W,Z}{minimize} \quad \|X - Z\mathrm{W}^{\mathrm{T}}\|_2^2 + \lambda \sum_{\{i,j\}\in G} G_{ij}\|Z_i - Z_j\|_2^2, \qquad s.t.\ W^TW = I$

# GraphDR representations allow direct comparison across datasets

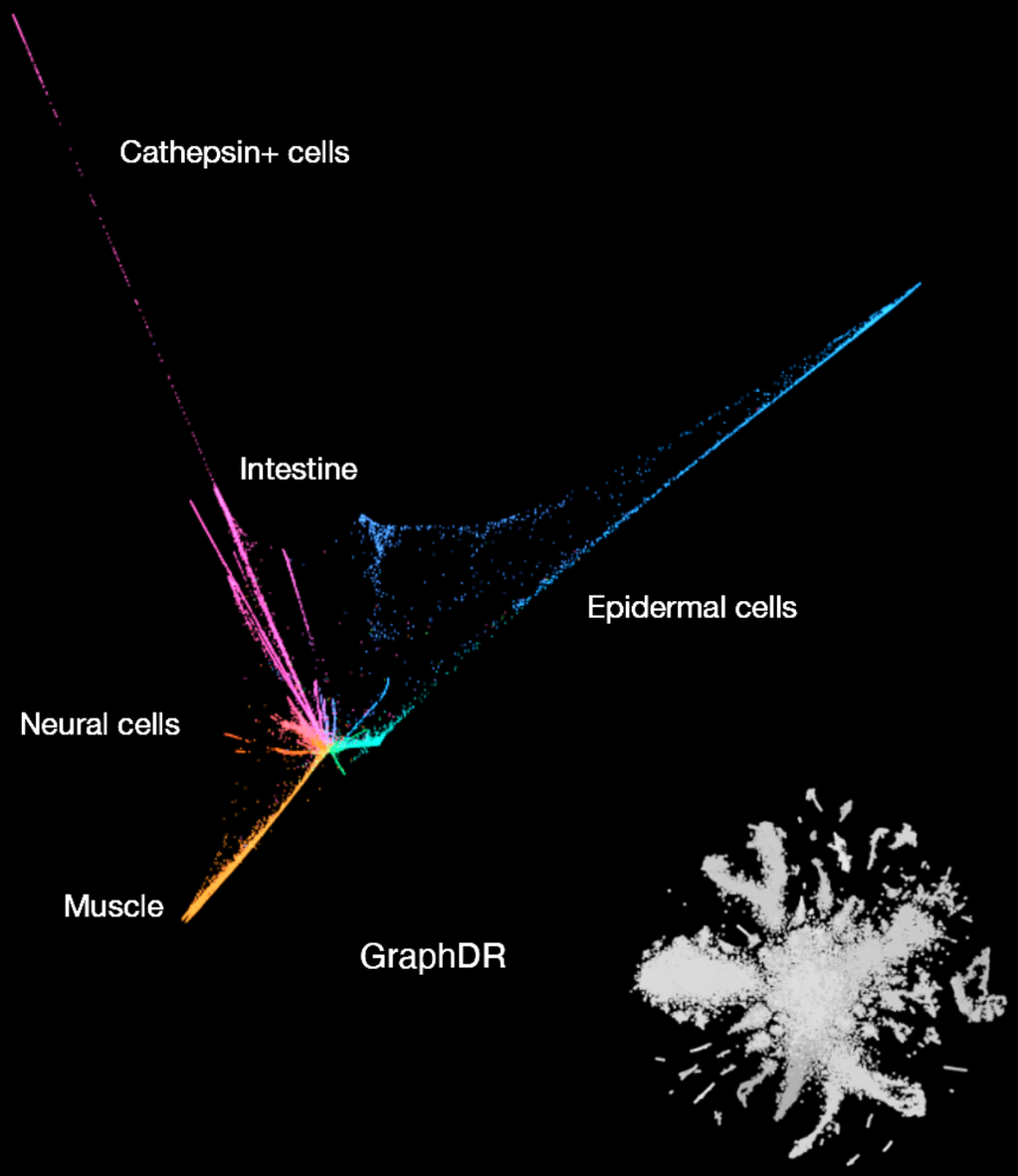



**Planarian whole animal (Fincher et. al.)**

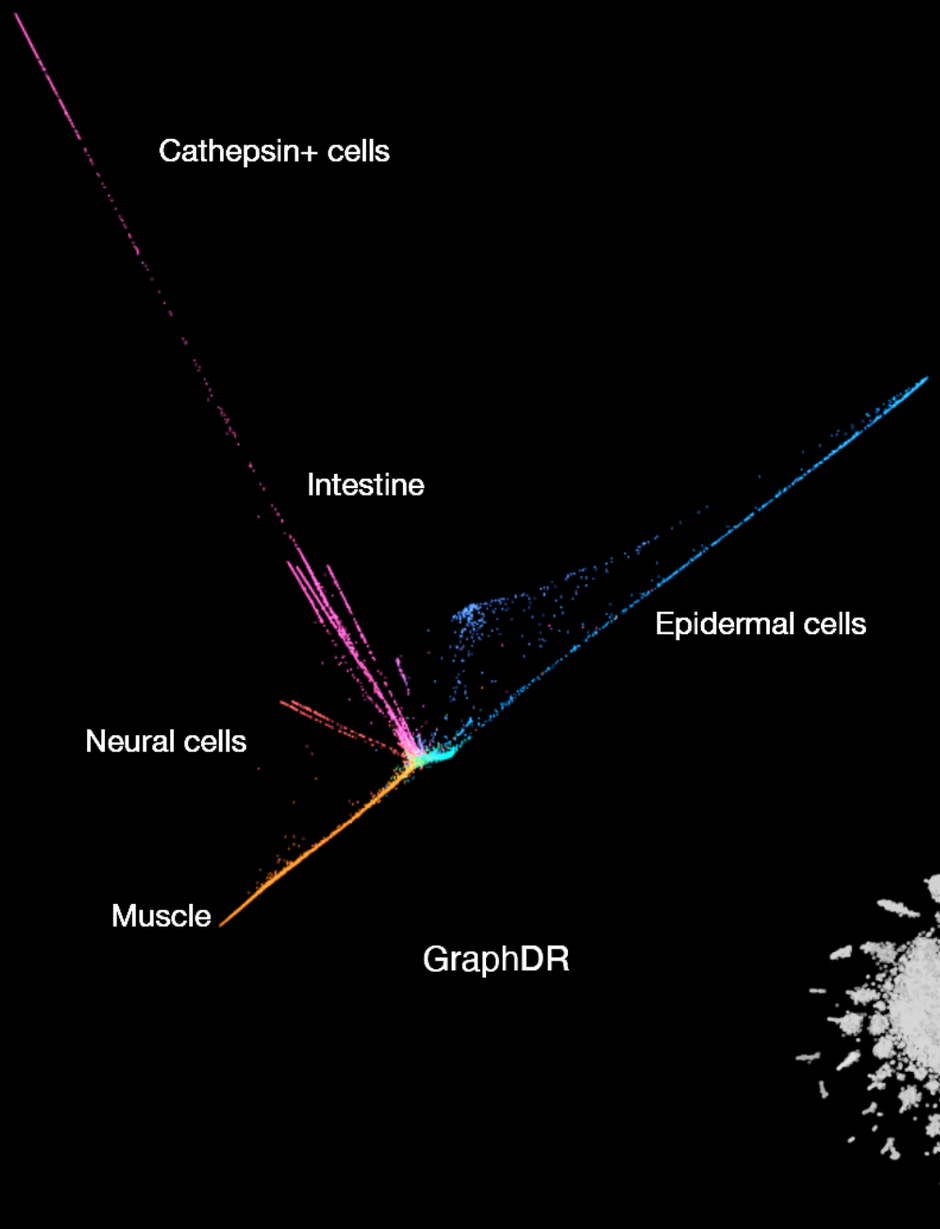



**Planarian whole animal (Plass et. al.)**

# GraphDR representations allow direct comparison across datasets

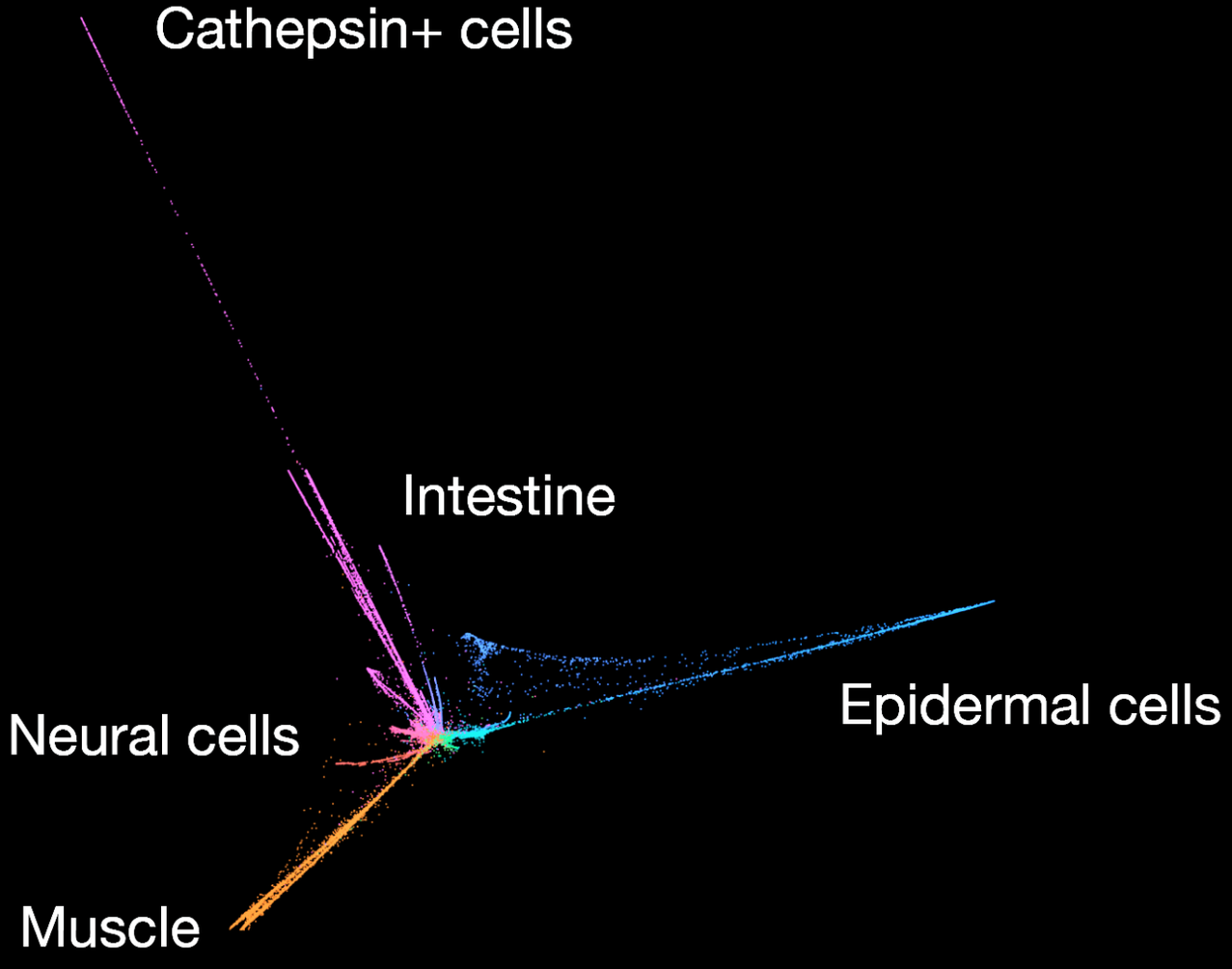



**Planarian whole animal (Fincher et. al.)**

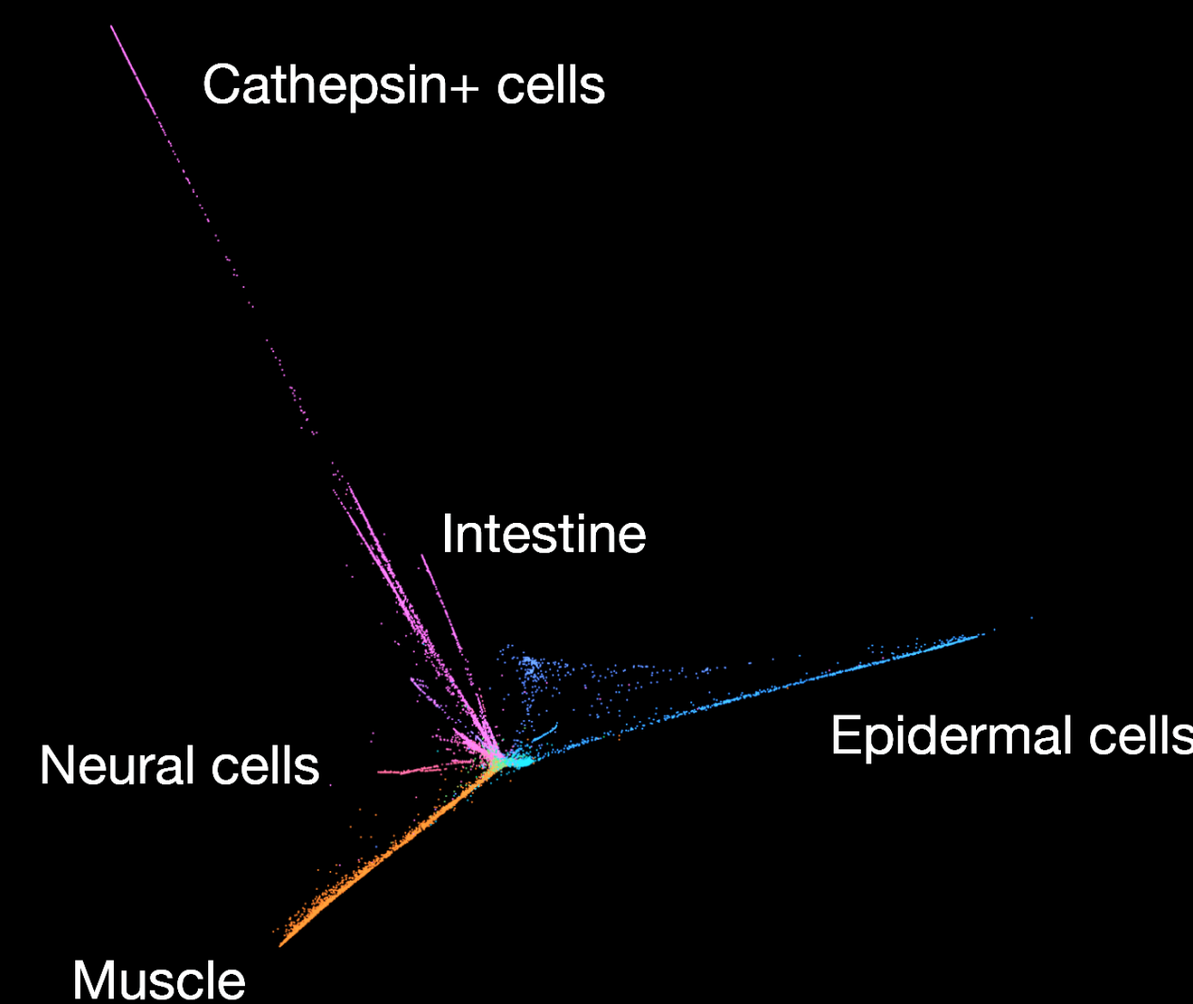



**Planarian whole animal (Plass et. al.)**

# Improved comparison cross datasets with graph-based alignment

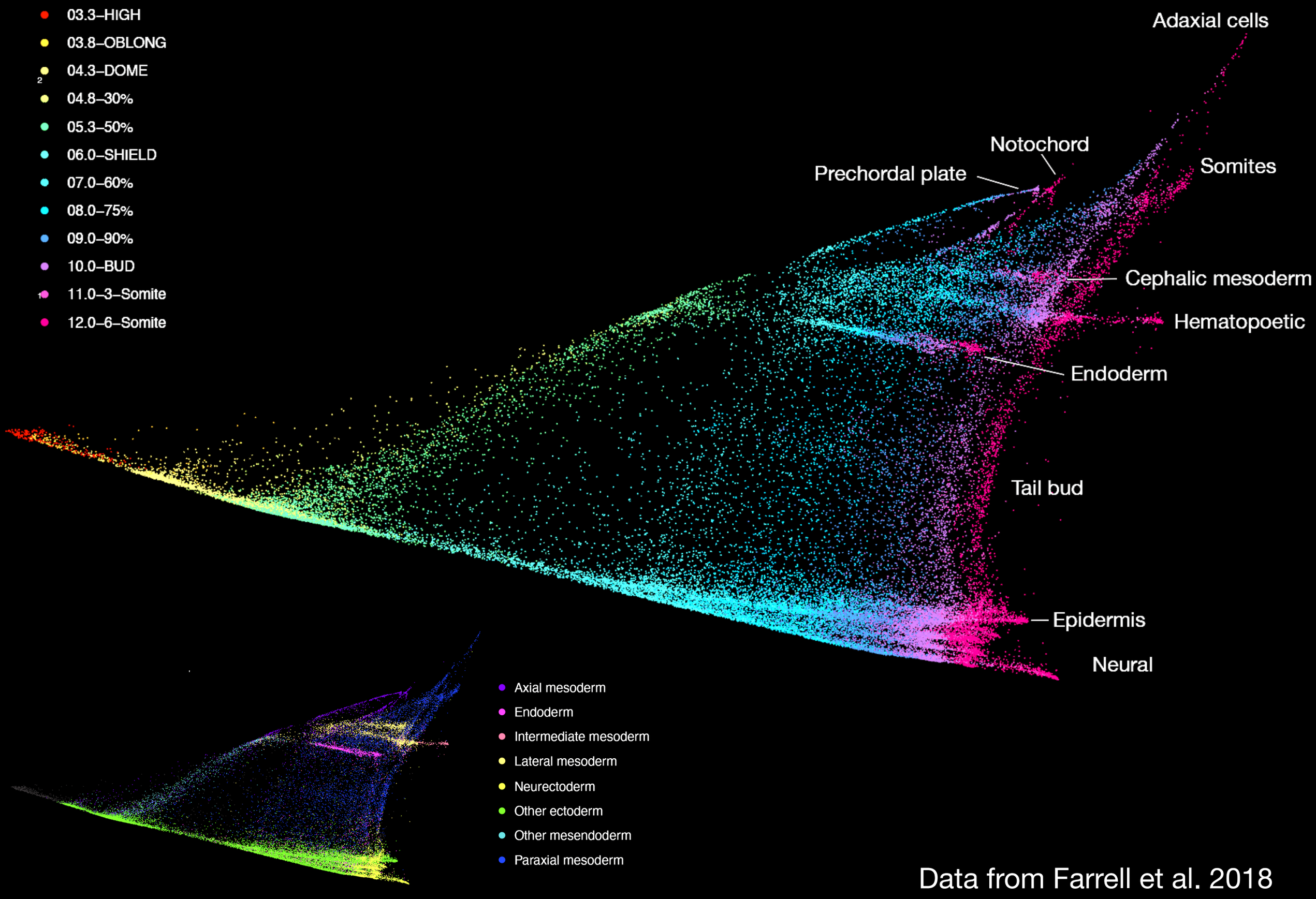



**Fincher et. al.**



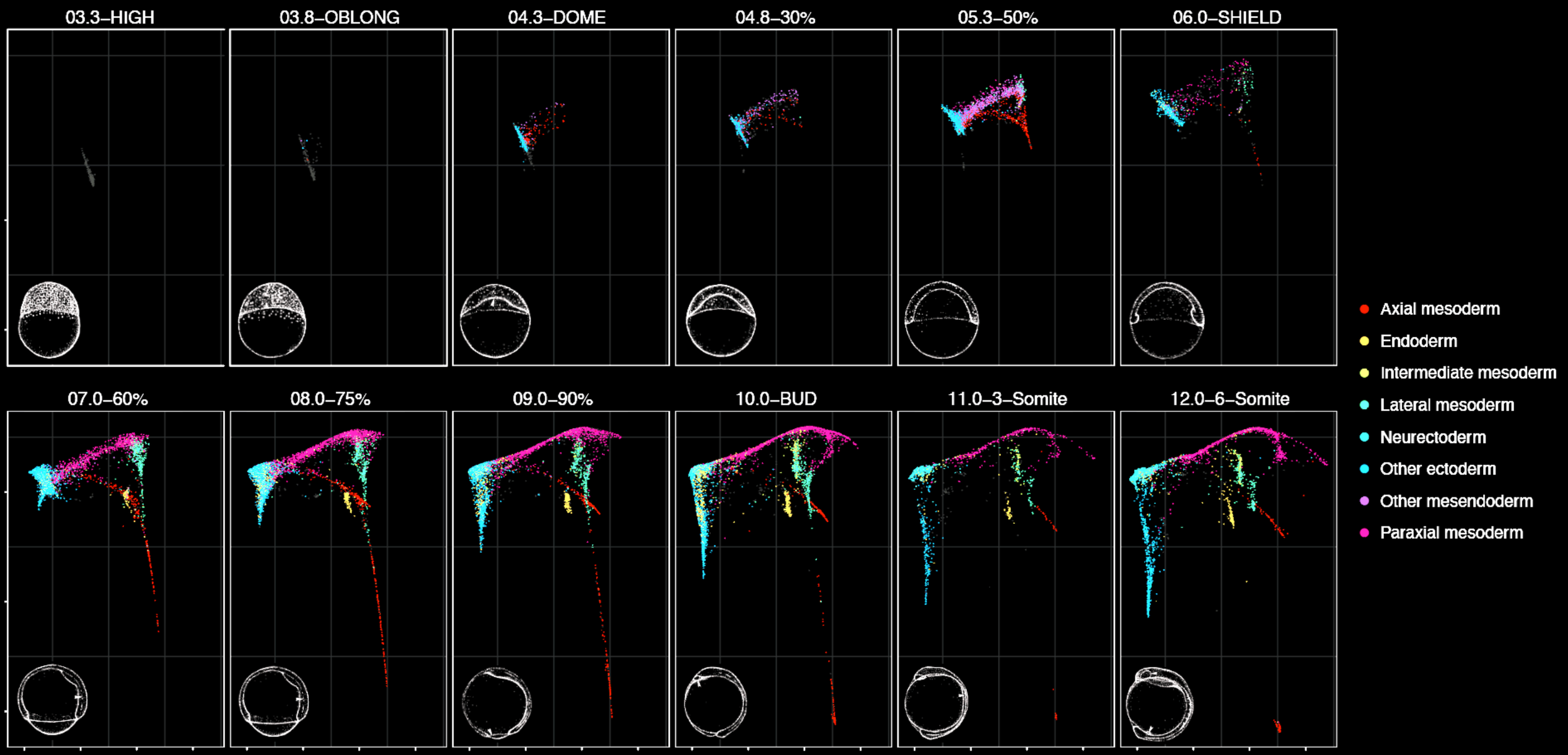

**Plass et. al.**

# GraphDR visualization of zebrafish embryonic development

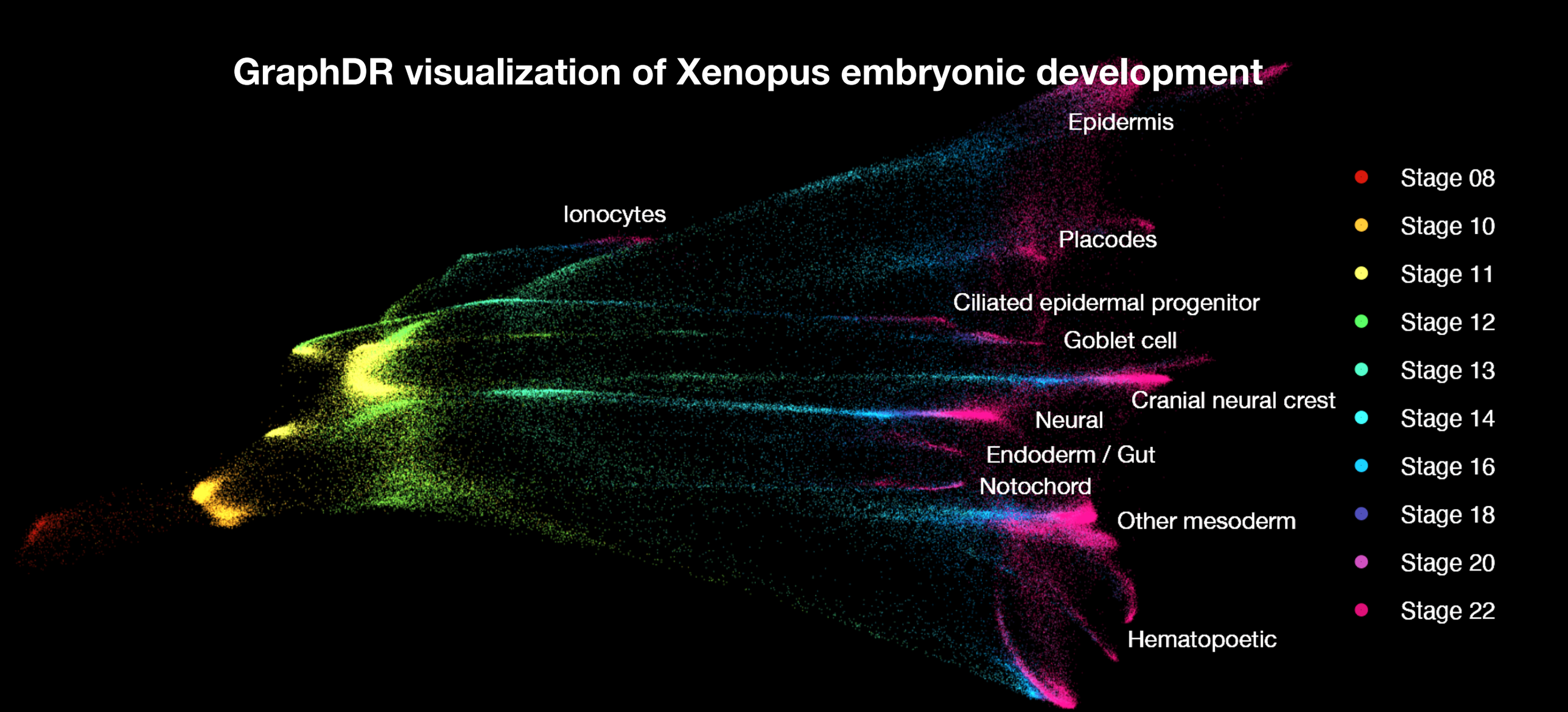


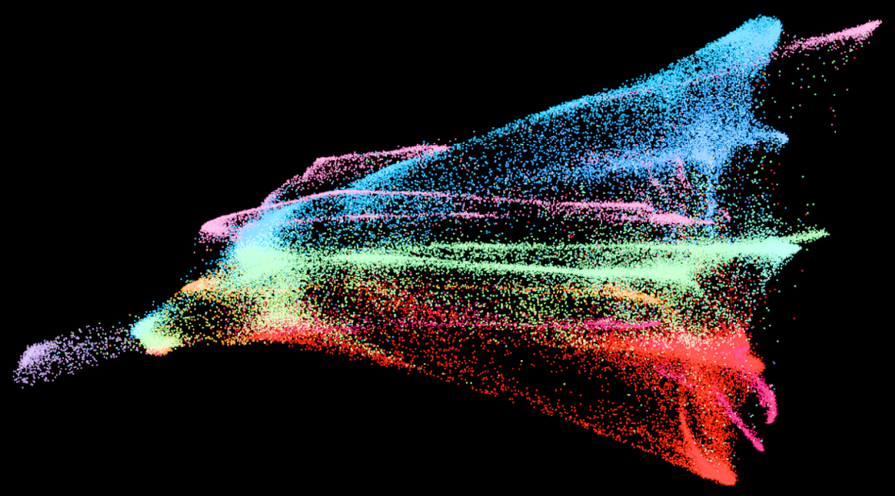

Data from Farrell et al. 2018

# Uniform interpretability allows direct comparison across temporal “slices”



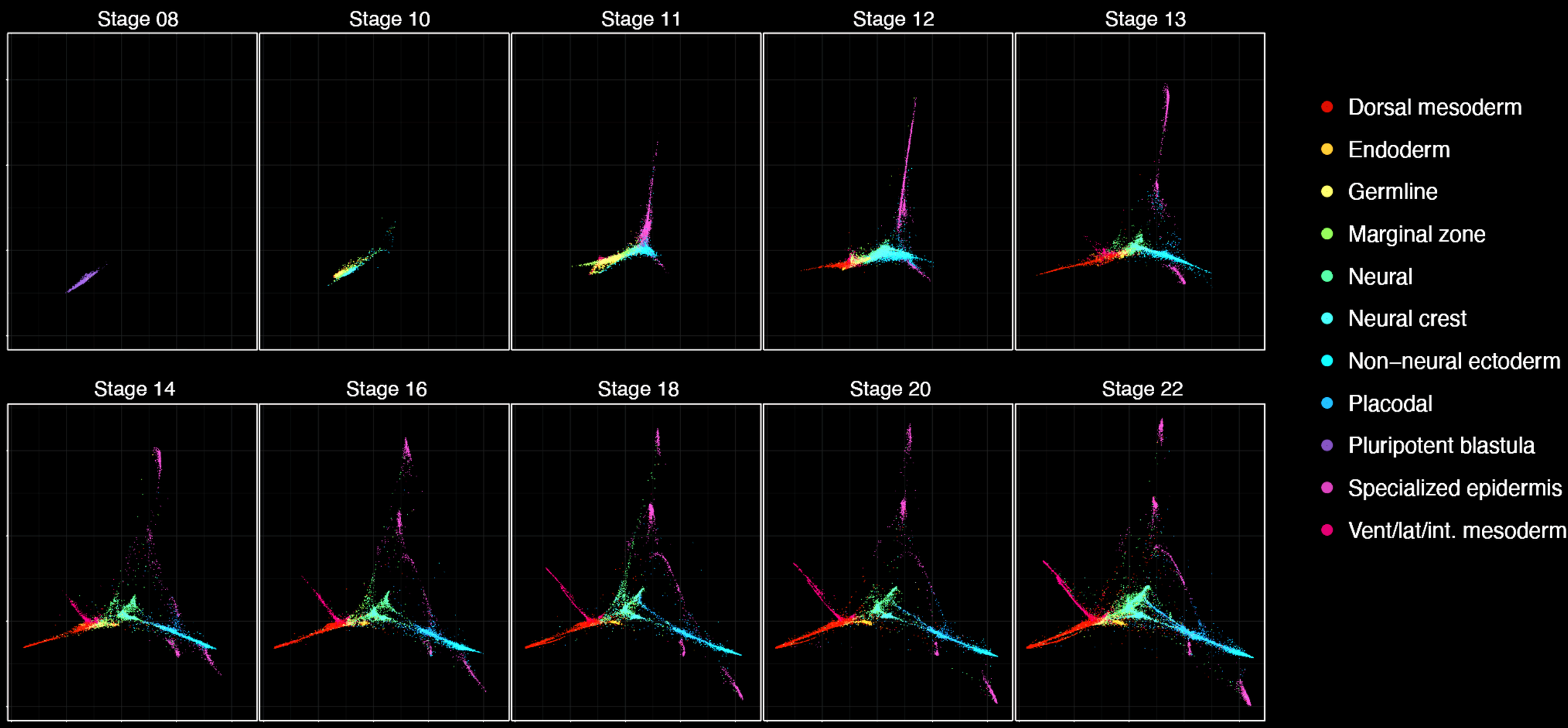
# GraphDR visualization of Xenopus embryonic development



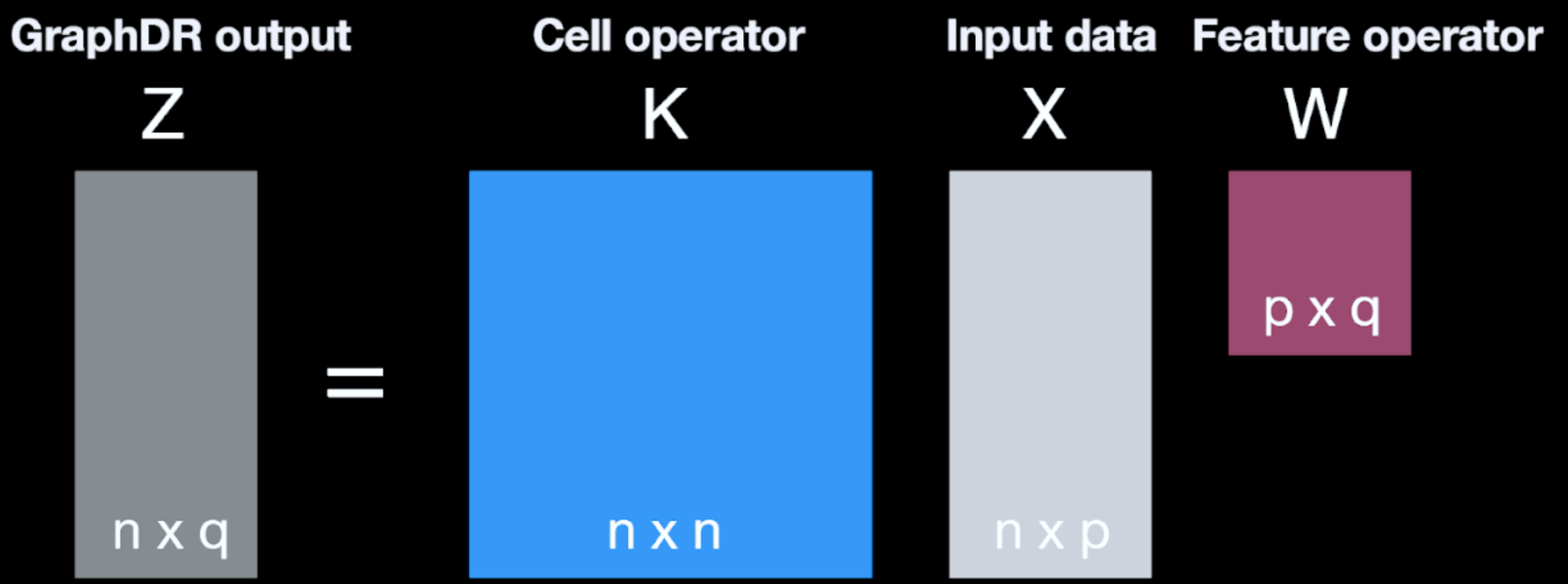
- Dorsal mesoderm
- Endoderm
- Germline
- Marginal zone
- Neural
- Neural crest
- Non–neural ectoderm
- Placodal
- Pluripotent blastula
- Specialized epidermis
- Vent/lat/int. mesoderm

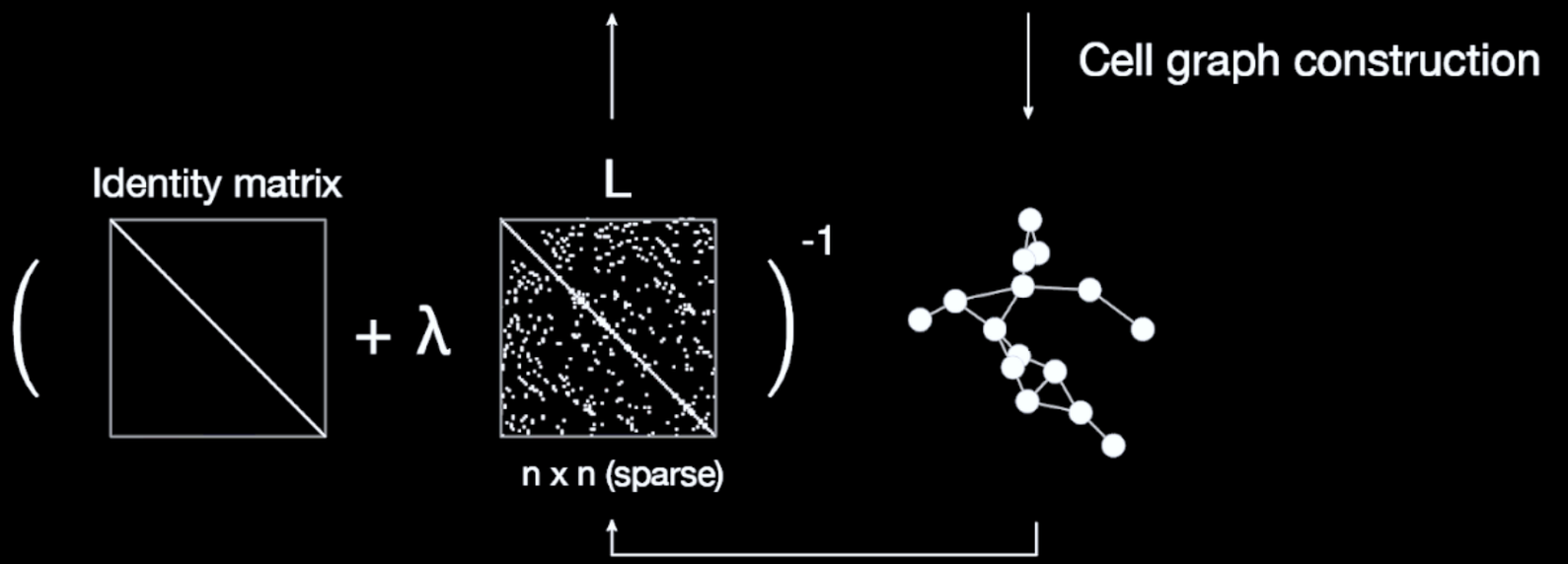
**Data from Briggs et al. 2018**


Stage 08
Stage 10
Stage 11
Stage 12
Stage 13
Stage 14
Stage 16
Stage 18
Stage 20
Stage 22
Dorsal mesoderm
Endoderm
Germline
Marginal zone
Neural
Neural crest
Non−neural ectoderm
Placodal
Pluripotent blastula
Specialized epidermis
Vent/lat/int. mesoderm

“Quasilinear” approaches:

**GraphDR - visualization and general-purpose representation**

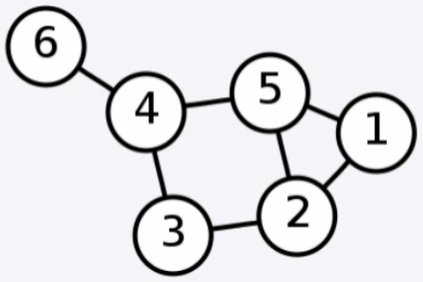



**GraphDR objective:** $\underset{W,Z}{minimize} \quad \|X - Z\mathrm{W}^{\mathrm{T}}\|_2^2 + \lambda \sum_{\{i,j\}\in G} G_{ij}\|Z_i - Z_j\|_2^2, \qquad s.t.\ W^TW = I$

New release! github.com/jzthree/quasildr

# Graph Laplacian

| Labelled graph | Degree matrix | Adjacency matrix | Laplacian matrix |
|---|---|---|---|
|  | $\begin{pmatrix} 2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 3 & 0 & 0 \\ 0 & 0 & 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$ | $\begin{pmatrix} 0 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{pmatrix}$ | $\begin{pmatrix} 2 & -1 & 0 & 0 & -1 & 0 \\ -1 & 3 & -1 & 0 & -1 & 0 \\ 0 & -1 & 2 & -1 & 0 & 0 \\ 0 & 0 & -1 & 3 & -1 & -1 \\ -1 & -1 & 0 & -1 & 3 & 0 \\ 0 & 0 & 0 & -1 & 0 & 1 \end{pmatrix}$ |



From https: // en. wikipedia. org/ wiki/ Laplacian_ matrix .

Why does it appear?

It automatically emerge when you compute sum of pairwise L2 distances

$$x^T L x = \sum_{(i,j) \in G} \|x_i - x_j\|^2$$

“Quasilinear” approaches:

**Implement GraphDR (you can do it in ~10 lines of code!)**

**GraphDR objective:** $\underset{W,Z}{minimize} \quad \|X - Z\mathrm{W}^{\mathrm{T}}\|_2^2 + \lambda \sum_{\{i,j\}\in G} G_{ij}\|Z_i - Z_j\|_2^2, \qquad s.t.\ W^TW = I$

**Analytical solution:** $Z = (I + \lambda L)^{-1}XW$

$W$ represents top d eigenvectors of $X^T(I + \lambda L)^{-1}X$

1. You may use: sklearn.neighbors.kneighbors_graph, scipy.sparse.csgraph.laplacian, scipy.sparse.eye, numpy.linalg.inv

2. Make skipping computing W an option (`no_rotation=True`):
   $Z' = (I + \lambda L)^{-1}X$ is the solution to the objective when we add the constraint $W = I$

3. Z does not have to reduce the number of dimensions in GraphDR

4. (Optional) make it scalable to > 1 million cells

Task for Day 3:

1. **Code review for Day 2 (Docstring for t-SNE)**
2. **Implement GraphDR (Code + Docstring)**
3. **Optional: 3D visualization (with plot.ly) or / and interactive visualization interface that allows adjusting regularization parameter (e.g. with Dash)**