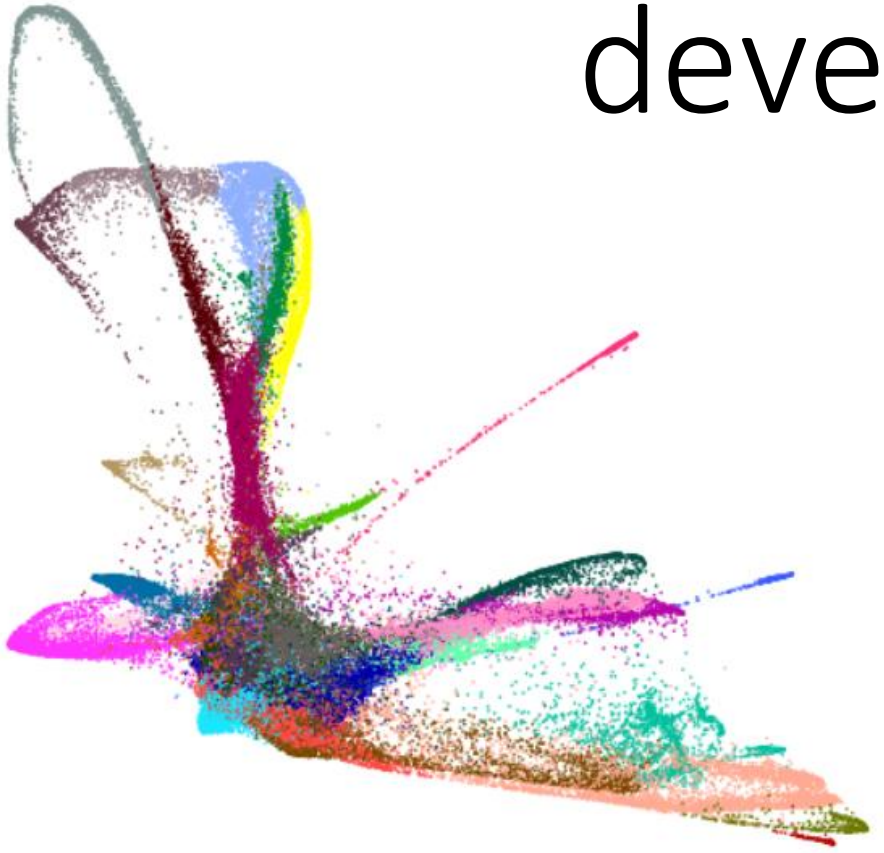
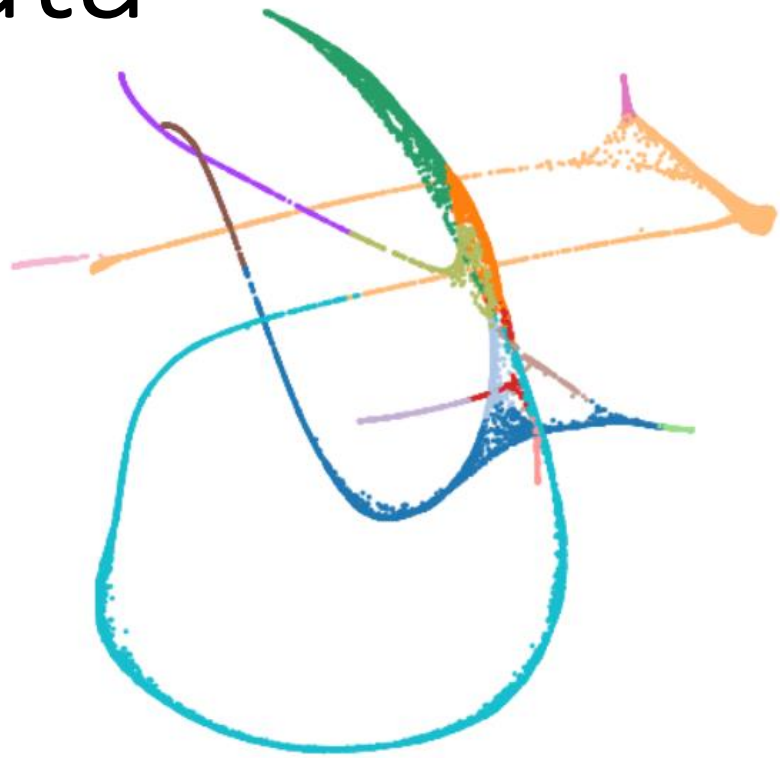
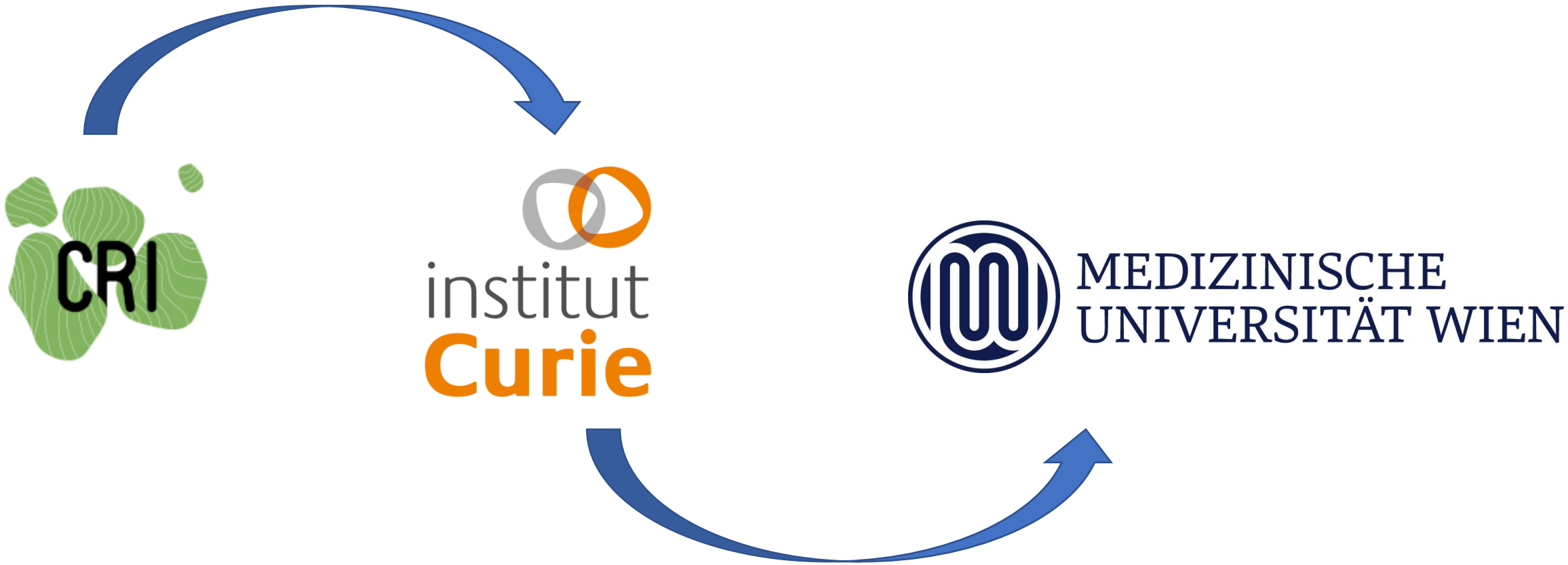
Deciphering dynamical developmental data

Louis Faure - PhD

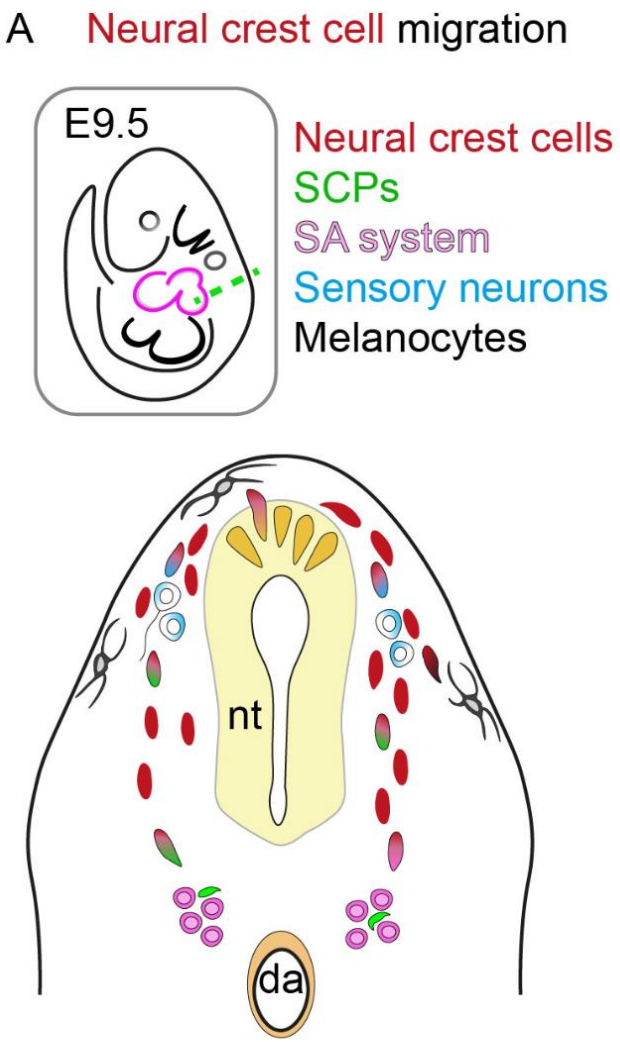




My background

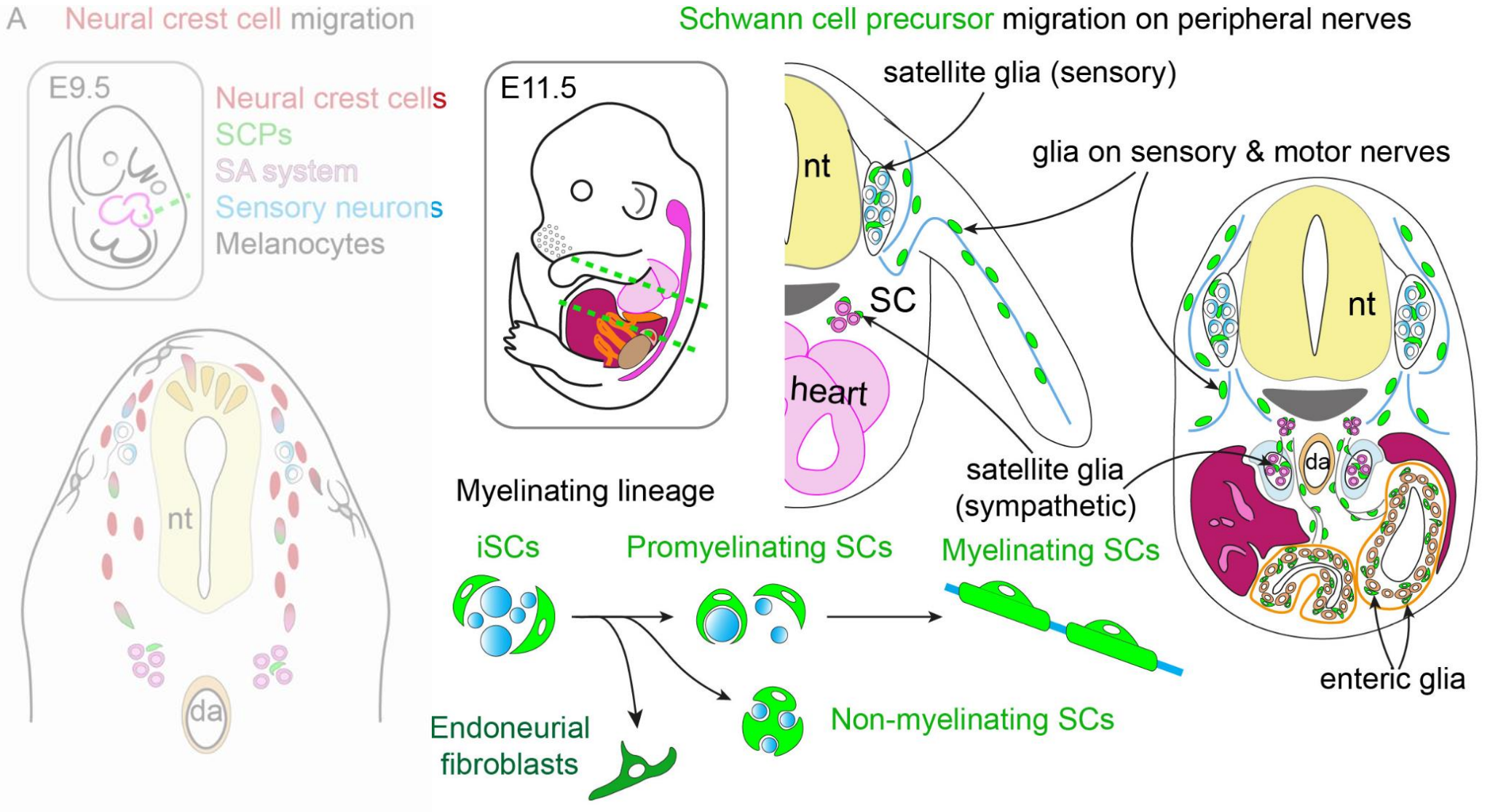


My PhD project



Kastriti M. et al. Manuscript in preparation.

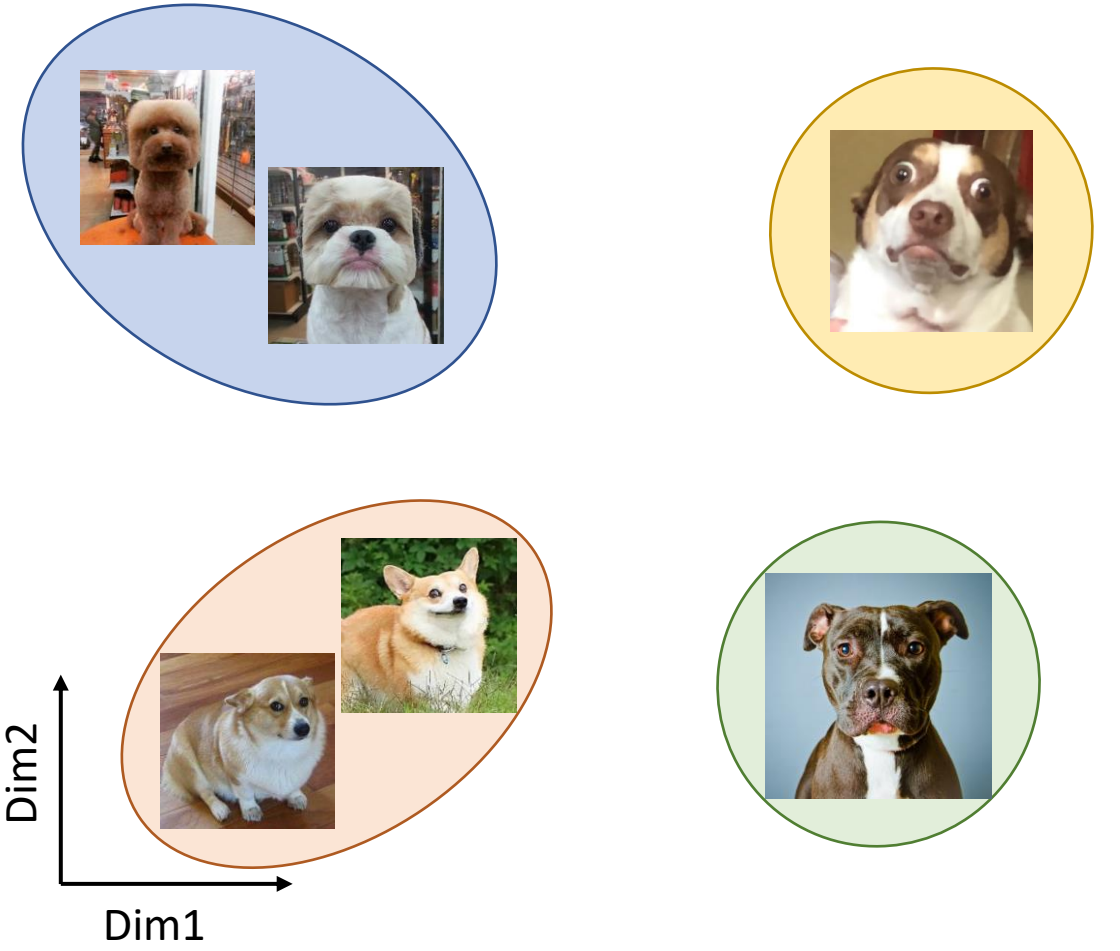
My PhD project



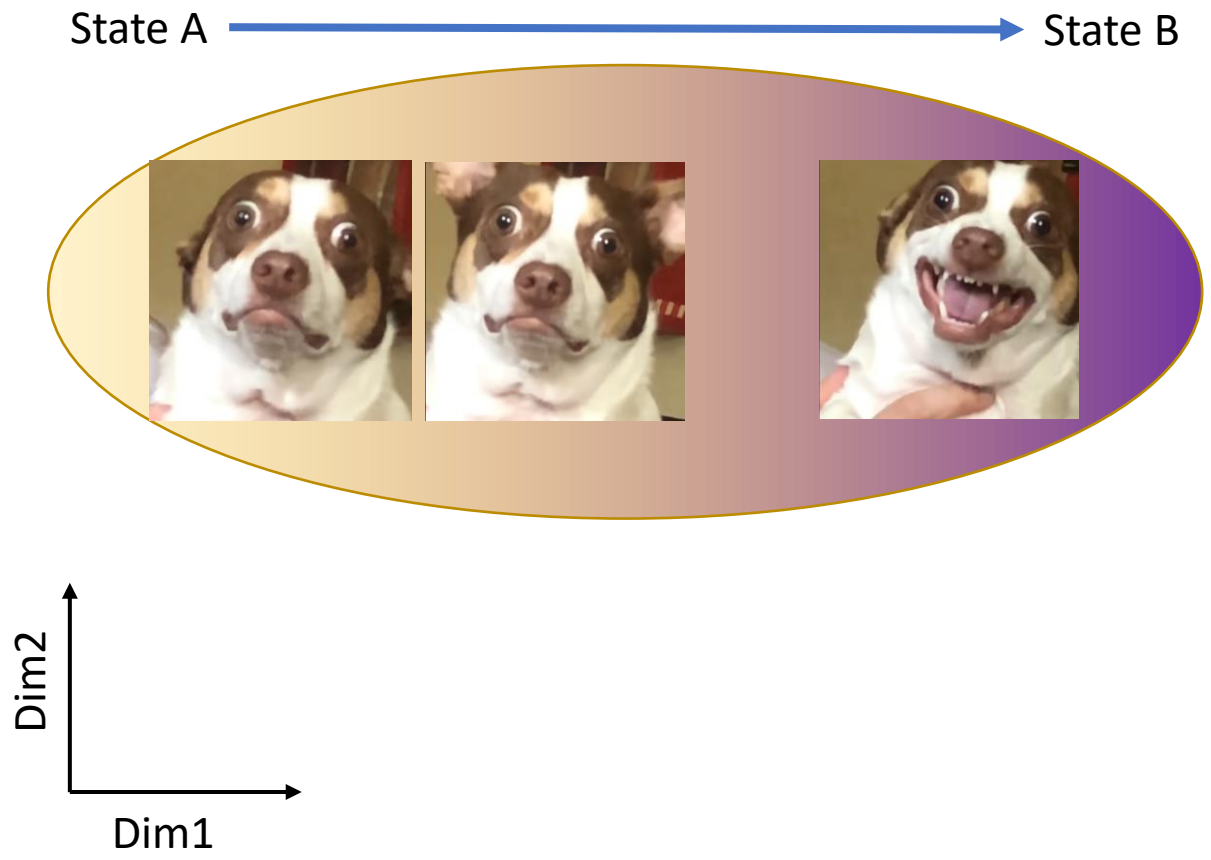
Kastriti M. et al. Manuscript in preparation.

Static populations versus dynamical ones

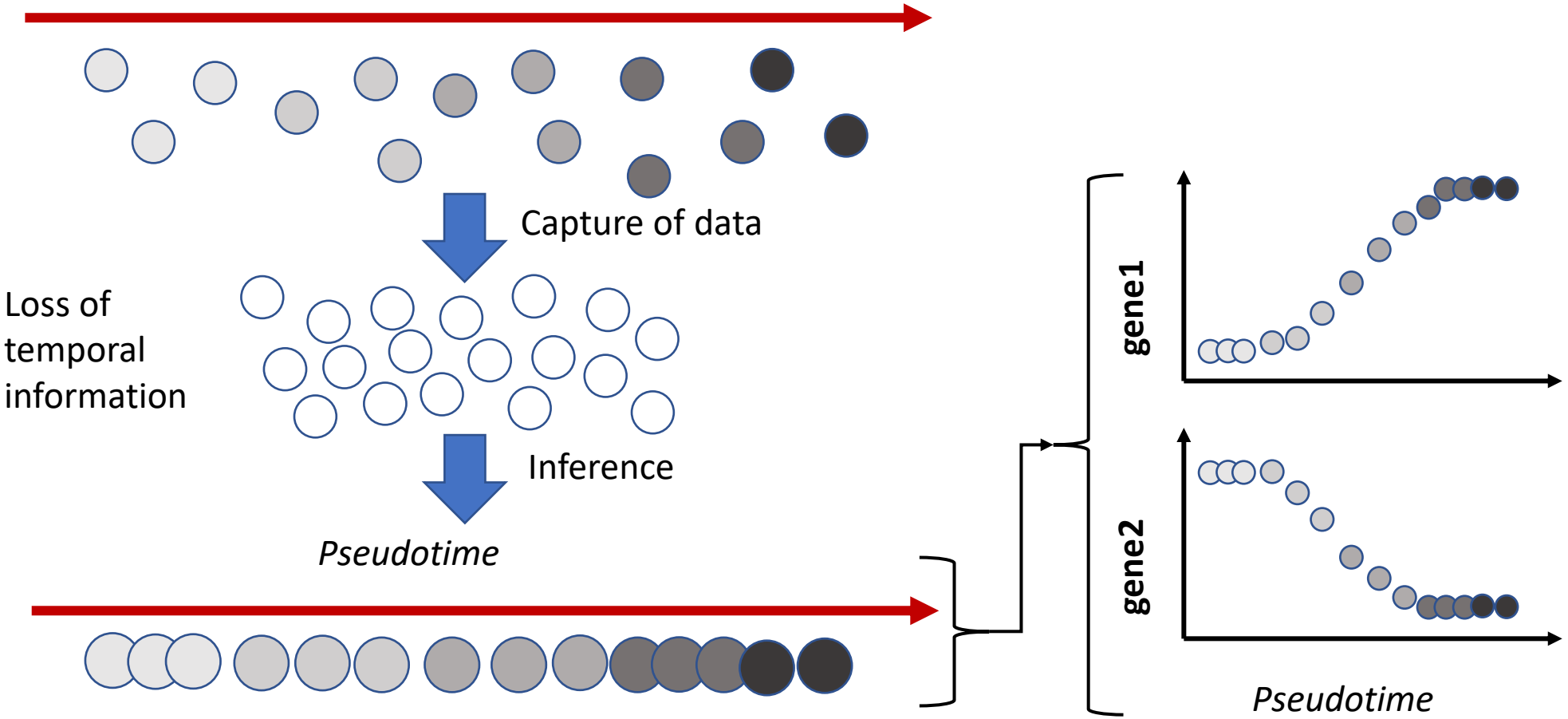
Static = well defined cell-type/clusters



Dynamic = definition of clusters is blurred



Concept of pseudotime ordering

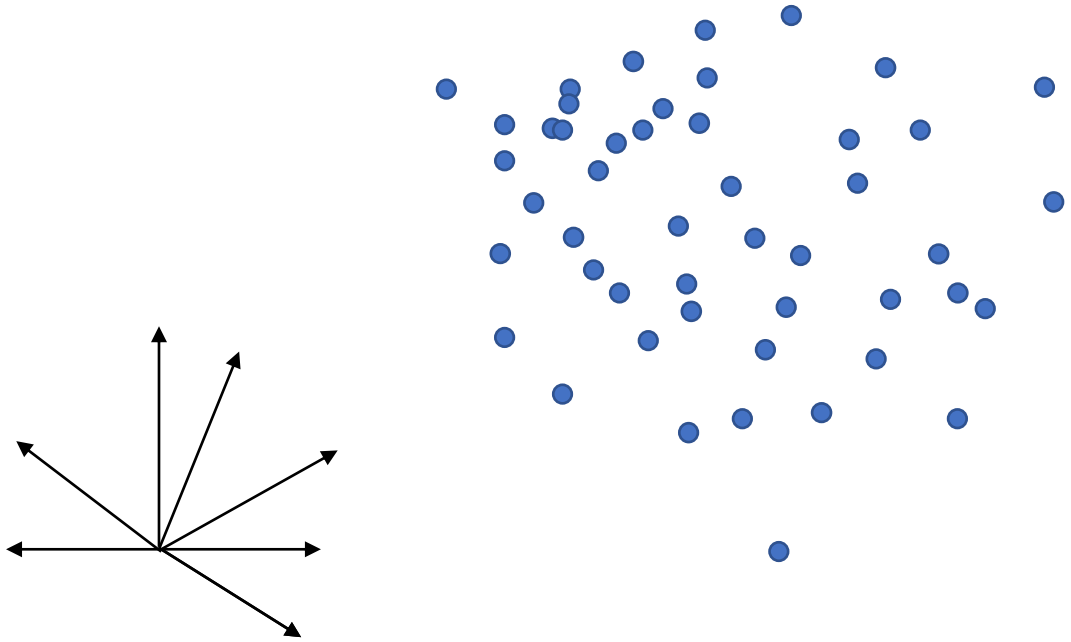


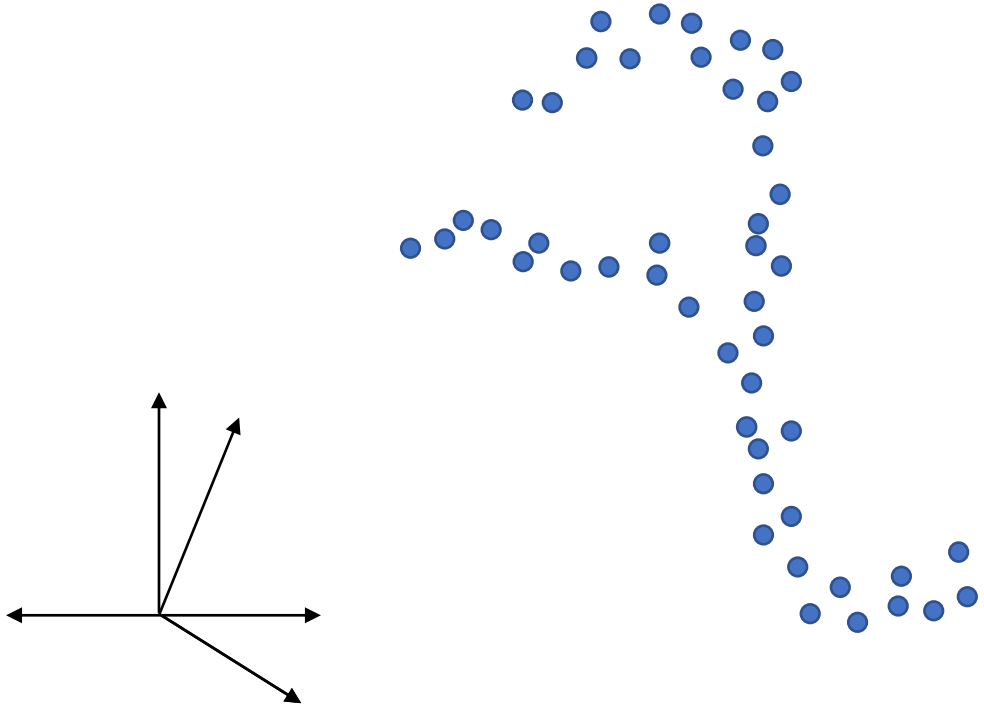
How do we apply this to scRNAseq data?

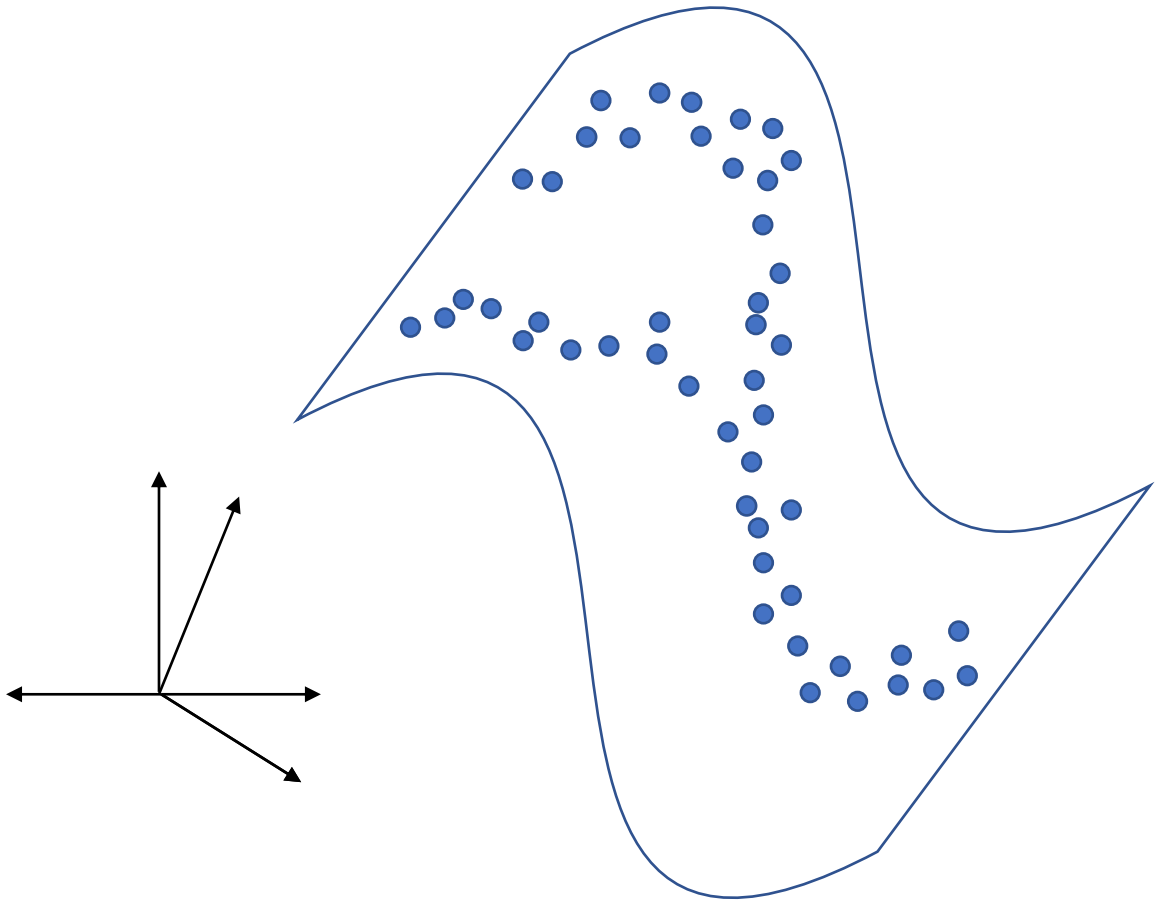
- scRNAseq is highly noisy data
  - Amplification bias
  - Library size differences
  - Sequencing depth
  - **Biology is stochastic and noisy by itself!**

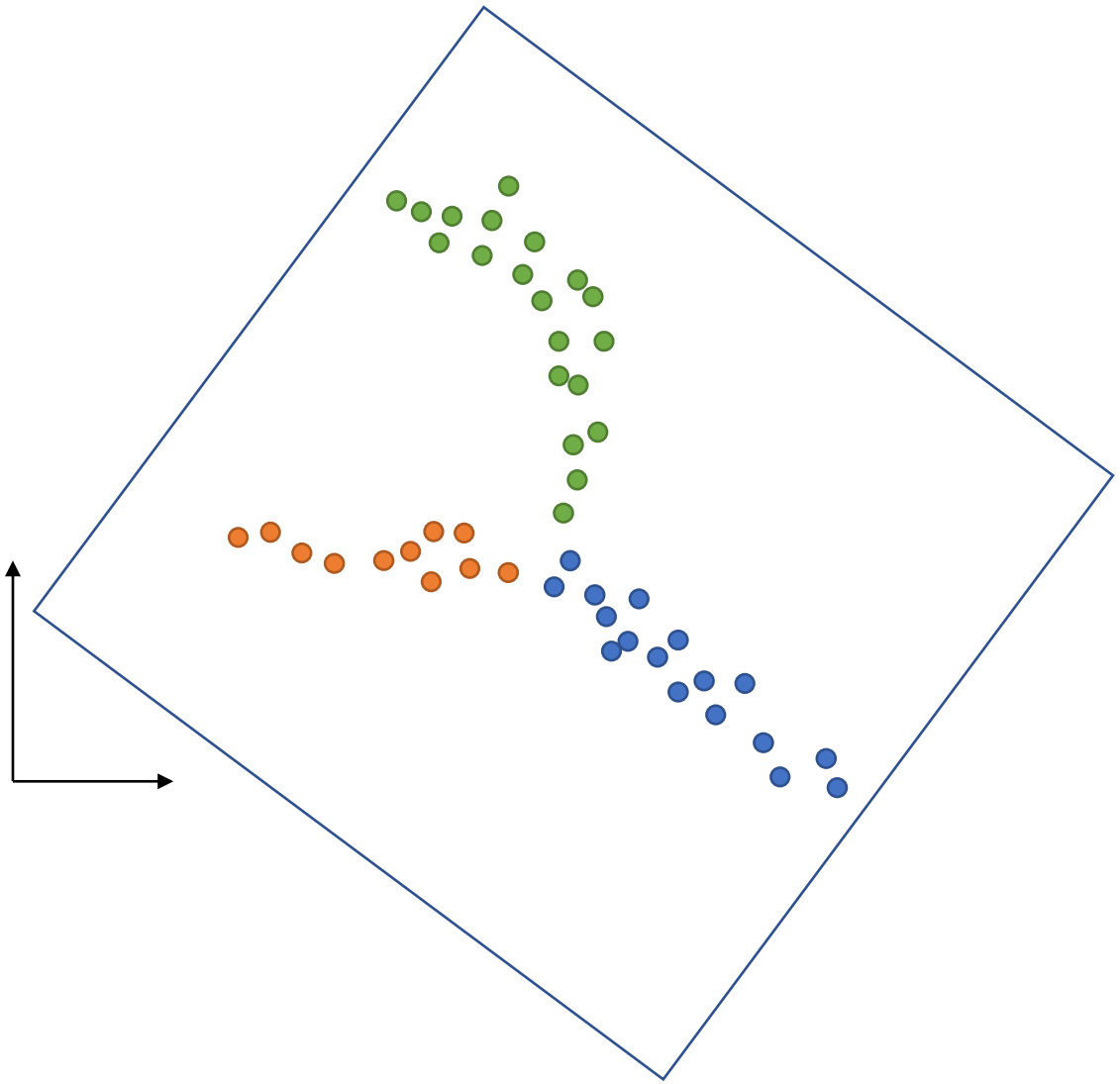
- But the good news is that transcriptional activity happens in **modules composed numerous genes**
  - Latent structure exists at a much lower dimension

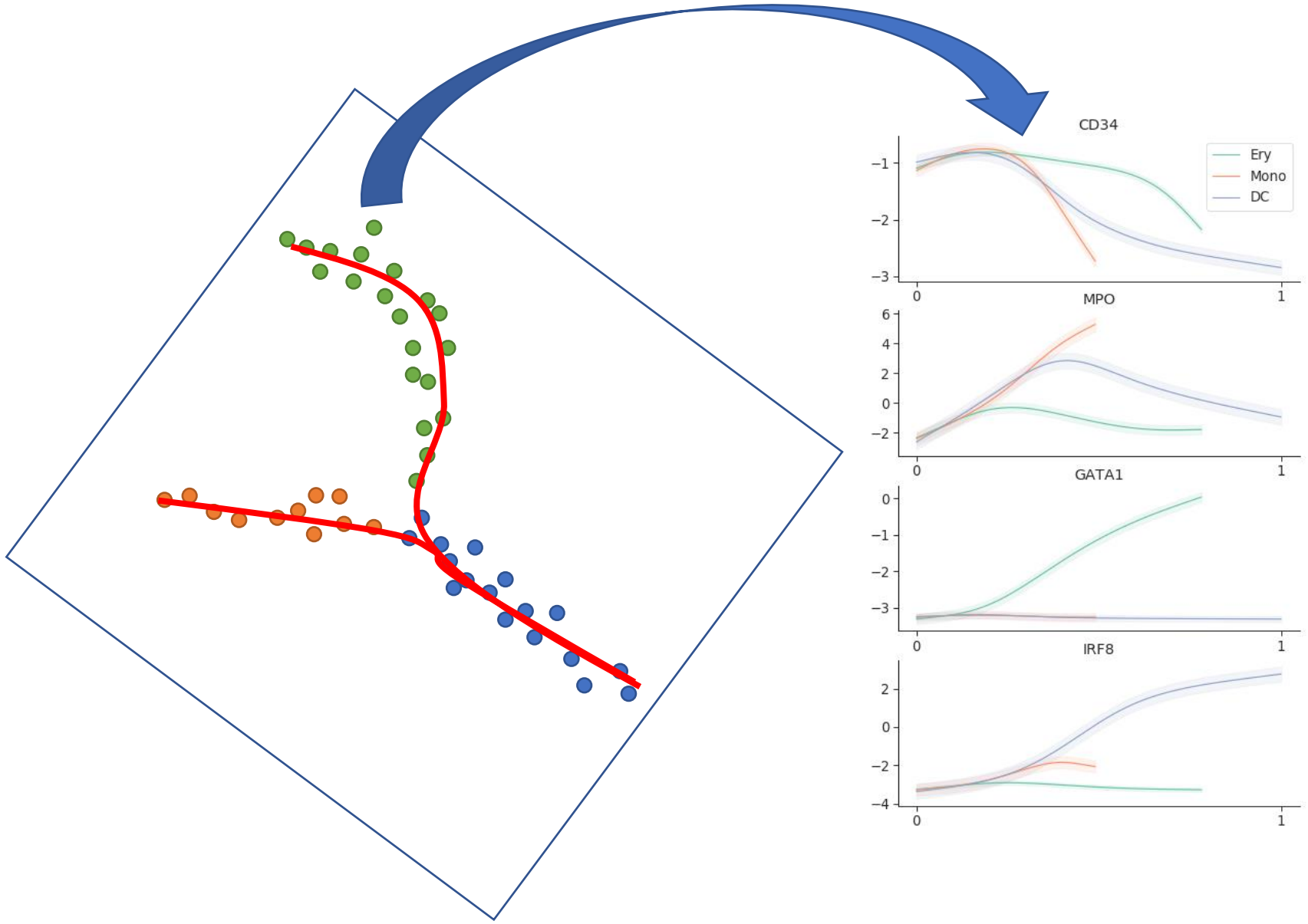
Additional processing steps are required to “denoise” and extract the latent structure of the data










CD34
Ery
Mono
DC
MPO
GATA1
IRF8

Denoising data: PCA

Highly dimensional data
n samples
m features

→ Obtain **covariance matrix** A *of size* $n \times n$

→ **eigen decompose** A by diagonalizing:
$\boldsymbol{A = XDX^{-1}}$

D is a diagonal matrix made of **eigenvalues**

$$\begin{bmatrix} \lambda_1 & 0 & 0 \\ 0 & \lambda_2 & 0 \\ 0 & 0 & \lambda_n \end{bmatrix}$$

Eigenvalues are found by solving $|A - \lambda I| = 0$ where I is an identity matrix

X is a matrix with each columns being **eigenvectors** of A

Eigenvector $\overrightarrow{x_n}$ are found by solving $(A - \lambda I)\overrightarrow{x_n} = \vec{0}$ for each eigenvalues λ_n previously found.

Important!

- We have decomposed our data into linear transformations with:
  - the eigenvectors being the "axes" or the "directions“
  - the eigenvalues being the scaling factors
- Each eigenvector has its own eigenvalue
- By choosing a reduced number among the top eigenvalues, we subset our eigenvectors columns which becomes our **reduced dimensions**!

In this case, dimensions are principal components

Denoising data: PCA

“Highly” dimensional data

20 samples

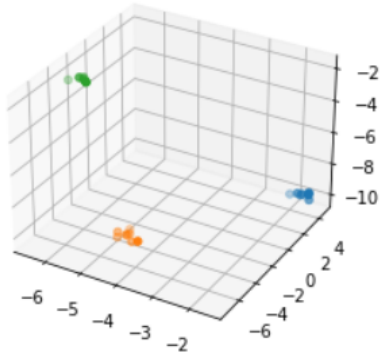
3 features

https://github.com/LouisFaure/Trajectory_Inference_workshop/blob/main/01.%20Eigen-decomposition.ipynb

```
[1]: import numpy as np
     from sklearn.datasets import make_blobs
     blobs,clusters=make_blobs(n_samples=20,
                               n_features=3,
                               cluster_std=.2,
                               random_state=1)
```

```
[2]: blobs=blobs[np.argsort(clusters),:]
     clusters=clusters[np.argsort(clusters)]
```

```
[3]: import matplotlib.pyplot as plt
     fig = plt.figure()
     ax = fig.add_subplot(111,
                          projection='3d')

     for c in range(3):
         ax.scatter(blobs[clusters==c,0],
                    blobs[clusters==c,1],
                    blobs[clusters==c,2])
```



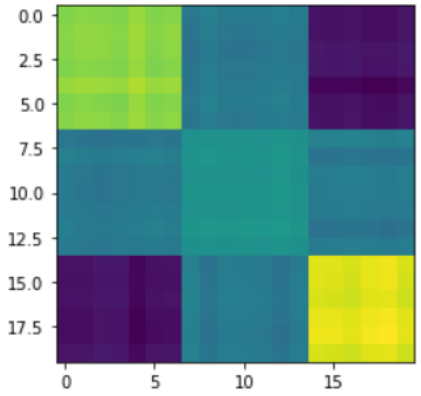
Obtain **covariance matrix** A *of size* $n \times n$

**data is scaled to remove the effect of the magnitude of the variables*

```
[4]: from sklearn.preprocessing import *

     scaler = StandardScaler()
     blobs_scaled = scaler.fit_transform(blobs)
```

```
[5]: import numpy as np
     cov = np.cov(blobs_scaled)
```

```
[6]: plt.imshow(cov);
```

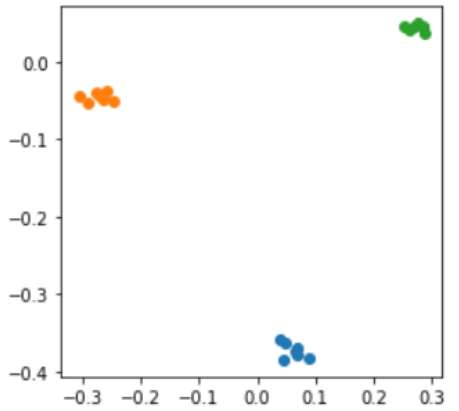


→ **eigen decompose** A by diagonalizing:

$\boldsymbol{A = XDX^{-1}}$

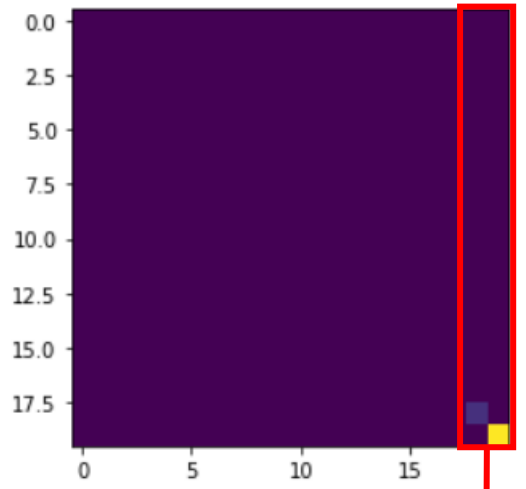
**the two last columns can be accessed via -2 and -1*

```
[9]: plt.figure(figsize=(4,4))
     for c in range(3):
         plt.scatter(
             X[clusters==c,-2],
             X[clusters==c,-1])
```

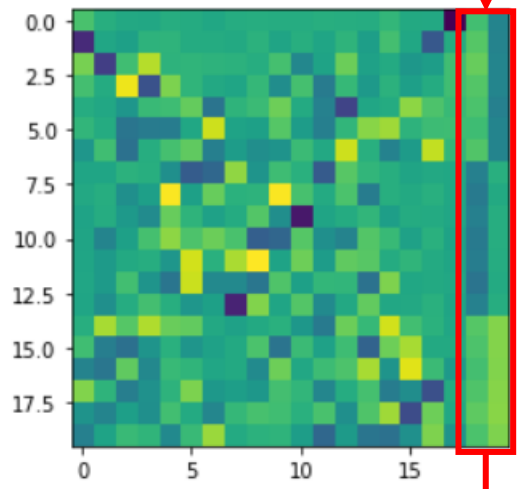


```
[7]: D,X = np.linalg.eigh(cov)
```

```
[8]: plt.imshow(np.diag(D));
```

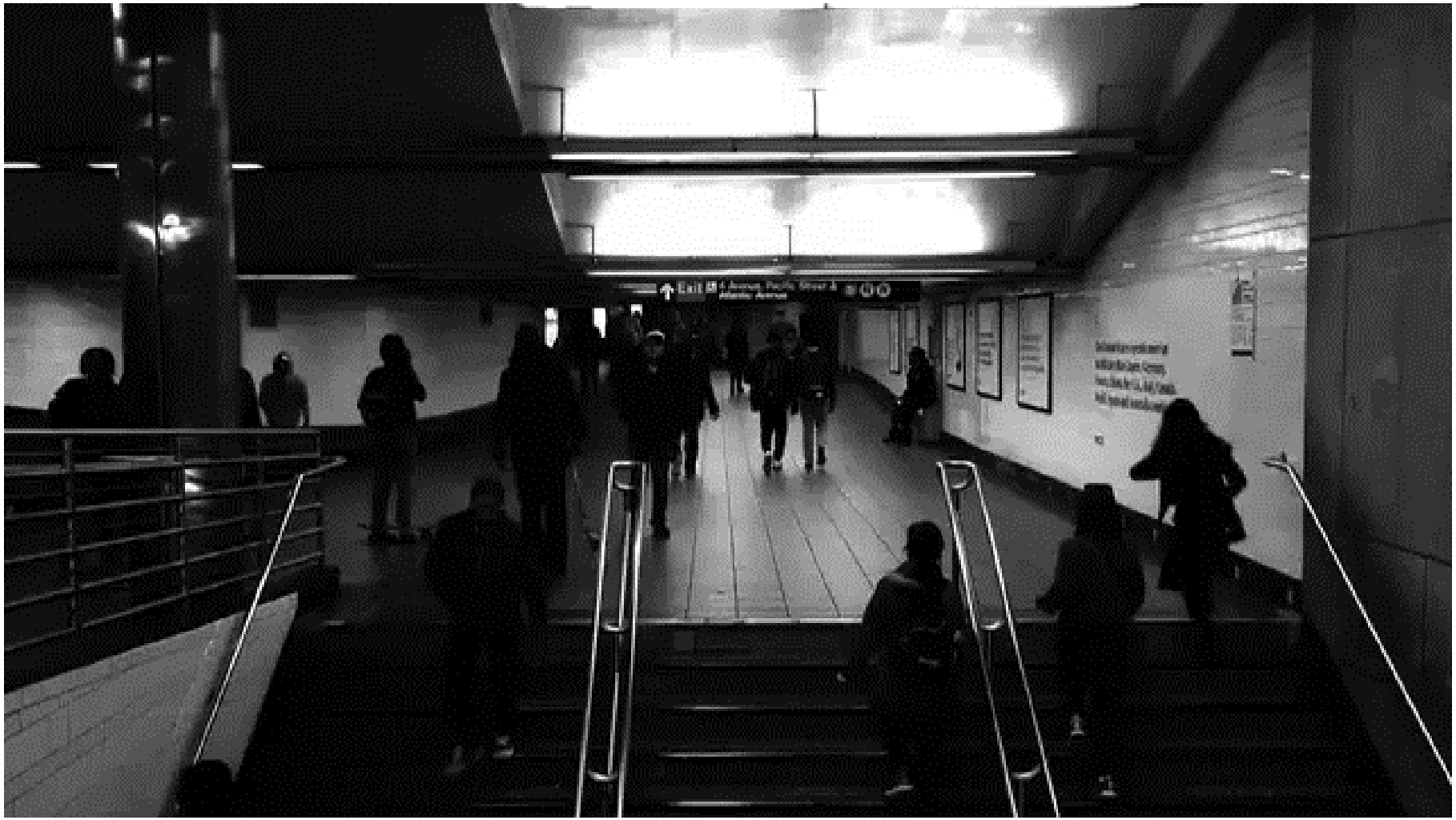


```
[9]: plt.imshow(X);
```

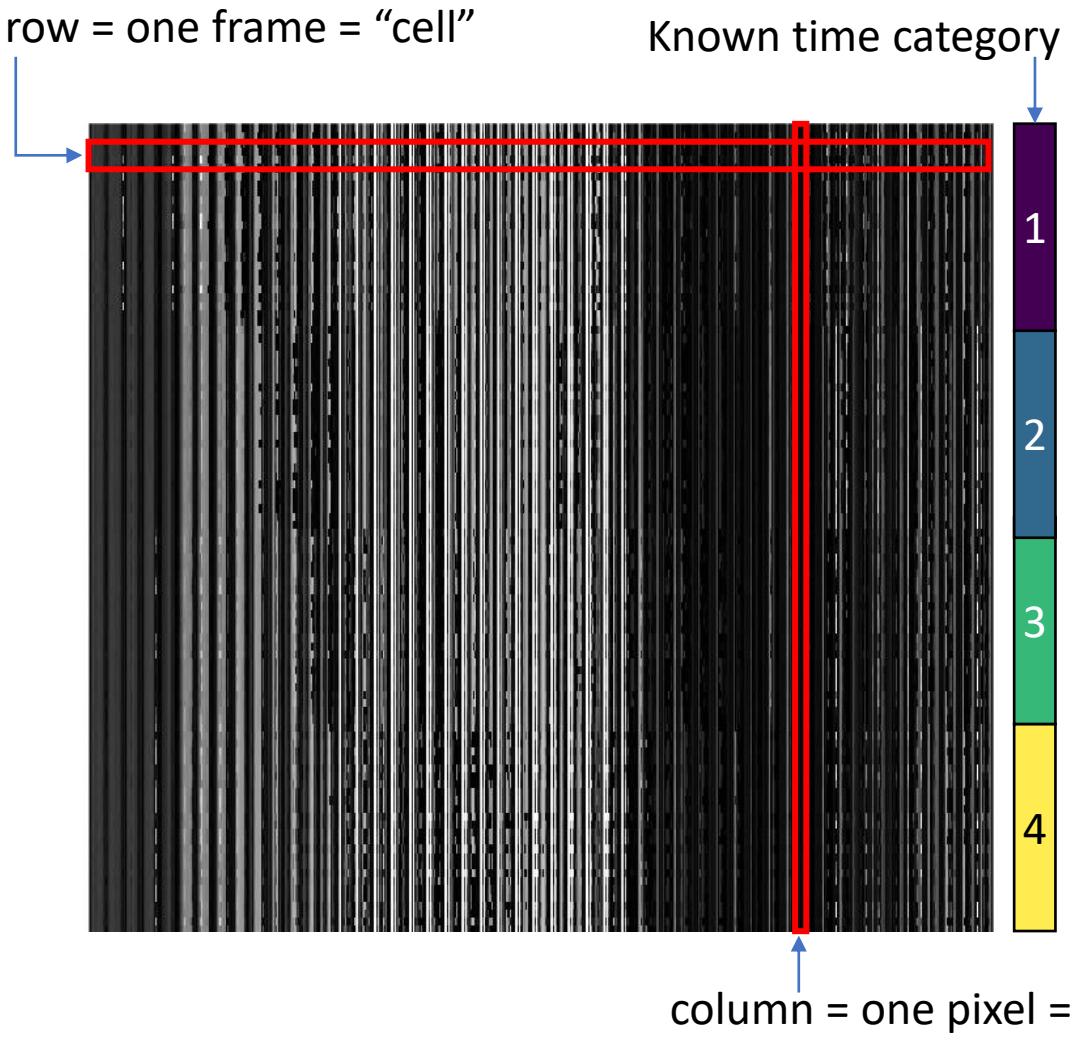


Select reduced dimensions

Pseudotime: recovering lost time information



Considering our video as our “count matrix”



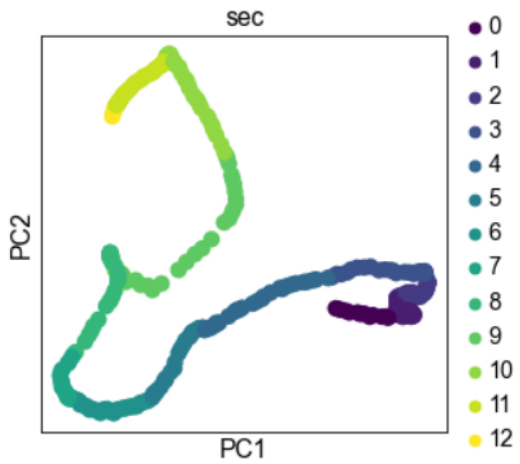

```
[1]: import matplotlib.pyplot as plt
     import matplotlib.image as mpimg
     import numpy as np
     import glob
     import scanpy as sc

[2]: files=glob.glob('frames/*.jpg')

[3]: def rgb2gray(path):
         return np.dot(mpimg.imread(path)[...,:3], [0.2989, 0.5870, 0.1140]).ravel()

     mat=np.vstack(list(map(rgb2gray,files)))

[4]: adata=sc.AnnData(mat)

[5]: adata

[5]: AnnData object with n_obs × n_vars = 349 × 230400

[6]: adata.obs["sec"]=[round(int(f.split("_")[1].split(".")[0])/30) for f in files]
     adata.obs["sec"]=adata.obs["sec"].astype("category")
```

https://github.com/LouisFaure/Trajectory_Inference_workshop/blob/main/02.%20Trajectory%20inference%20on%20a%20video.ipynb

Dimensionality reduction methods help revealing dynamical axis

Running PCA on scaled values

```
[7]: adata.obsm["X_pca"]=sc.pp.pca(sc.pp.scale(adata.X,copy=True),n_comps=20)
     sc.set_figure_params()
     sc.pl.pca(adata,color='sec',palette="viridis")
```



Infer a principal graph in PCA space

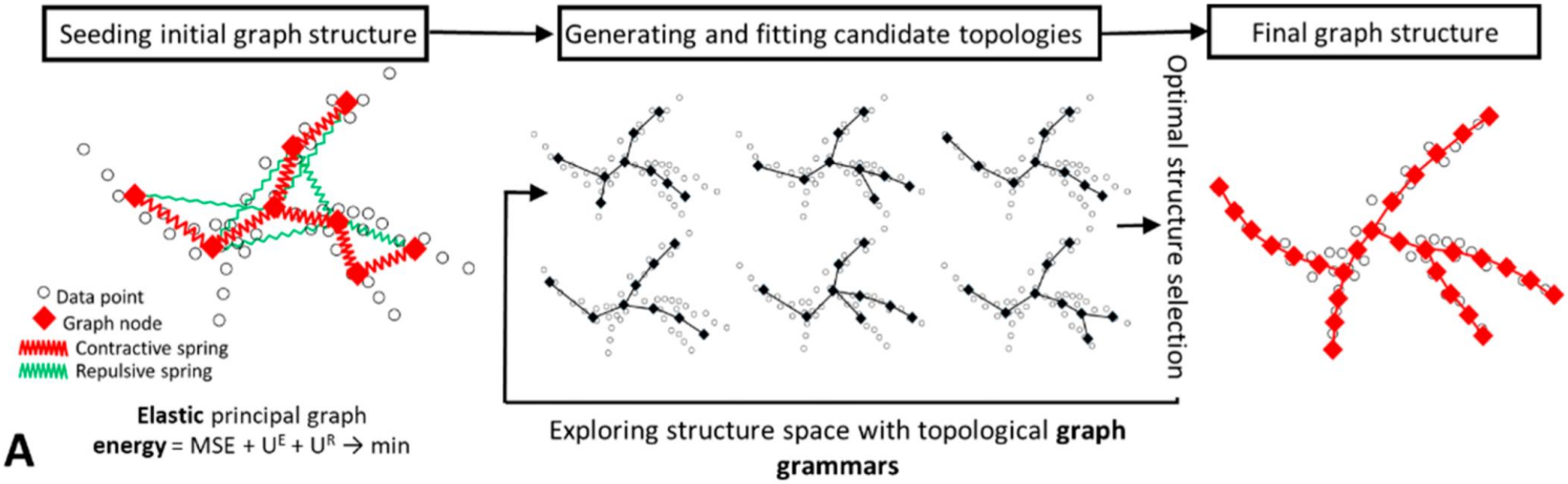
```
[9]: import scFates as scf
```

```
[10]: scf.tl.curve(adata,use_rep='X_pca',Nodes=30,seed=42)
```

```
inferring a principal tree --> parameters used
    30 principal points, mu = 0.1, lambda = 0.01
    finished (0:00:00) --> added
    .uns['epg'] dictionnary containing inferred elastic curve generated from elpigraph.
    .uns['graph']['B'] adjacency matrix of the principal points.
    .uns['graph']['R'] hard assignment of cells to principal point in representation space.
    .uns['graph']['F'], coordinates of principal points in representation space.
```

https://github.com/LouisFaure/Trajectory_Inference_workshop/blob/main/02.%20Trajectory%20inference%20on%20a%20video.ipynb

ElPiGraph

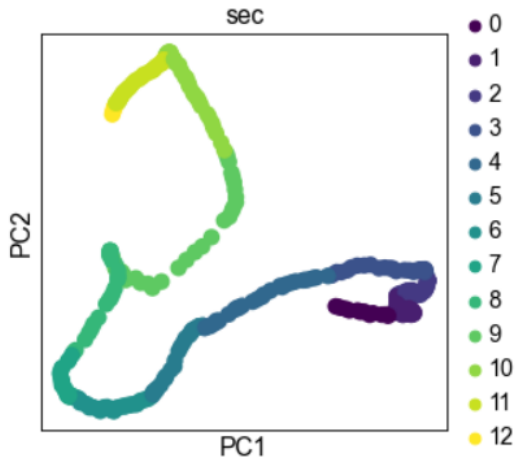


Albergante *et al.* (2019), *Robust and Scalable Learning of Complex Intrinsic Dataset Geometry via ElPiGraph*, Entropy.

Dimensionality reduction methods help revealing dynamical axis

Running PCA on scaled values

```
[7]: adata.obsm["X_pca"]=sc.pp.pca(sc.pp.scale(adata.X,copy=True),n_comps=20)
     sc.set_figure_params()
     sc.pl.pca(adata,color='sec',palette="viridis")
```



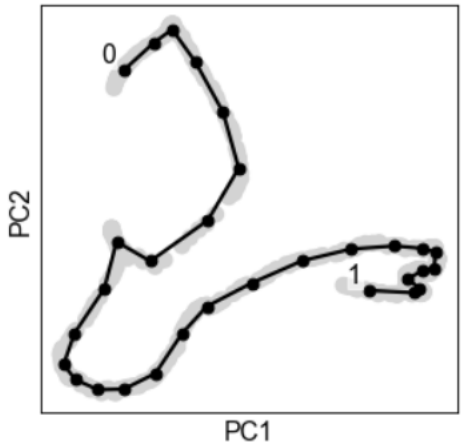
Infer a principal graph in PCA space

```
[9]: import scFates as scf
```

```
[10]: scf.tl.curve(adata,use_rep='X_pca',Nodes=30,seed=42)
```

```
inferring a principal tree --> parameters used
    30 principal points, mu = 0.1, lambda = 0.01
    finished (0:00:00) --> added
    .uns['epg'] dictionnary containing inferred elastic curve generated from elpigraph.
    .uns['graph']['B'] adjacency matrix of the principal points.
    .uns['graph']['R'] hard assignment of cells to principal point in representation space.
    .uns['graph']['F'], coordinates of principal points in representation space.
```

```
[11]: scf.pl.graph(adata,basis="pca")
```



https://github.com/LouisFaure/Trajectory_Inference_workshop/blob/main/02.%20Trajectory%20inference%20on%20a%20video.ipynb

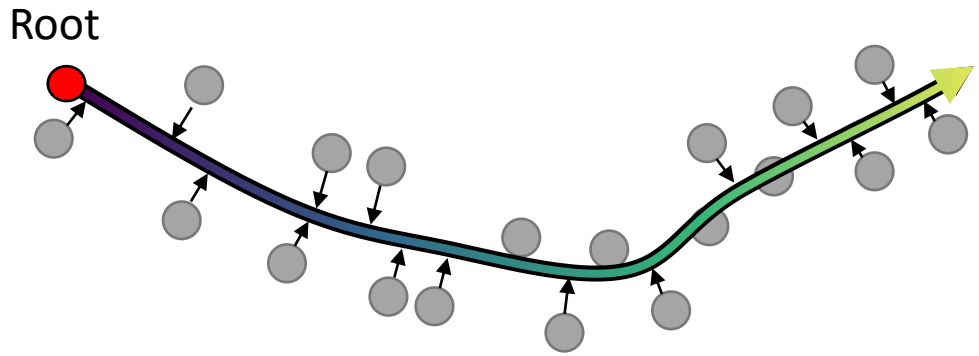
Pseudotime consists in reordering the cells along an inferred trajectory

```
[12]: scf.tl.root(adata,1)
```

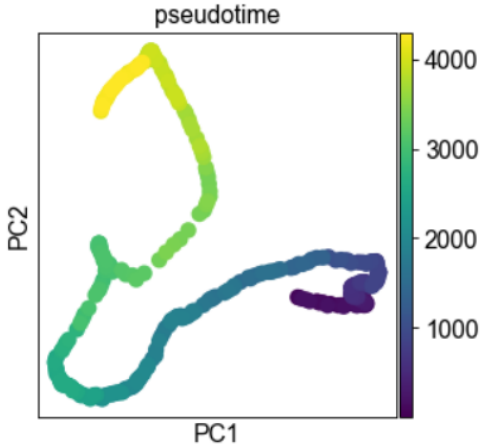
```
root selected --> added
    .uns['graph']['root'] selected root.
    .uns['graph']['pp_info'] for each PP, its distance vs root and segment assignment.
    .uns['graph']['pp_seg'] segments network information.
```

```
[13]: scf.tl.pseudotime(adata)
```

```
projecting cells onto the principal graph
    finished (0:00:00) --> added
    .obs['edge'] assigned edge.
    .obs['t'] pseudotime value.
    .obs['seg'] segment of the tree assigned.
    .obs['milestones'] milestone assigned.
    .uns['pseudotime_list'] list of cell projection from all mappings.
```

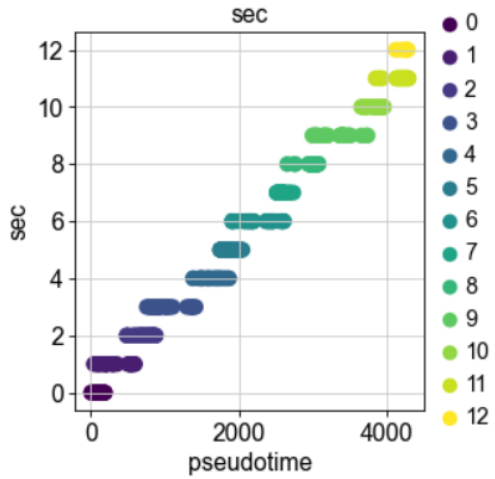



```
[14]: sc.pl.pca(adata,color="t",title='pseudotime')
```




```
[15]: ax=sc.pl.scatter(adata,x="t",y="sec",color="sec",show=False)
      ax.set_xlabel("pseudotime")
```

```
[15]: Text(0.5, 0, 'pseudotime')
```



https://github.com/LouisFaure/Trajectory_Inference_workshop/blob/main/02.%20Trajectory%20inference%20on%20a%20video.ipynb

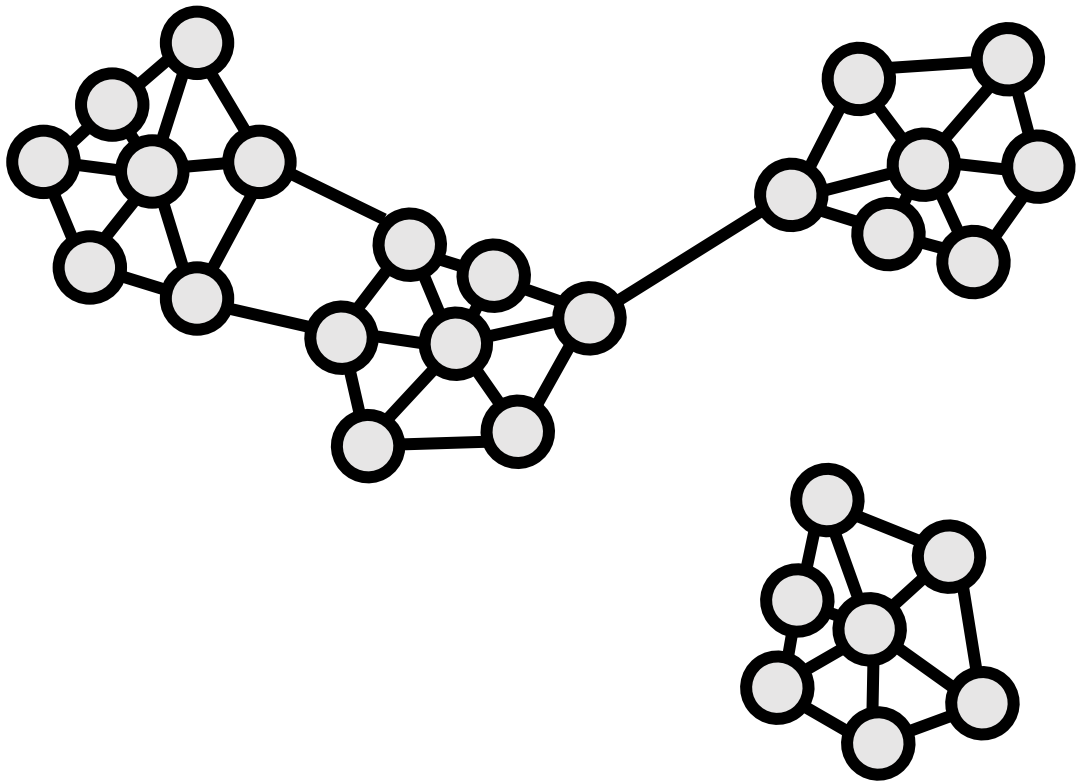
Recovered time! Yay!

Denoising scRNAseq data: beyond PCA

PCA → Distance matrix → Knn graph

pairwise distances

Knn graph

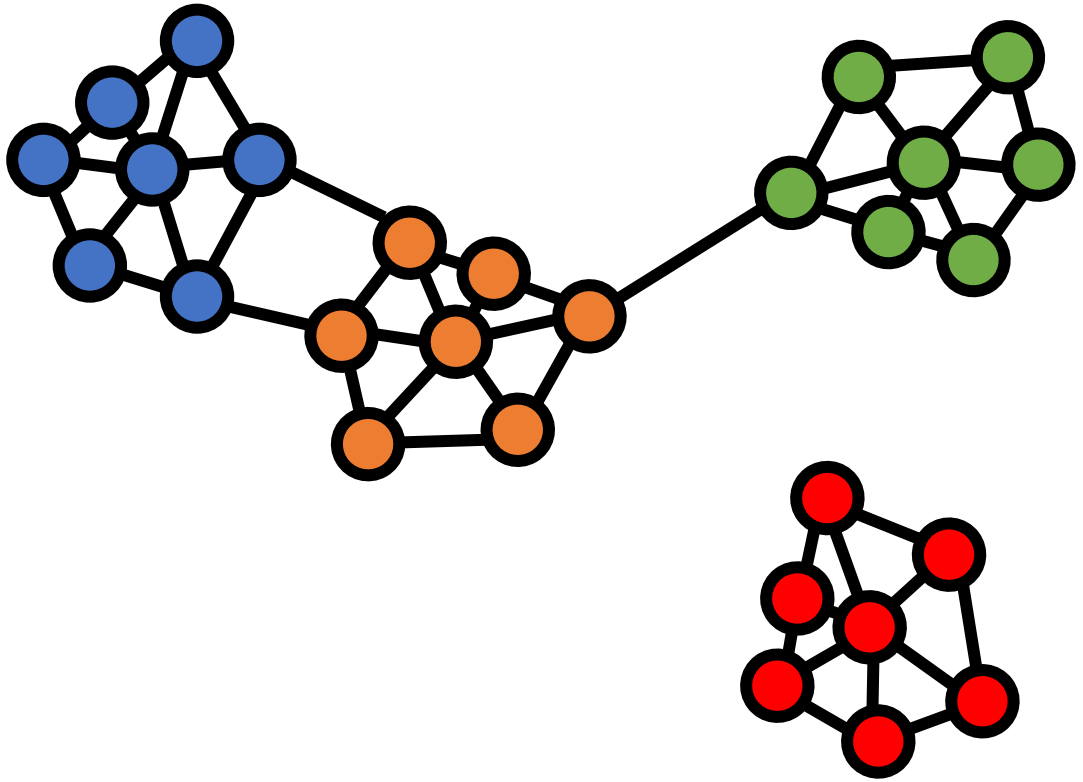


Denoising scRNAseq data: beyond PCA

PCA → Distance matrix → Knn graph + clustering

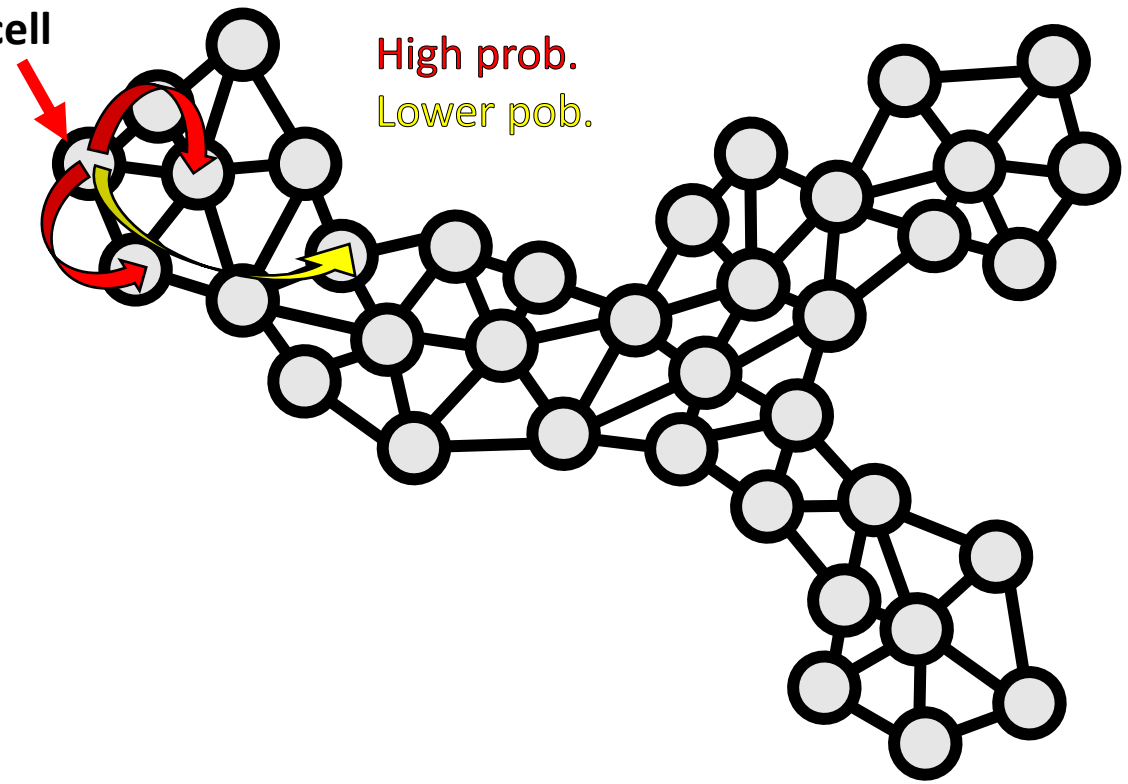
pairwise distances

Knn graph



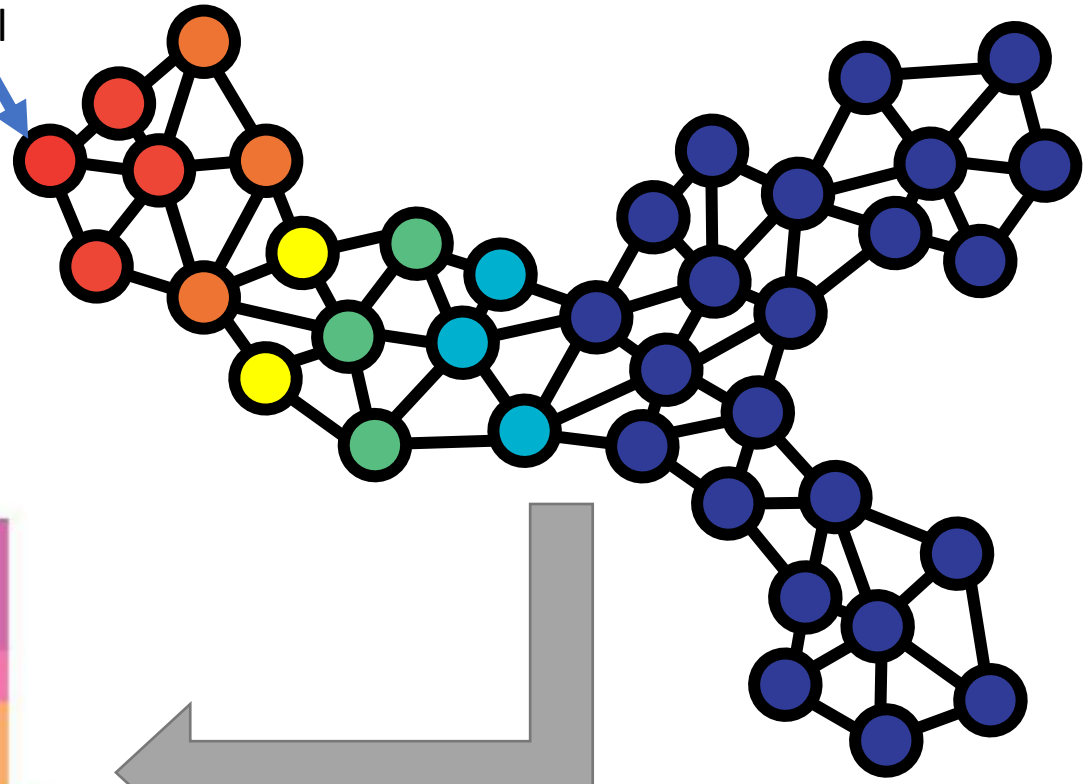
Denoising developmental scRNAseq data

PCA → *pairwise distances* → Distance matrix → *Knn graph* → Knn graph

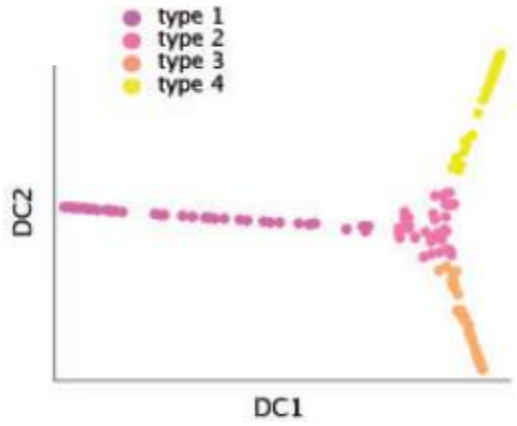


Denoising developmental scRNAseq data

PCA → *pairwise distances* → Distance matrix → *Knn graph* → Knn graph

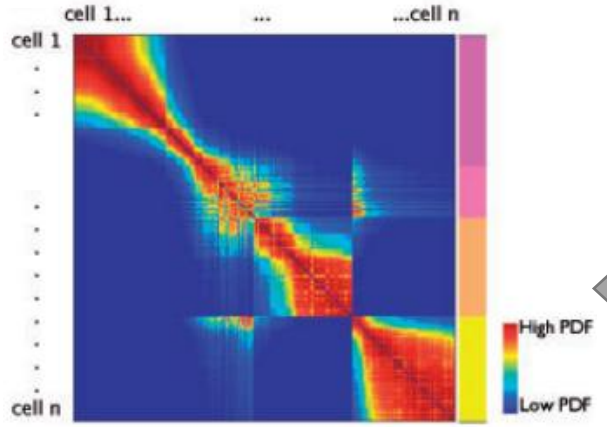


Haghverdi, L., Buettner, F. & Theis, F. J. Diffusion maps for high-dimensional single-cell analysis of differentiation data. *Bioinformatics* (2015) doi:10.1093/bioinformatics/btv325.



Diffusion maps

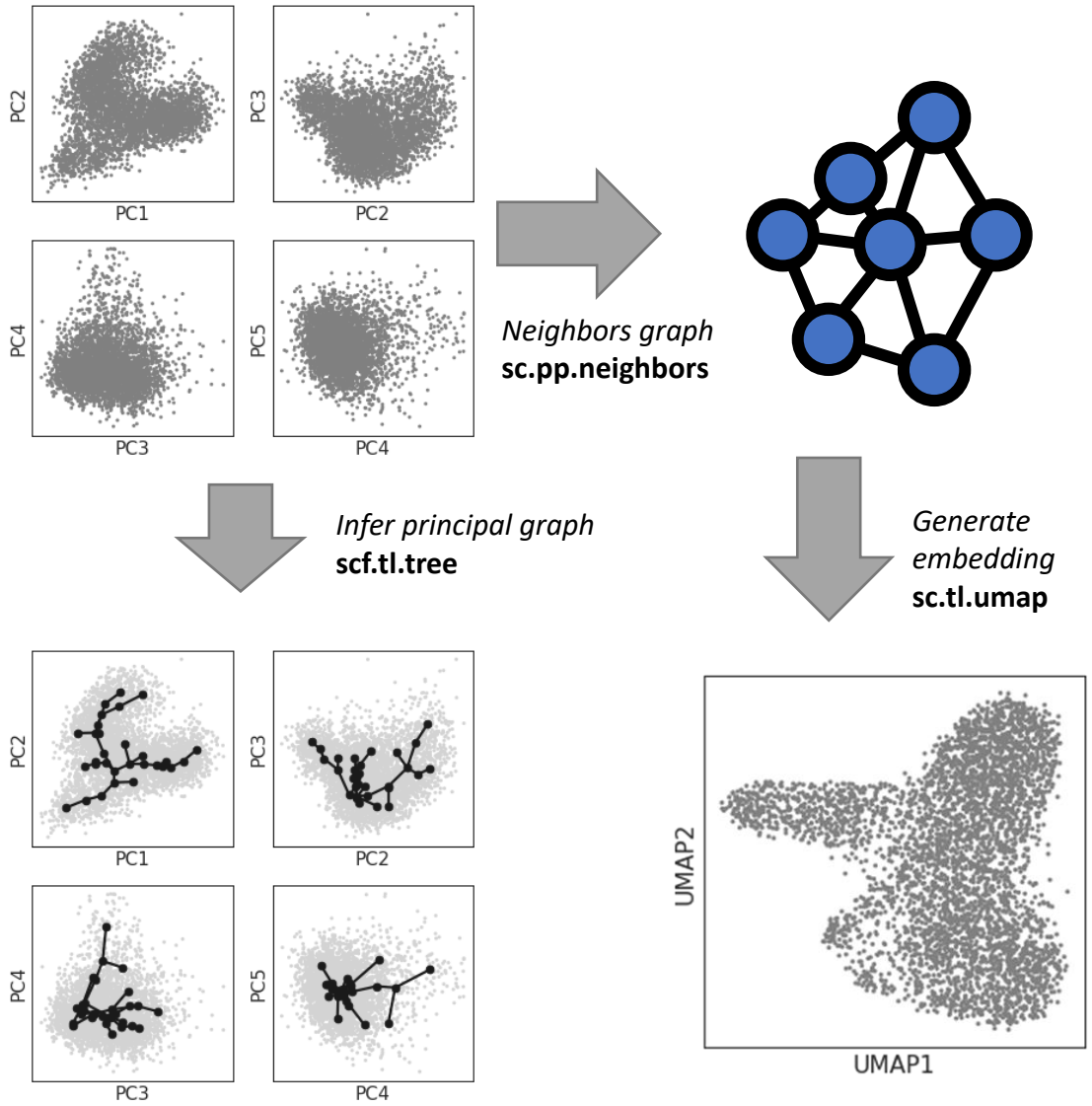
Eigen decomposition



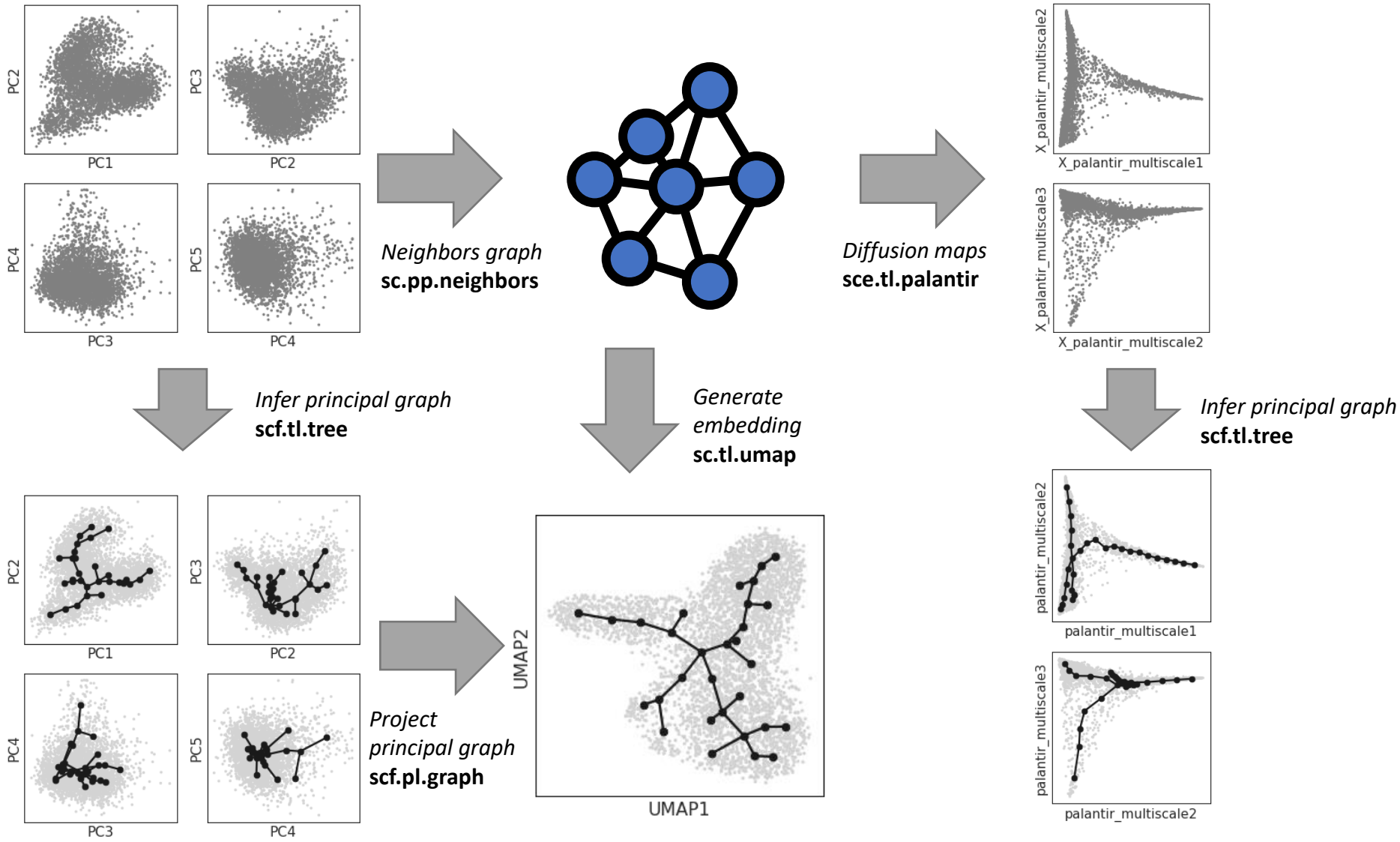
n × n Markovian transition probability matrix

Random walk following a time dependent diffusion process, constrained by a gaussian kernel

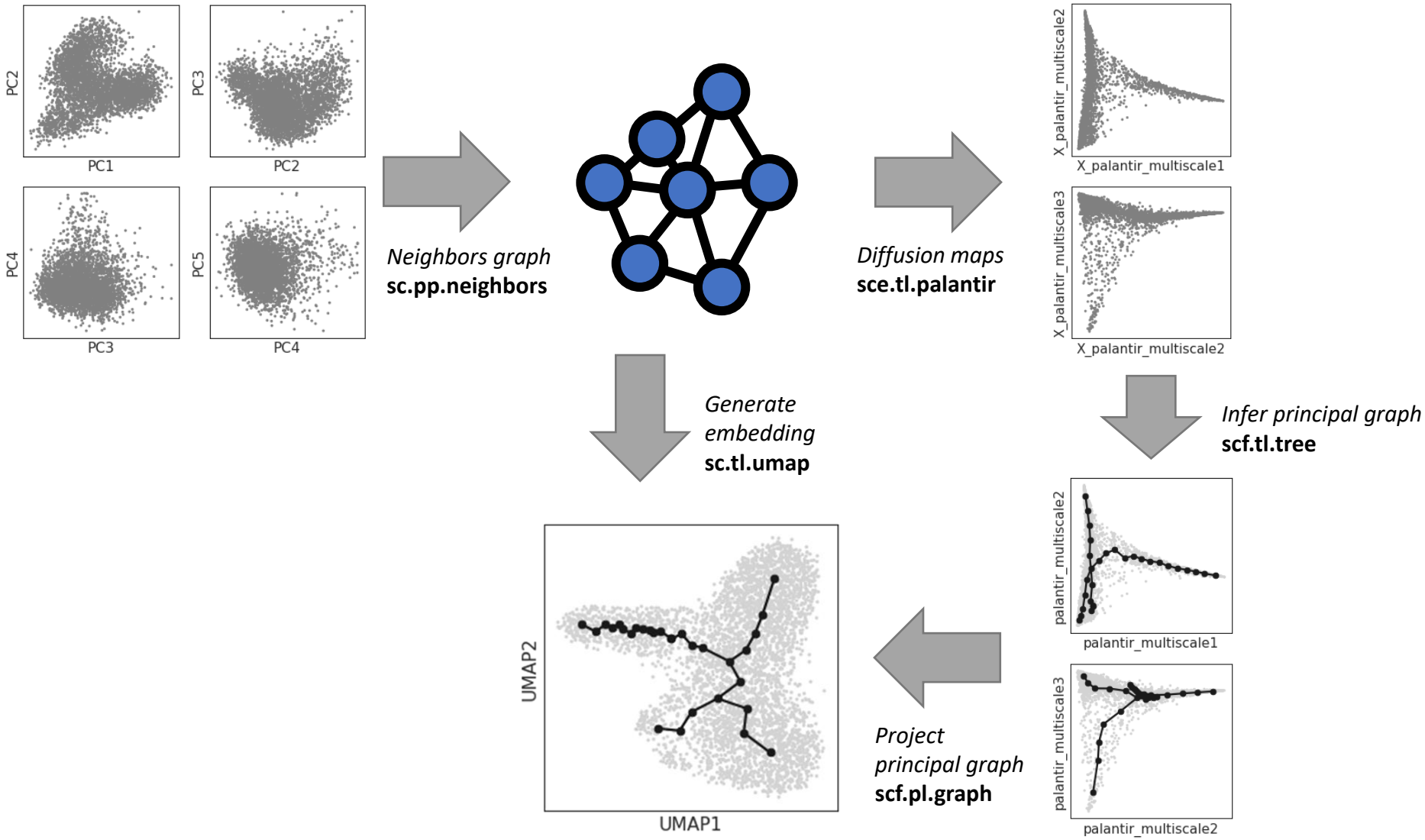
Trajectory inference in scRNAseq



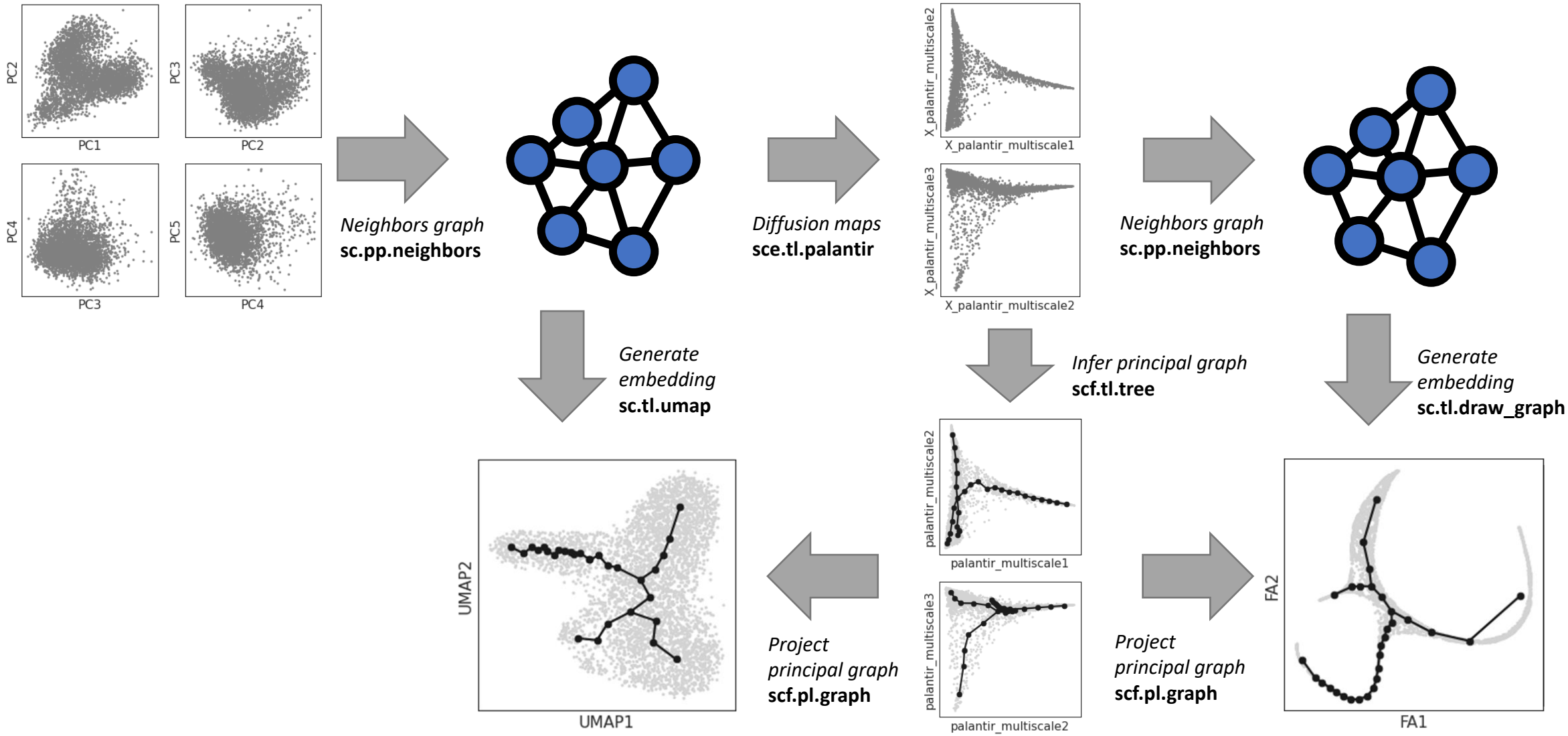
Trajectory inference in scRNAseq



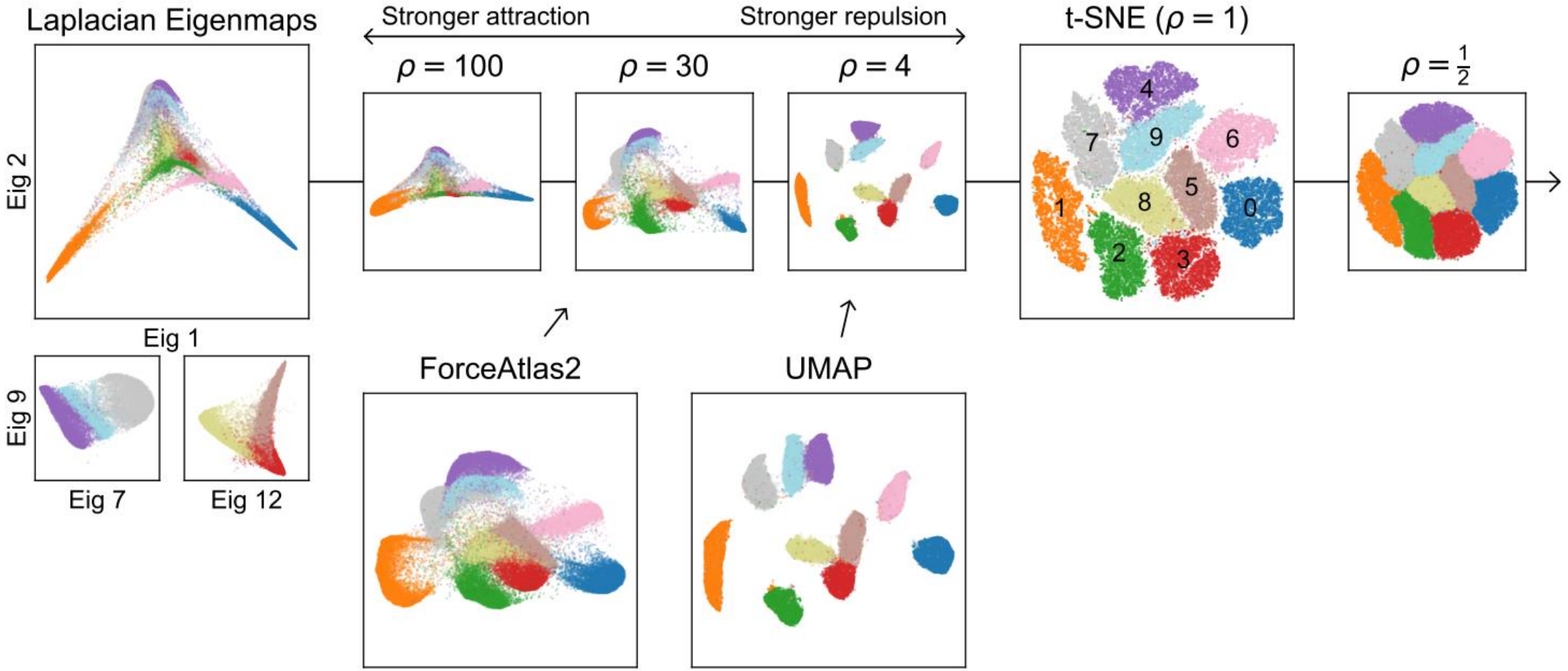
Trajectory inference in scRNAseq



Trajectory inference in scRNAseq

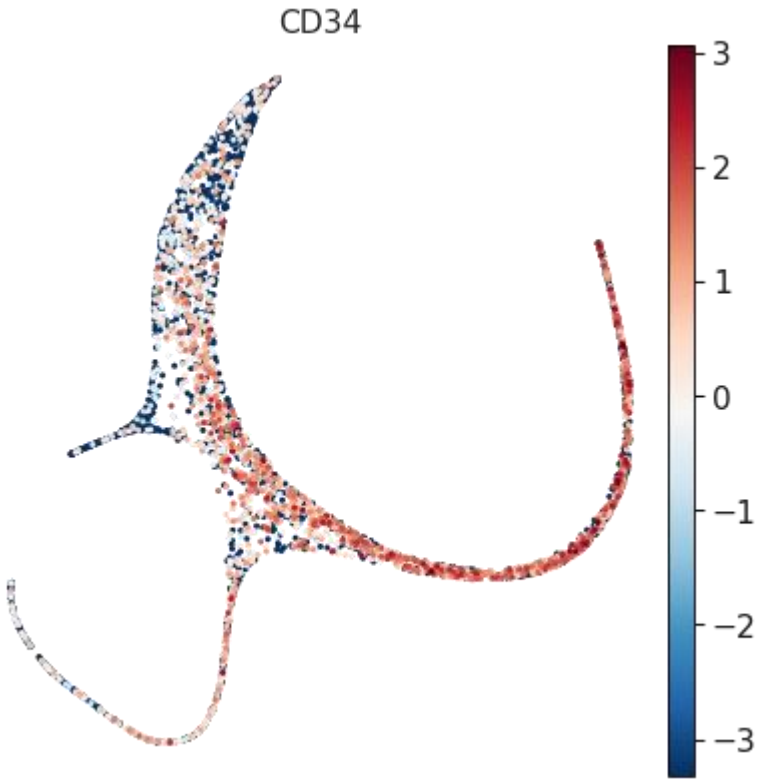


tSNE? UMAP? ForceAtlas2?



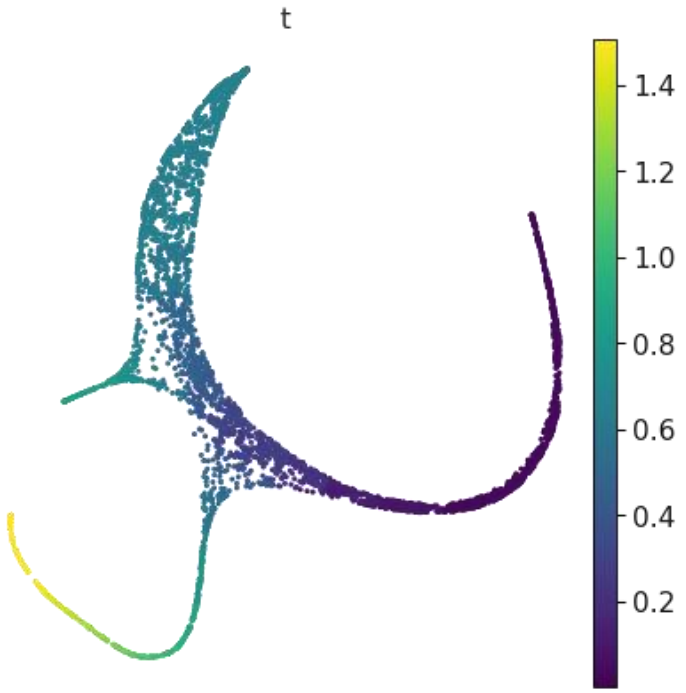
Must read!
https://arxiv.org/pdf/2007.08902.pdf

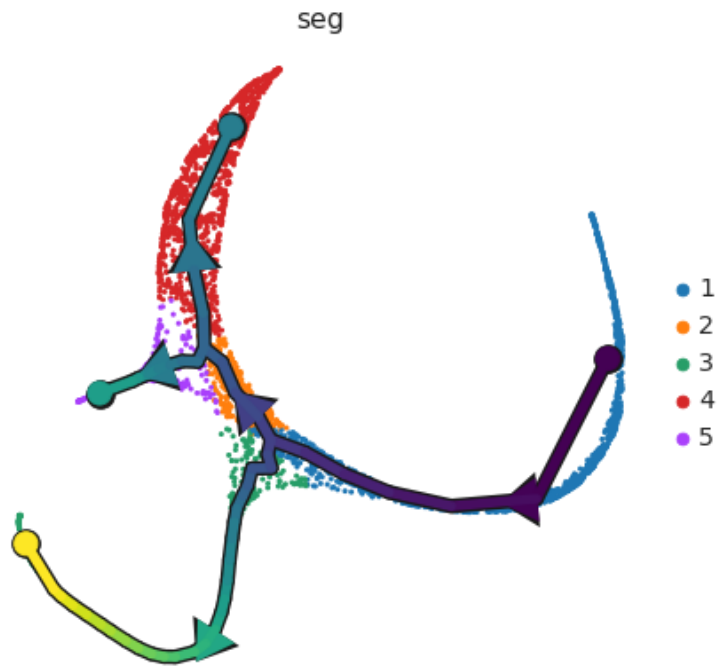
Selecting the root



Use the help from:

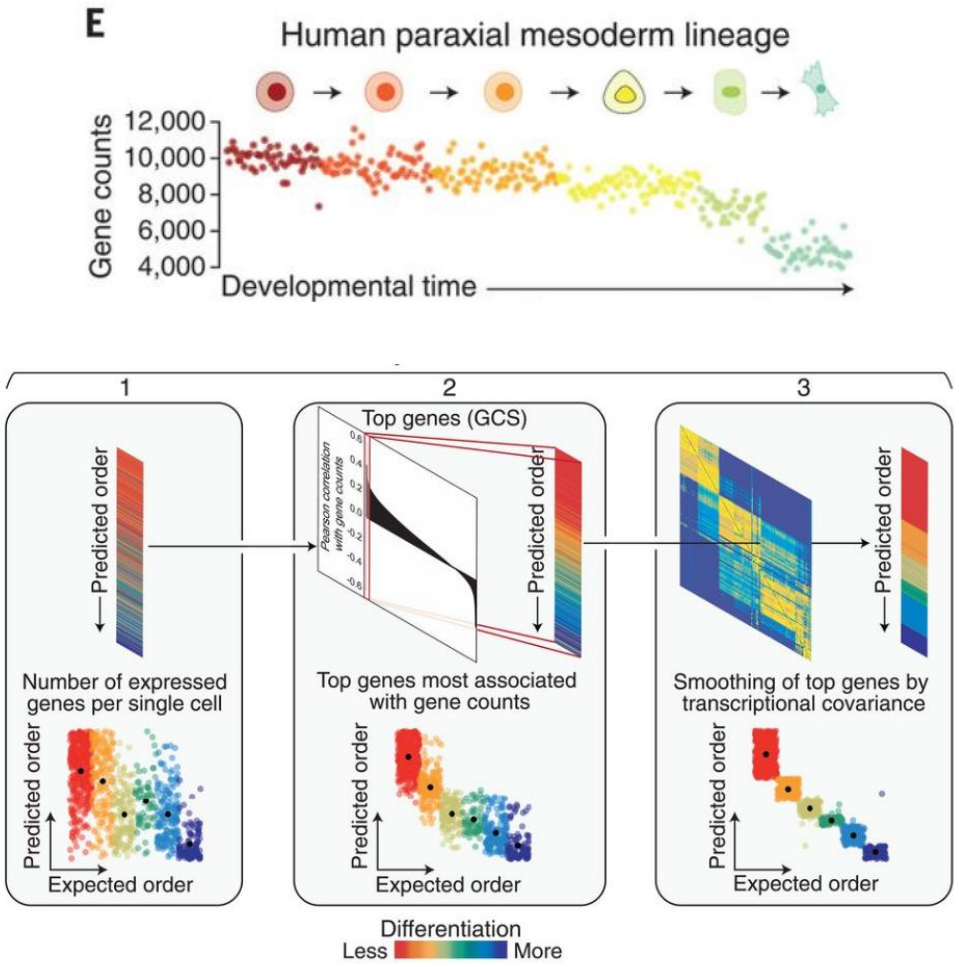
- Experimental knowledge (sampling timepoint)
- Biological knowledge (known progenitor markers)





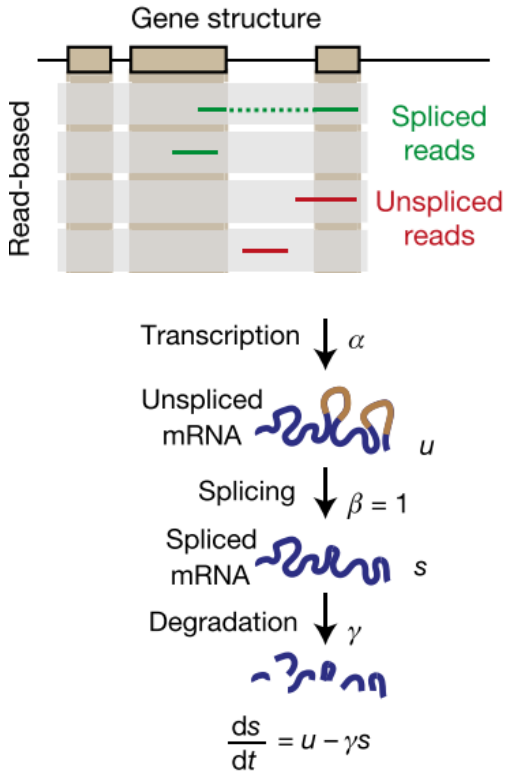
Need help finding the root?

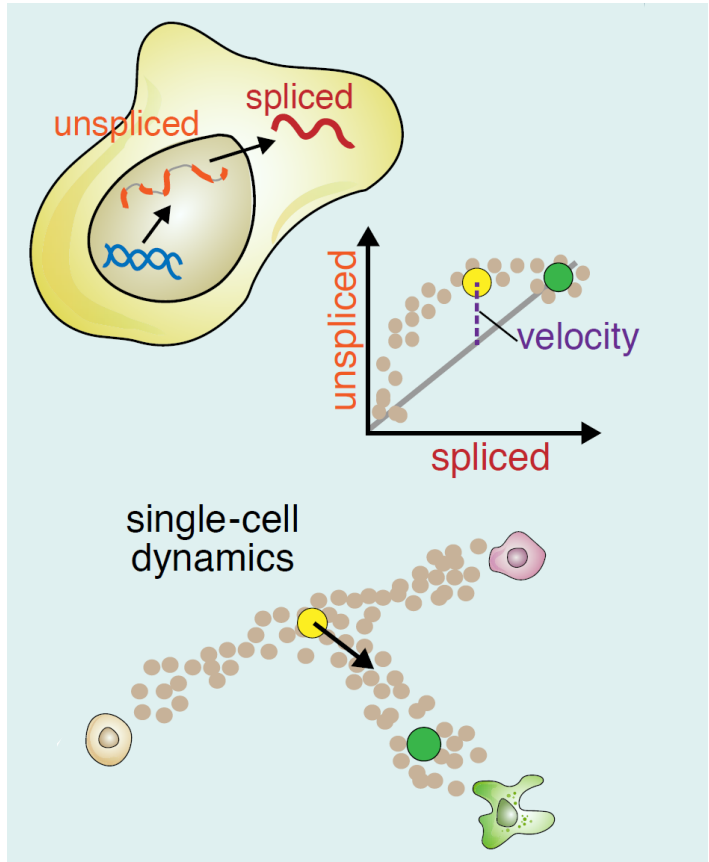
CytoTRACE



Gulati, G. S. *et al.* Single-cell transcriptional diversity is a hallmark of developmental potential. *Science (80-.).* (2020) doi:10.1126/science.aax0249.

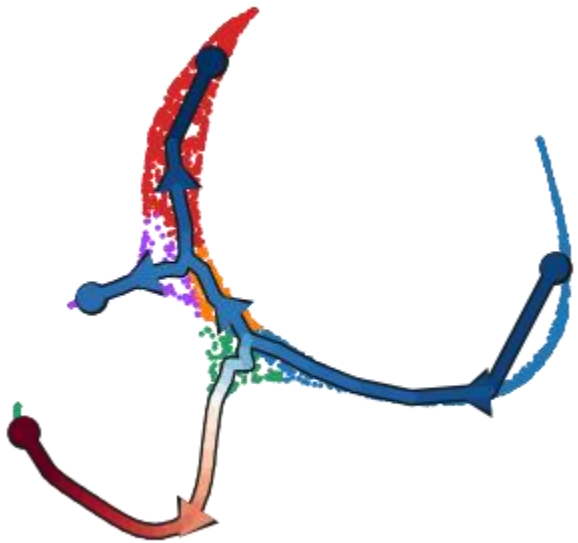
RNA Velocity

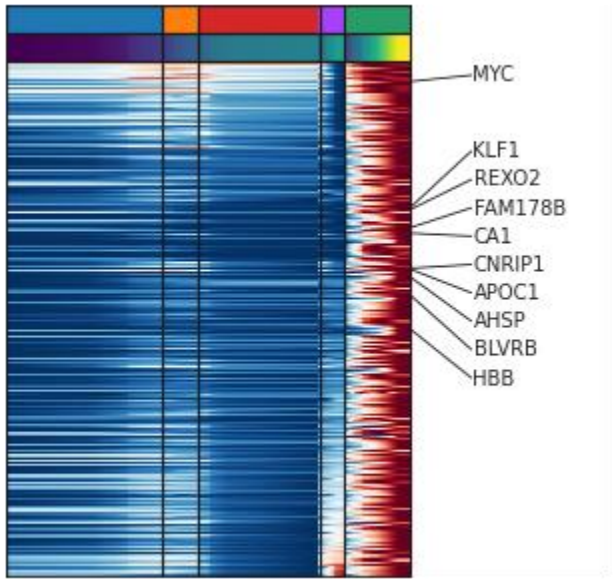


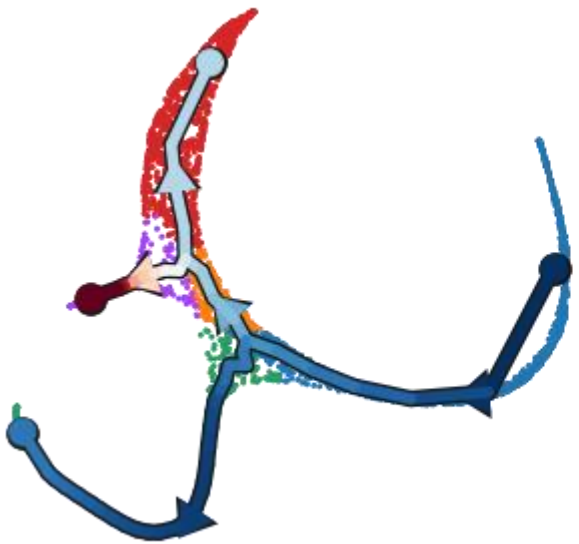


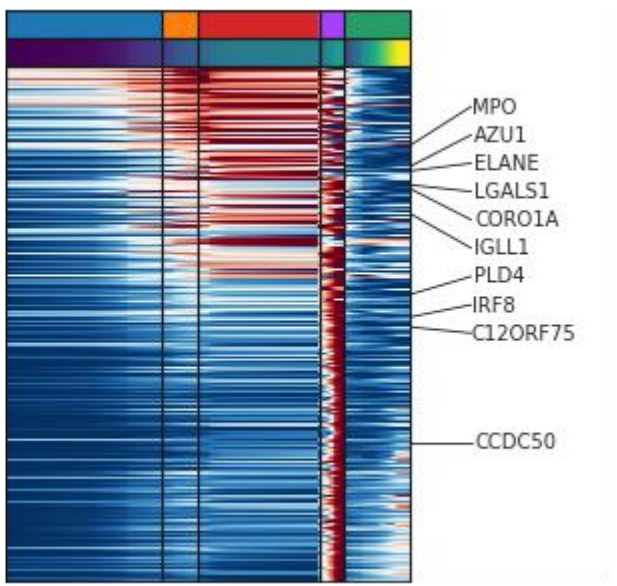
Edited from: Lederer, A. R. & La Manno, G. The emergence and promise of single-cell temporal-omics approaches. *Current Opinion in Biotechnology* (2020). doi:10.1016/j.copbio.2019.12.005

Recover the dynamics

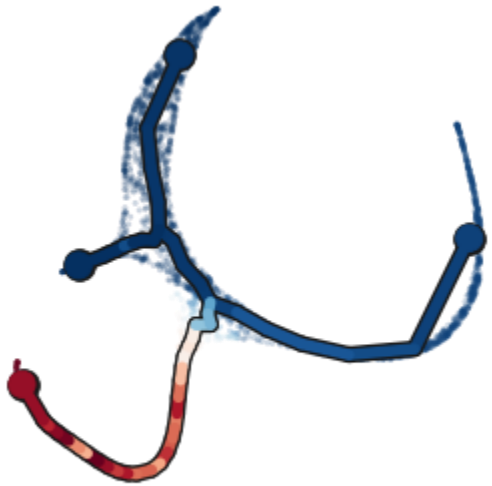


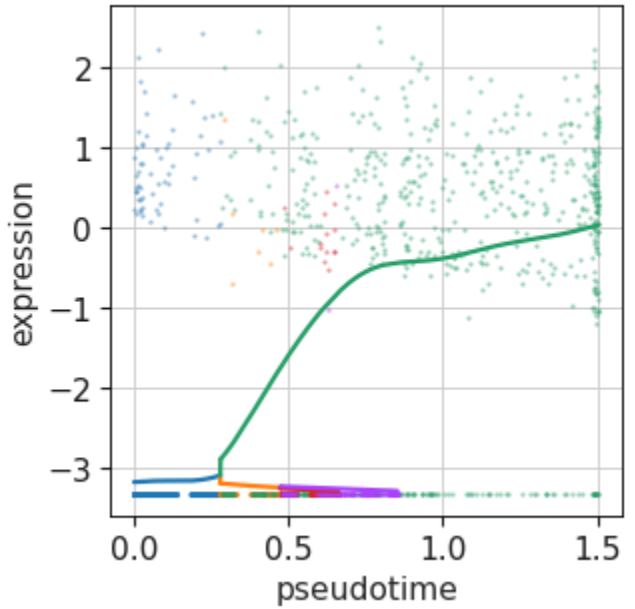




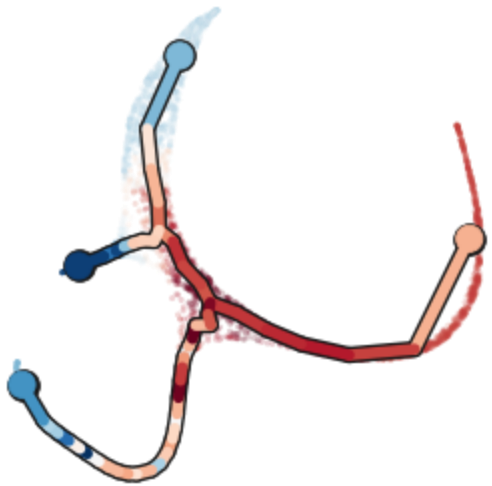


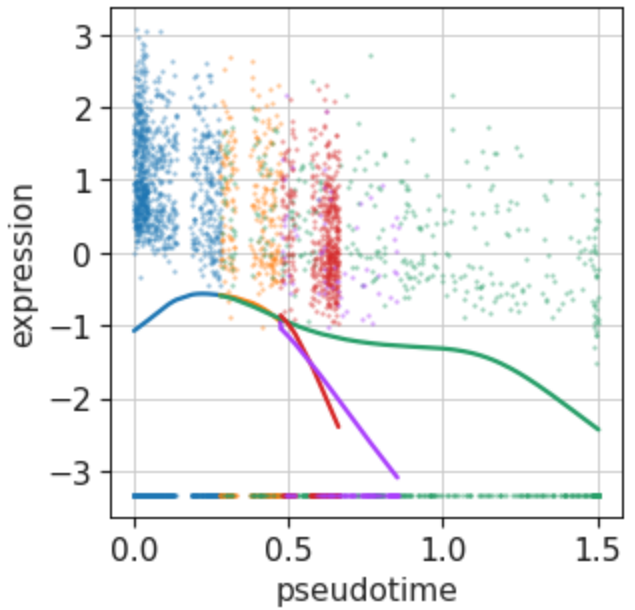
GATA1



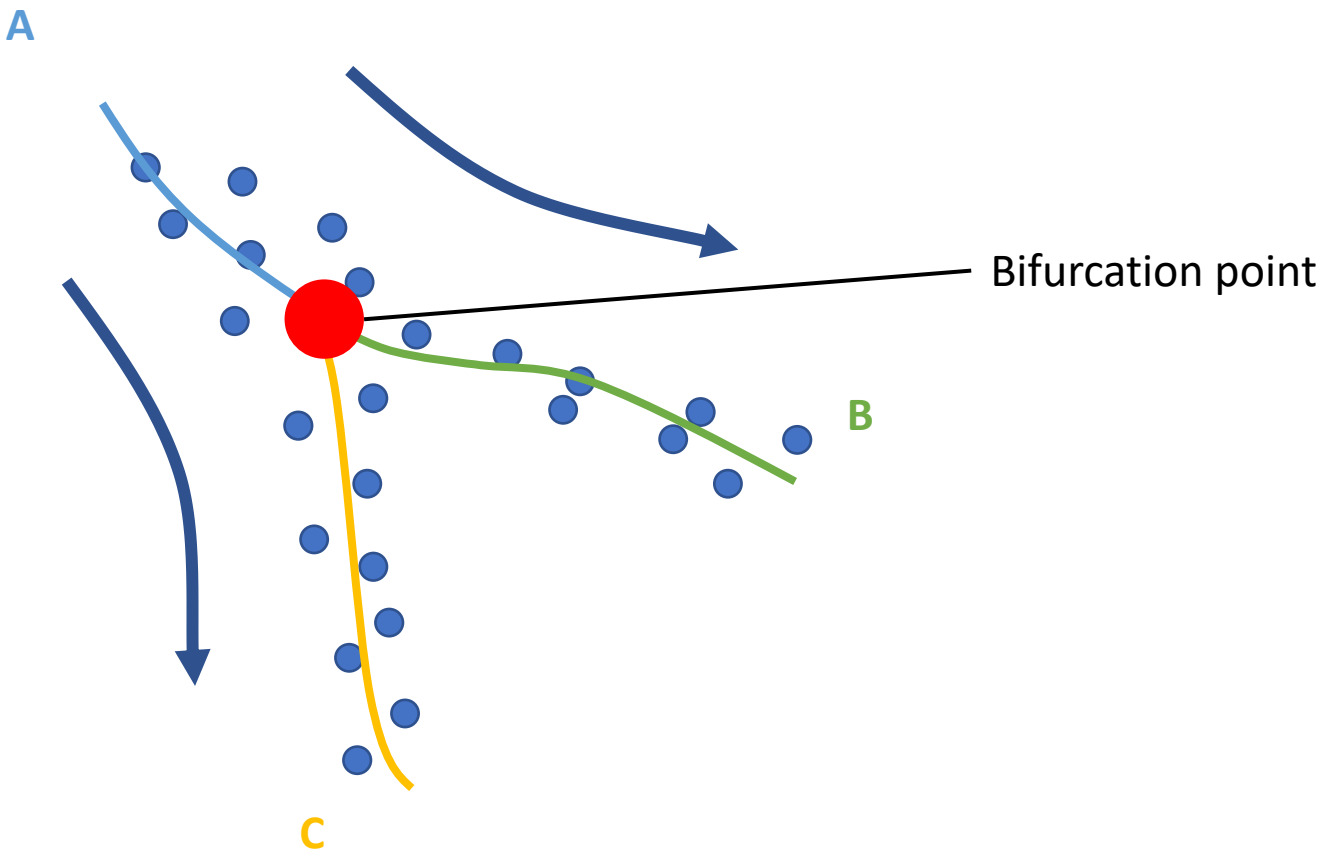


CD34

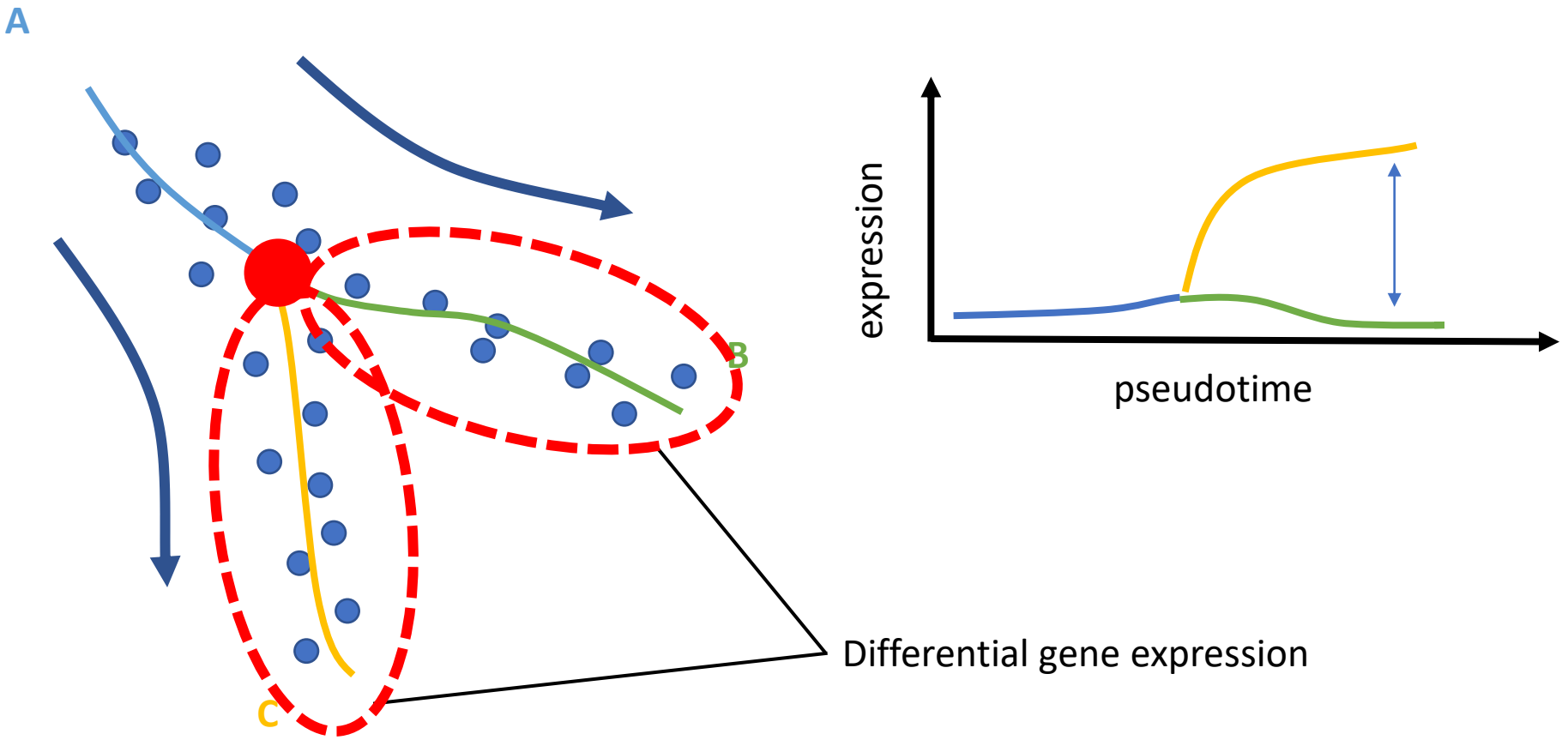




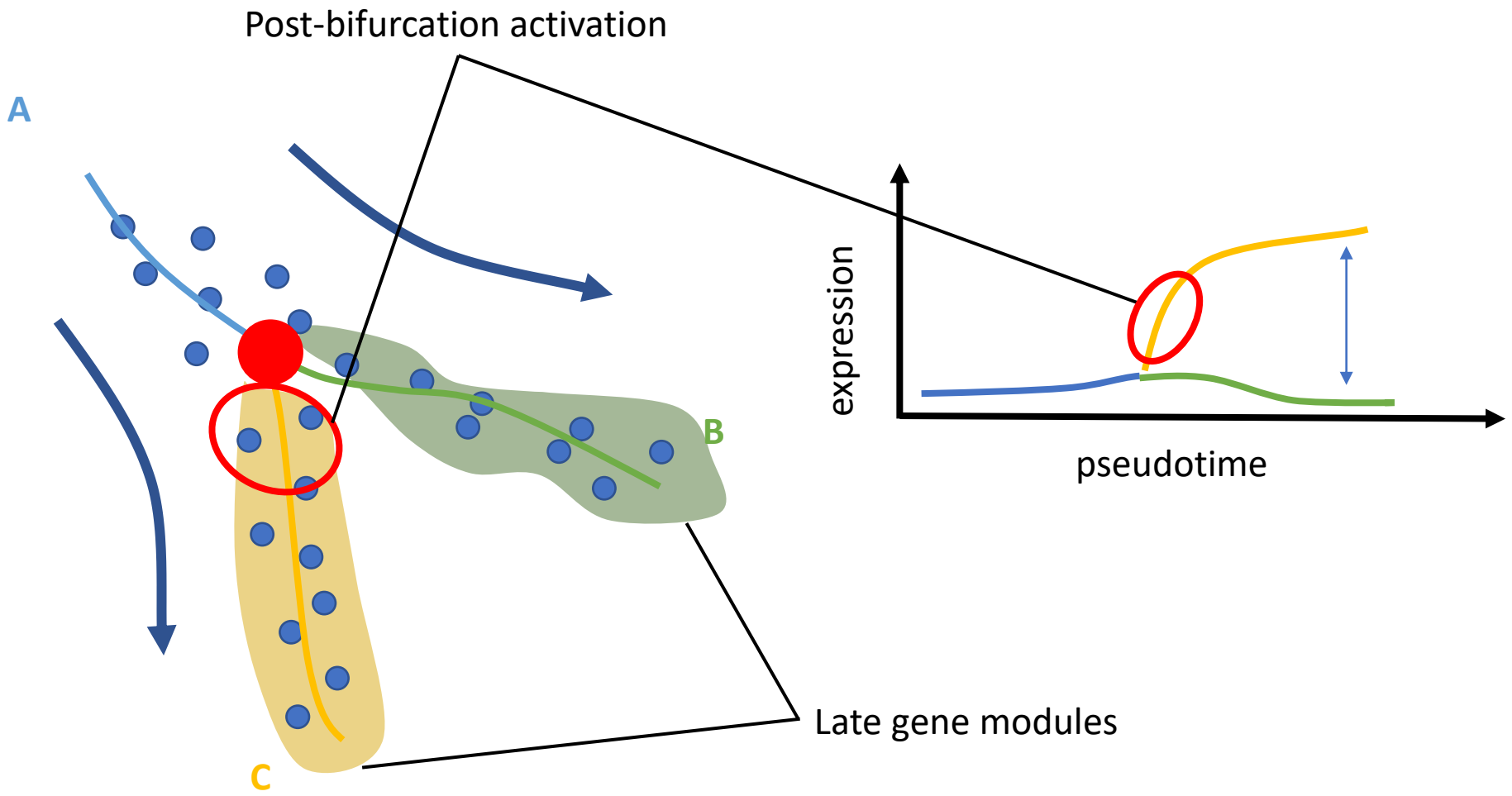
Probing bifurcations



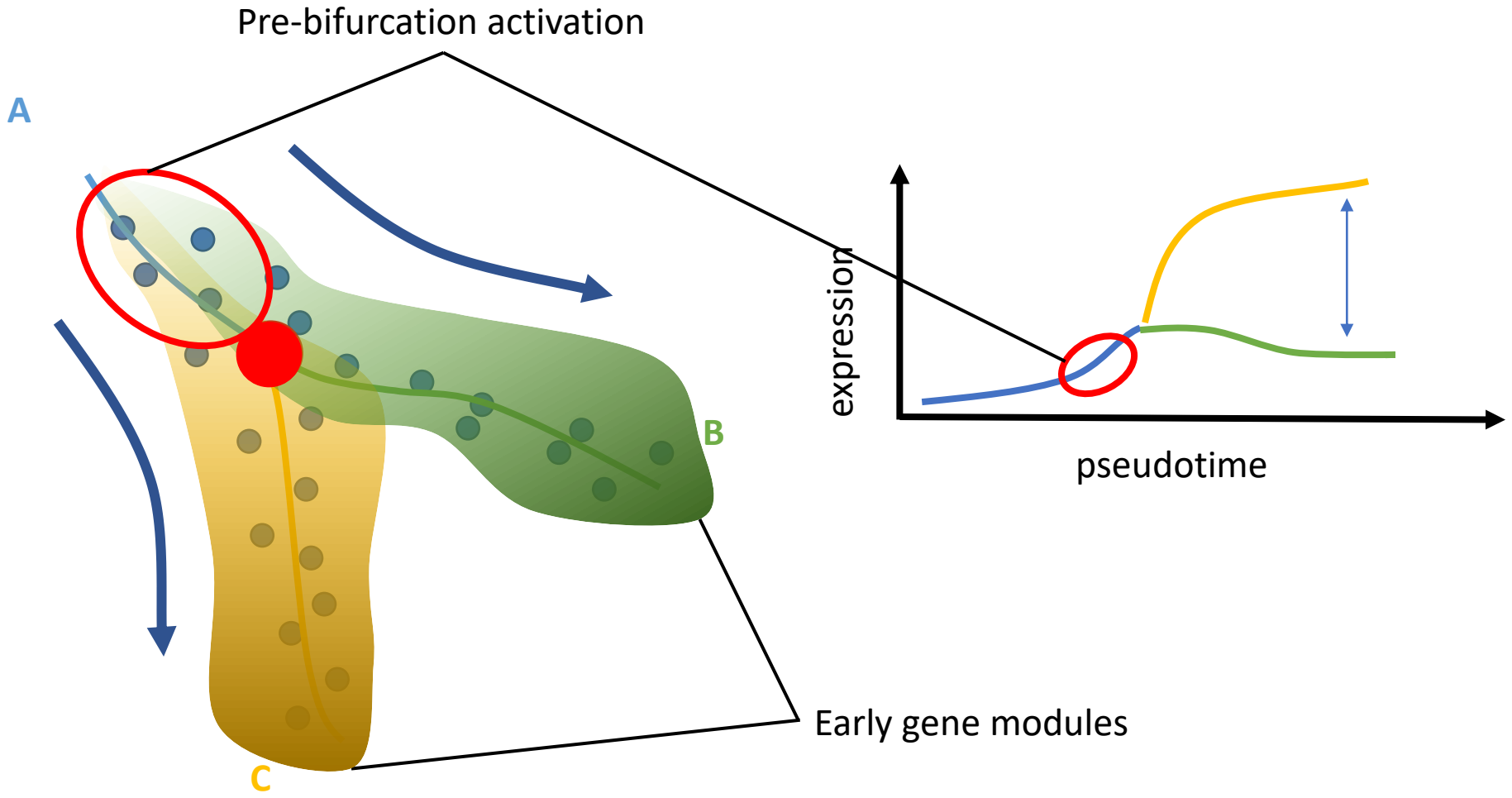
Probing bifurcations



Probing bifurcations

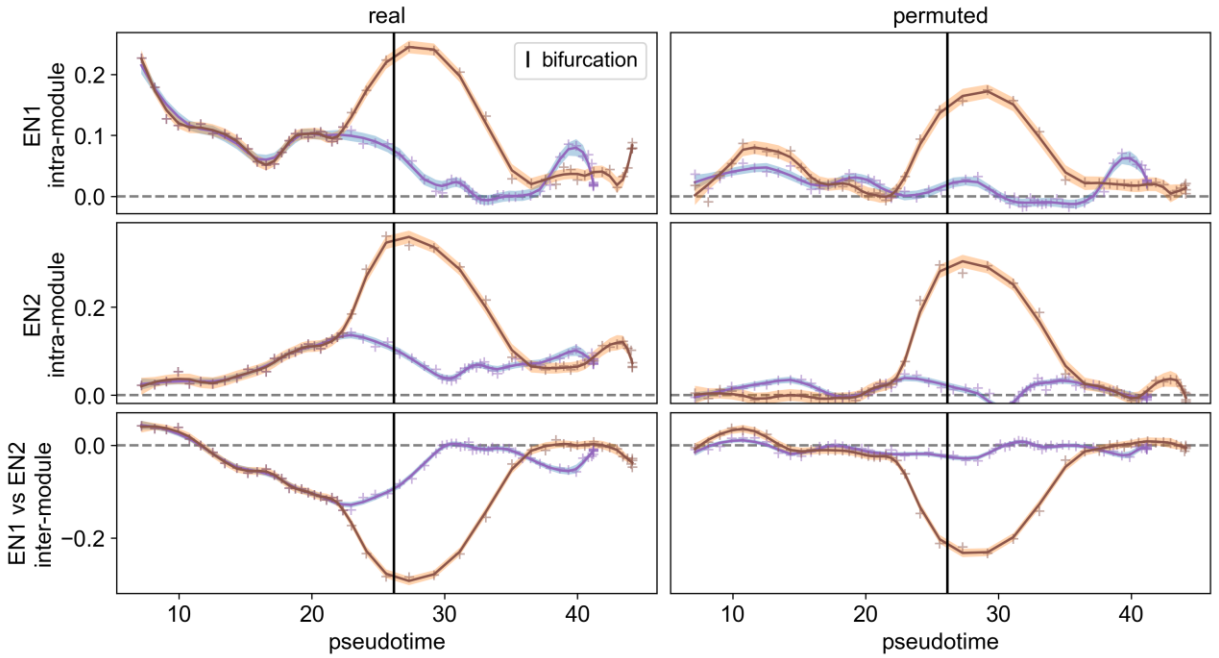


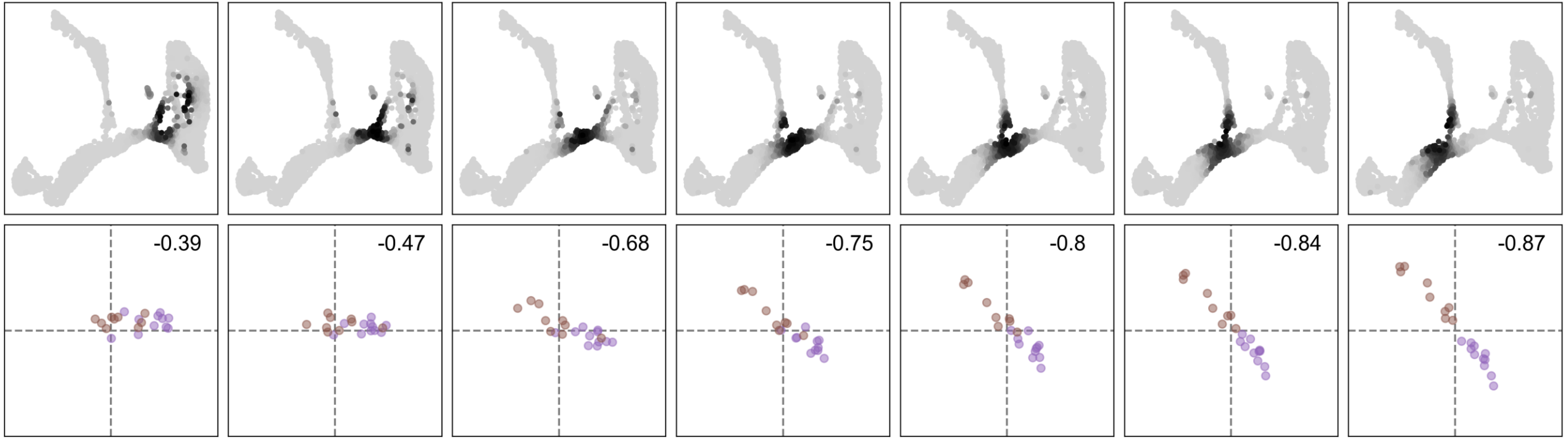
Probing bifurcations



Probing bifurcations

https://scfates.readthedocs.io/en/latest/notebooks/Advanced_bifurcation_analysis.html#Bifurcation-analysis





Thank you for listening!

scFates
Github repo: github.com/LouisFaure/scFates
Documentation: scfates.readthedocs.io
Package: pypi.org/project/scFates

Reproducibility: https://github.com/LouisFaure/Trajectory_Inference_workshop/

Soldatov *et al.* (2019), *Spatiotemporal structure of cell fate decisions in murine neural crest*, Science.