

The Krishnaswamy Laboratory
Yale Genetics and Yale SEAS present

Machine Learning for Single Cell Analysis

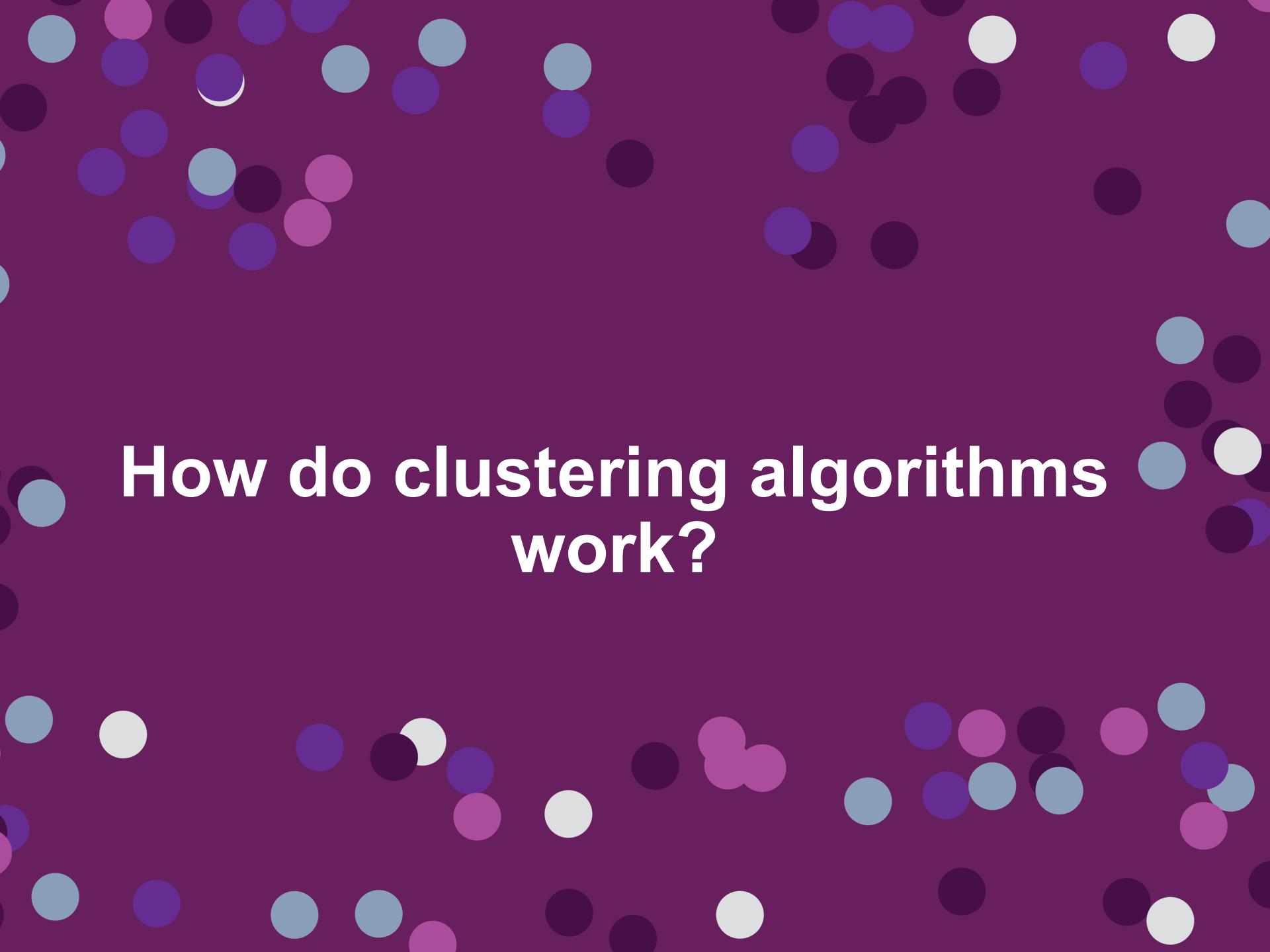
Online - May 20-29, 2020

When poll is active, respond at **PollEv.com/yaleml**

Text **YALEML** to **22333** once to join

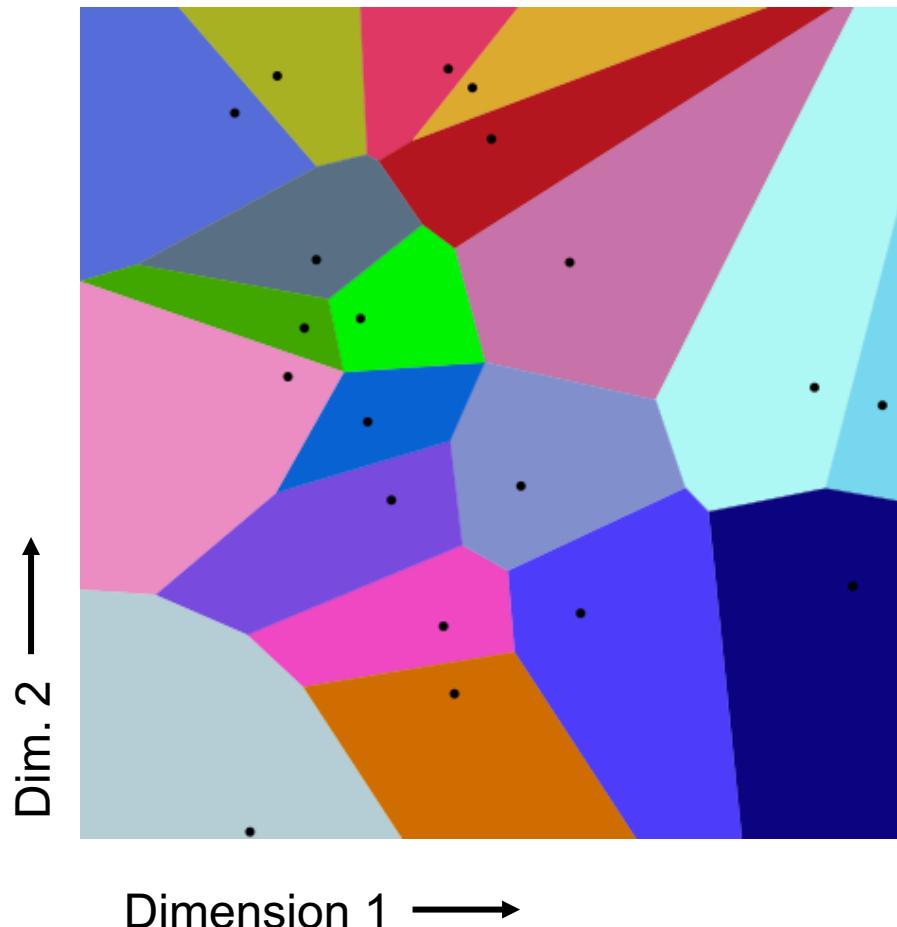
Where will you travel for your first trip post-quarantine? (use underscores for multi-word answers)

Learning Clusters, Trajectories, and Gene Relationships from scRNA-seq data



How do clustering algorithms work?

Clustering algorithms partition the data space



How is the partition picked?

- By minimizing different objective criteria
 - Closeness of members *within* the group
 - Distance/Separation between groups
 - Ratio of the two
 - Minimum cut of an NN-graph
- Other criteria?
- Modularity: actual edges/ expected edges

Mean Squared Error

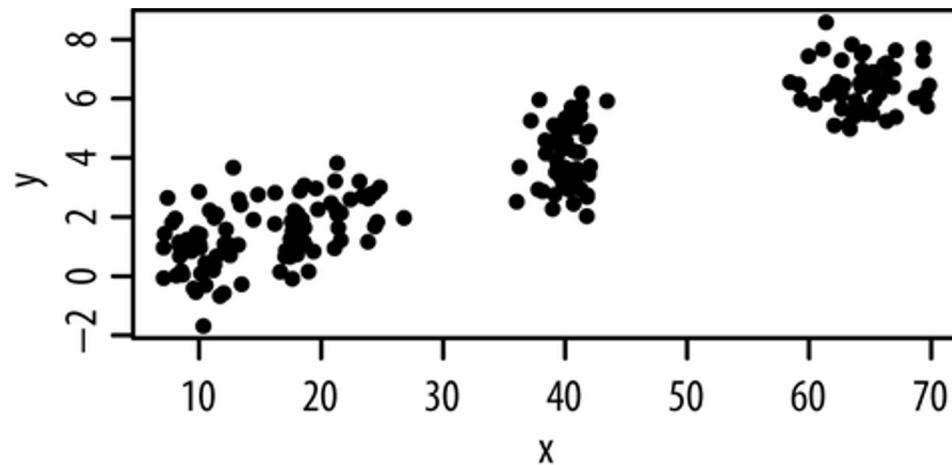
- Given data $X = \{x_1, x_2, \dots, x_n\}$
- Partition into k clusters
- Such that the within-cluster variance is minimized

$$\sum_{i=0}^n \min_{\mu_j \in C} (||x_i - \mu_j||^2)$$

“The sum of all squared euclidean distances between each point and that point’s closest cluster mean”

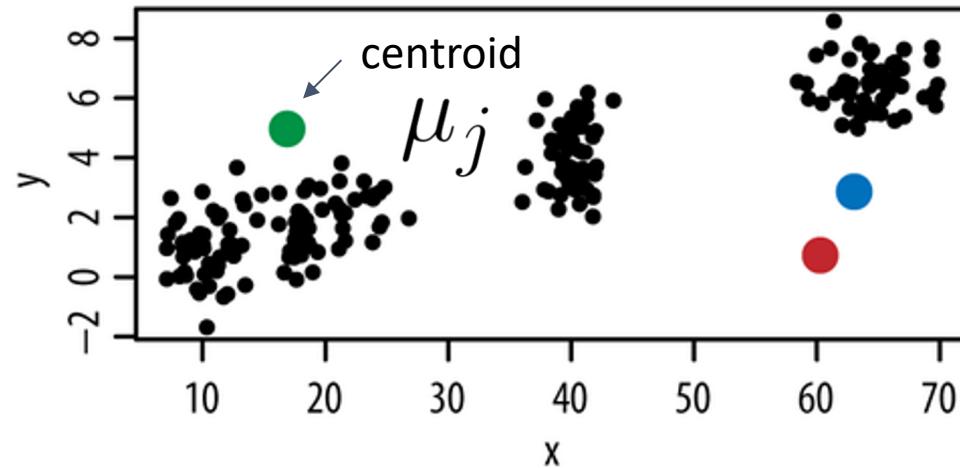
k-Means minimizes within-cluster sum-of-squares

Unlabelled data

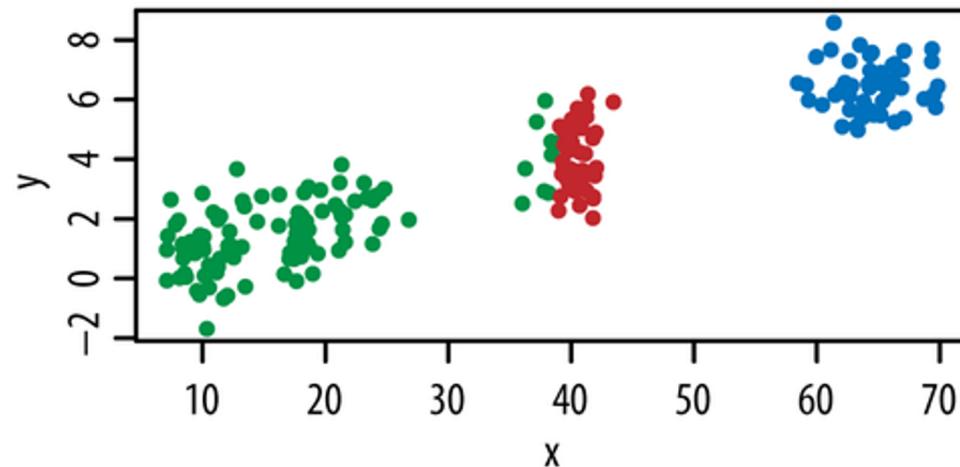


k-Means minimizes within-cluster sum-of-squares

1. Random initialization



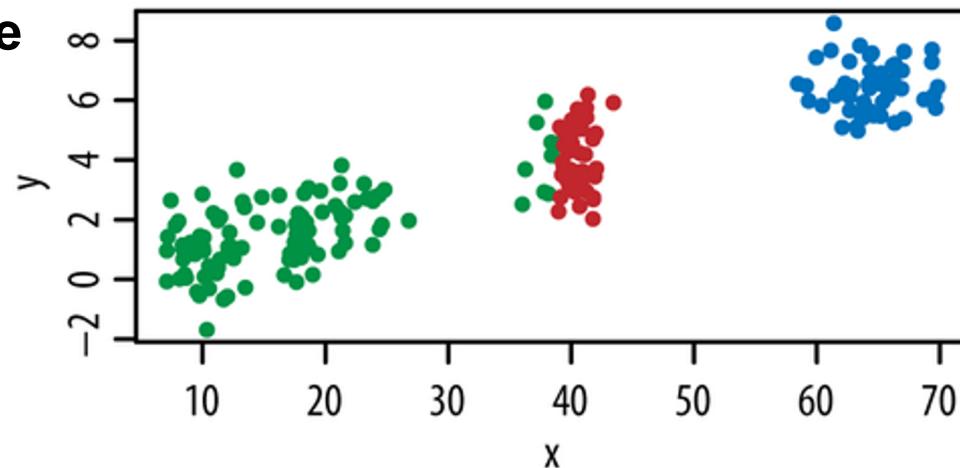
2. Assign points to nearest centroid



k-Means minimizes within-cluster sum-of-squares

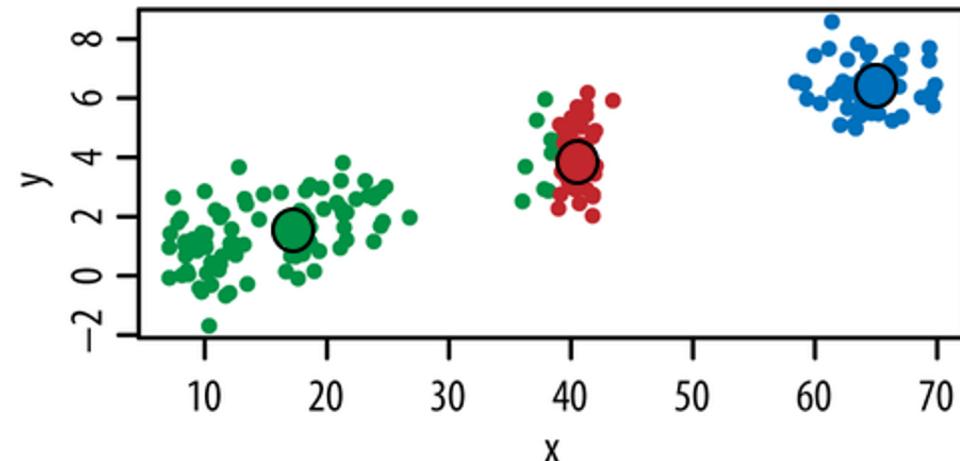
Repeat until convergence

2. Assign points to nearest centroid



3. Move centroids to middle of cluster

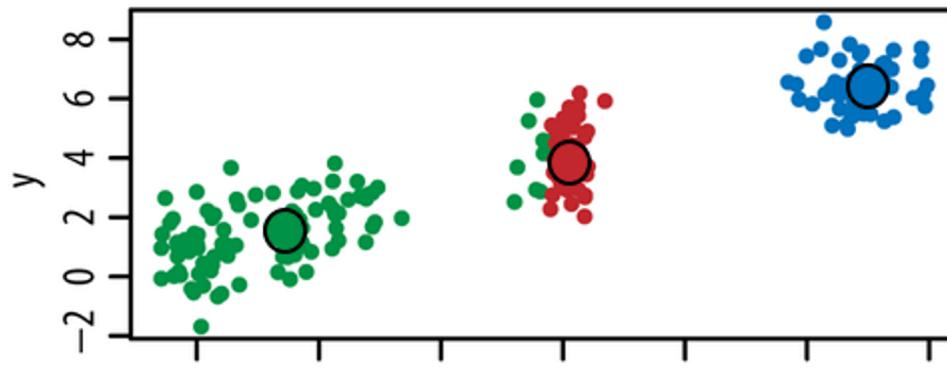
$$\mu_i = \frac{1}{|S_i|} \sum_{x_j \in S_i} x_j$$



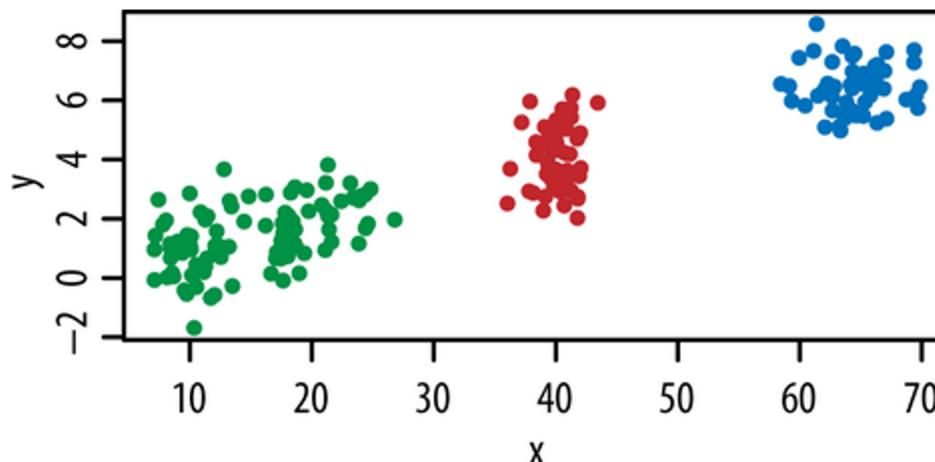
k-Means minimizes within-cluster sum-of-squares

Repeat until convergence

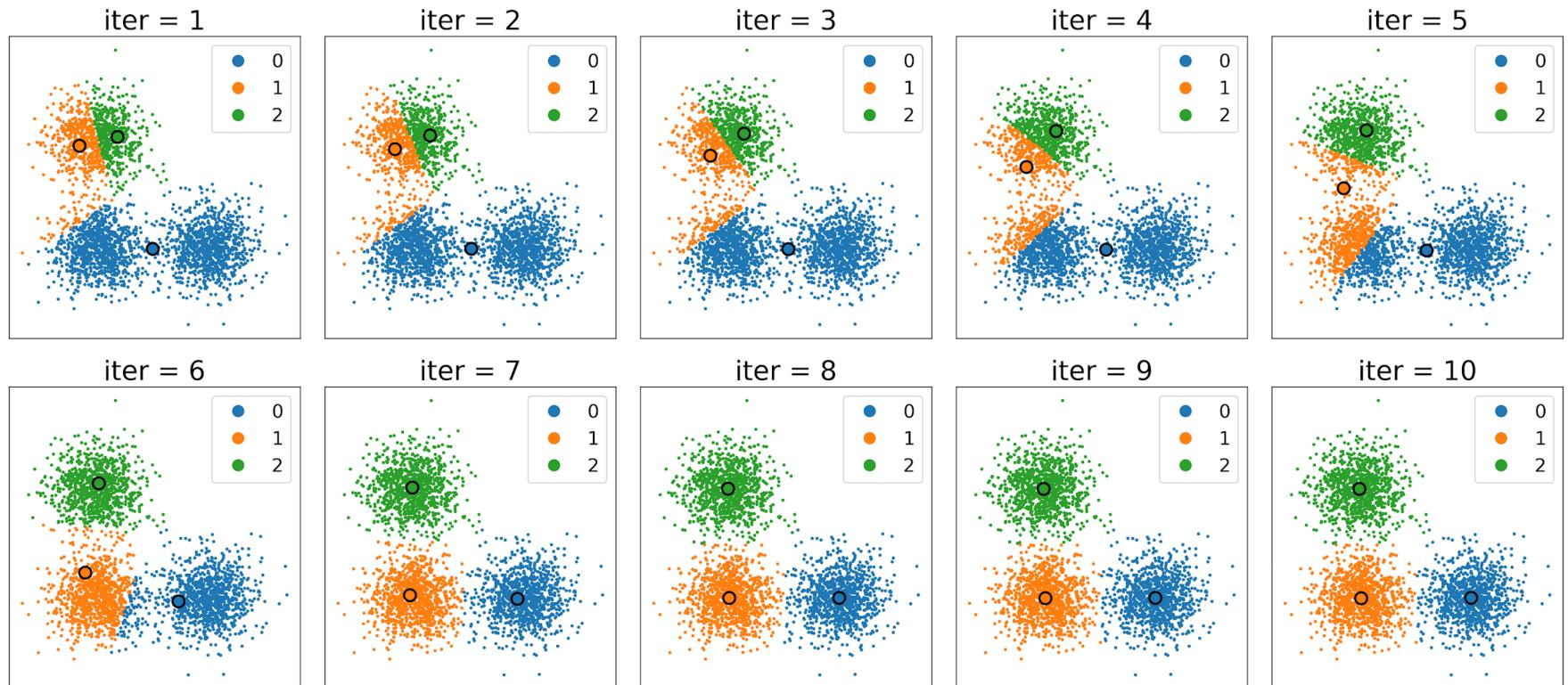
3. Move centroids to middle of cluster



4. Assign points to nearest centroid

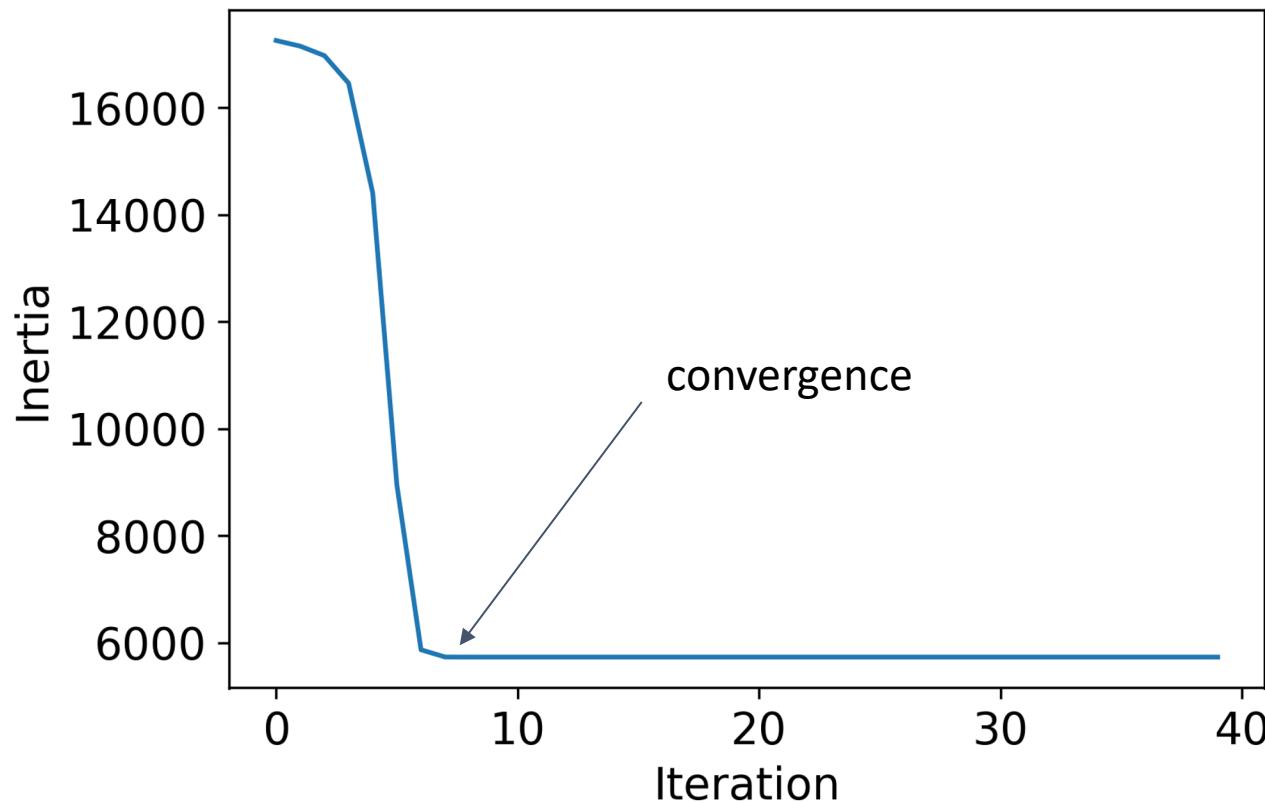


10 iterations of KMeans on Gaussians



What does convergence look like?

$$\text{Inertia} = \text{within-cluster sum-of-squares} = \sum_{i=0}^n \min_{\mu_j \in C} (\|x_i - \mu_j\|^2)$$



What does it converge to?

- Generally a local minima.
- Sensitive to initial conditions
- Methods for initialization:
 - Forgy: K observations chosen as means
 - Random Partition: Randomly partitions data

Do you think KMeans interations from a given initialization will converge at a single solution?

Yes, the algorithm will always arrive at the same solution given the same initialization

No, the algorithm can produce different solutions given the same initialization

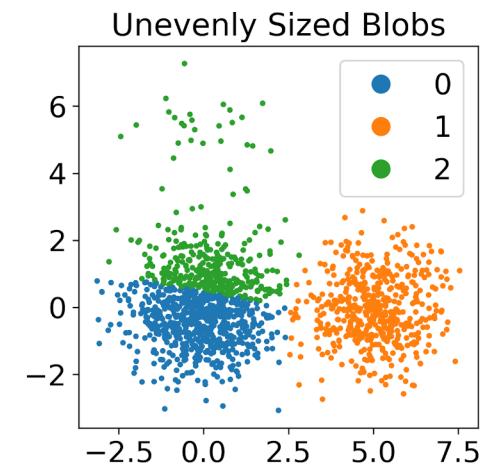
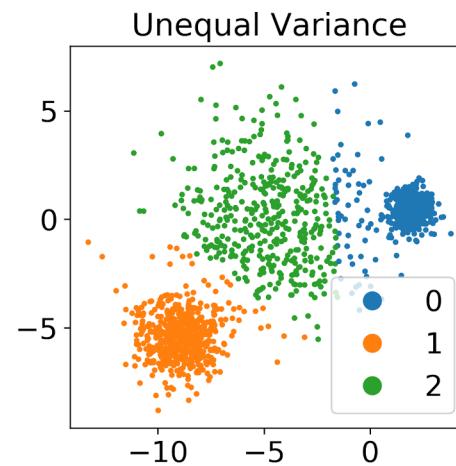
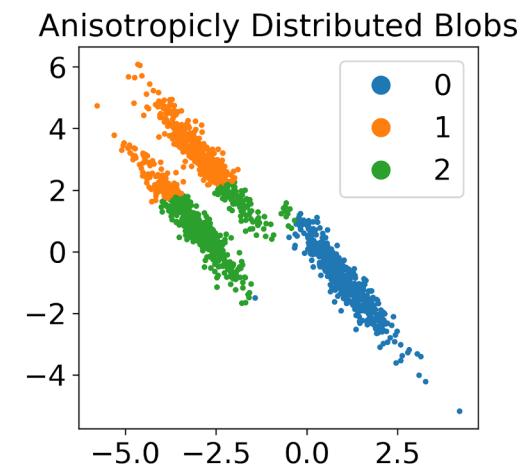
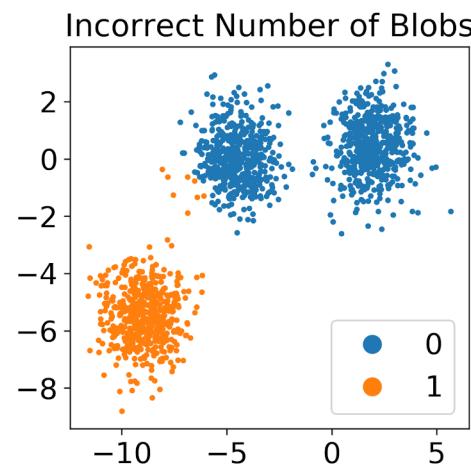
“K-means clustering is not a free lunch”

k-Means assumes:

1. K is chosen correctly
2. The data is distributed normally around the mean
3. All clusters have equal variance
4. All clusters have the same number of points

K-Means is sensitive to:

1. Initialization
2. Outliers



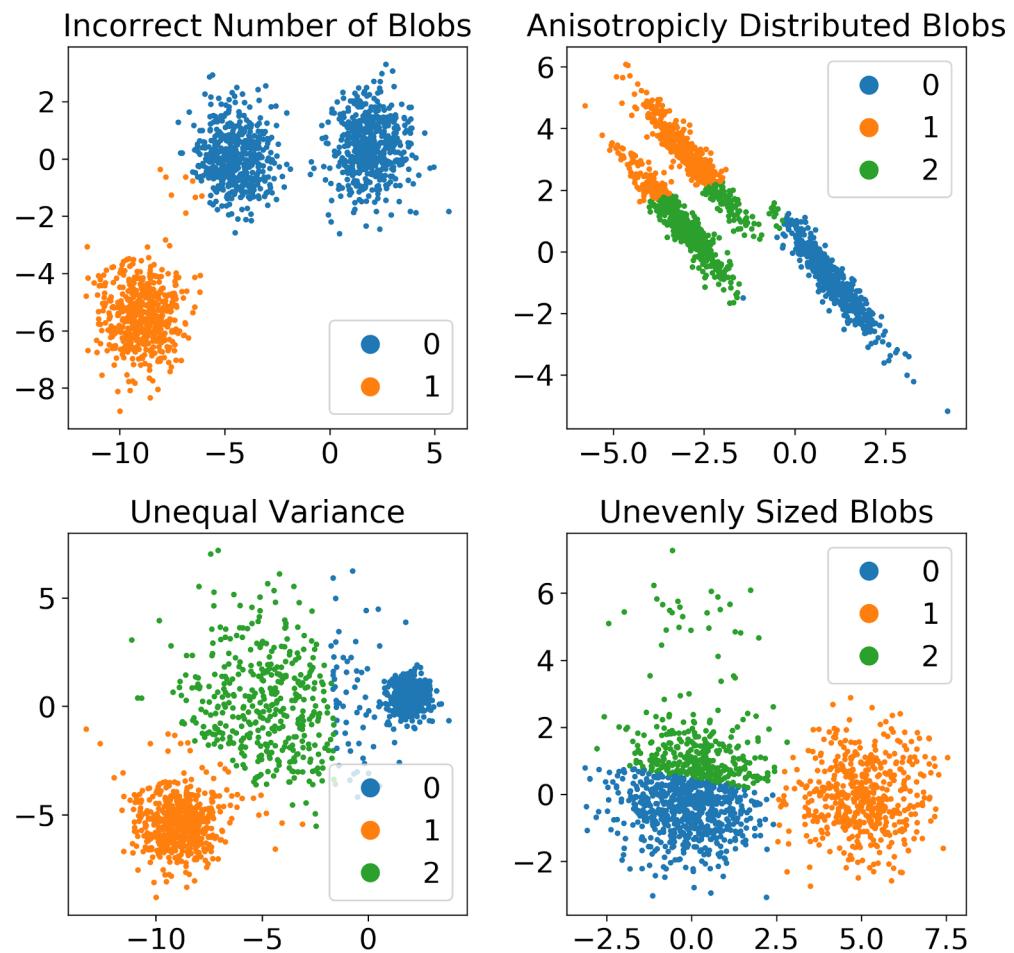
Limitations of this method

- Assumes a shape for the cluster: spectral clustering can help with this
- Must give number of clusters: there are methods to choose the number of clusters
- Sensitive to outliers: k-medoids helps with this
- Finds a local minima: repeat with different initializations

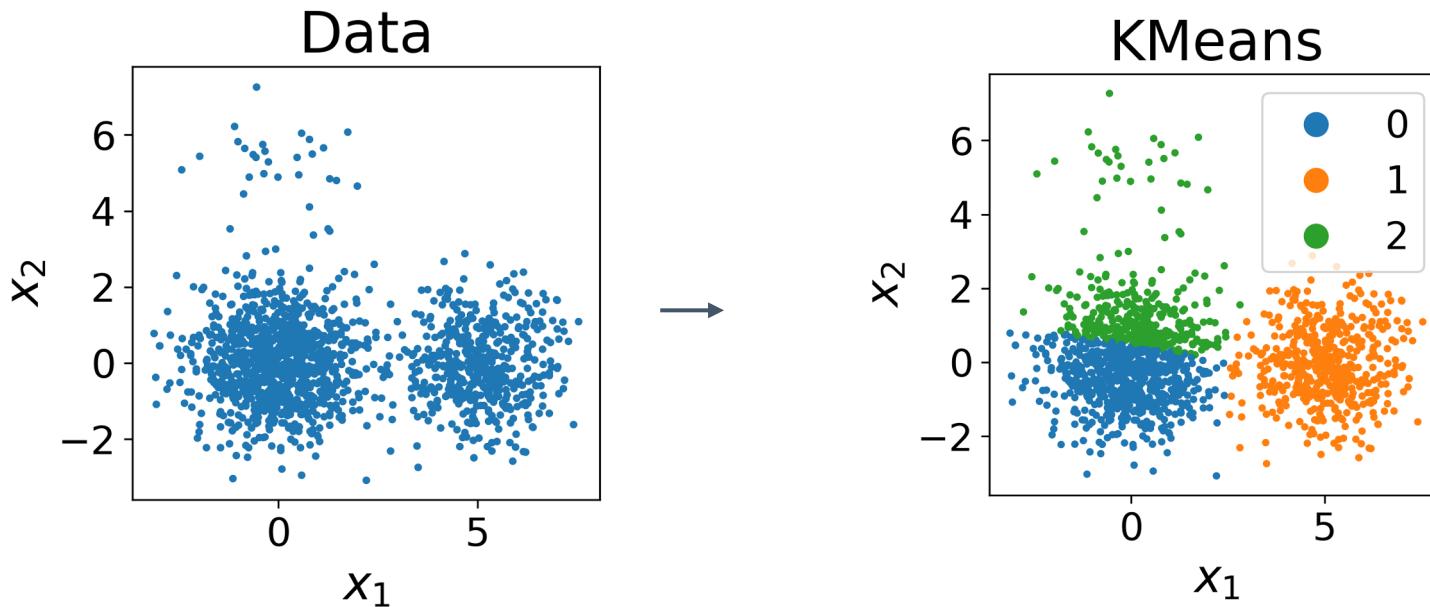
“K-means clustering is not a free lunch”

k-Means assumes:

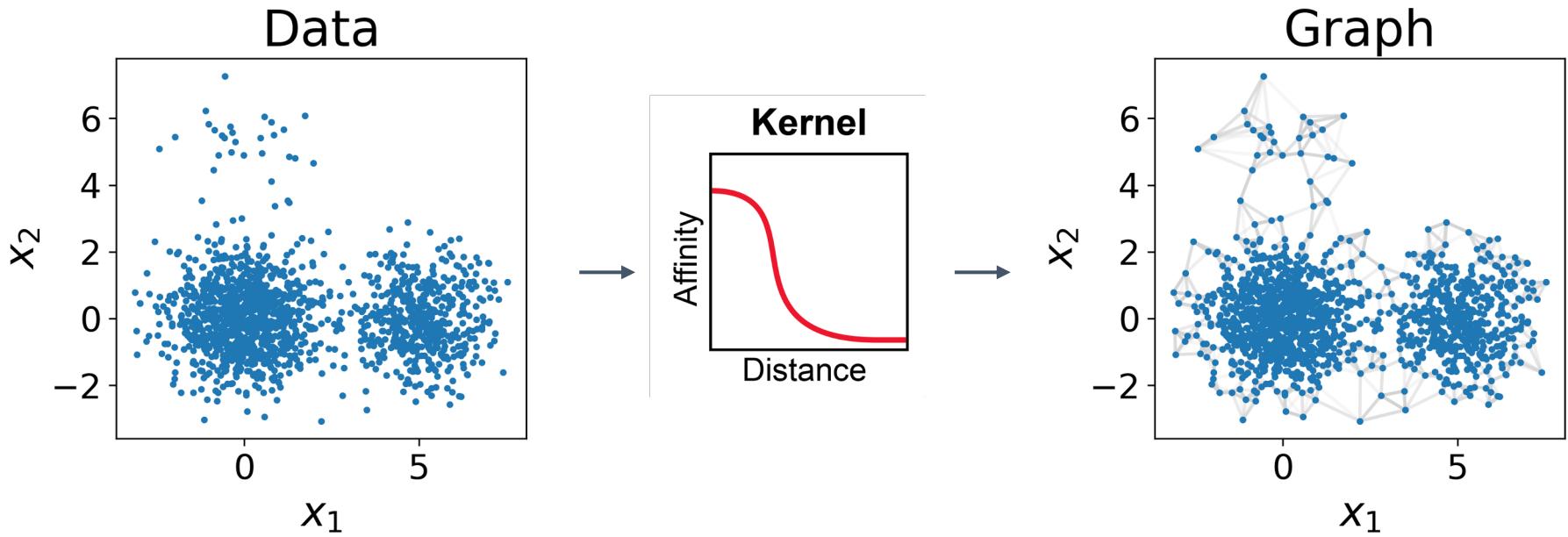
1. K is chosen correctly
2. The data is distributed normally around the mean
3. All clusters have equal variance
4. All clusters have the same number of points



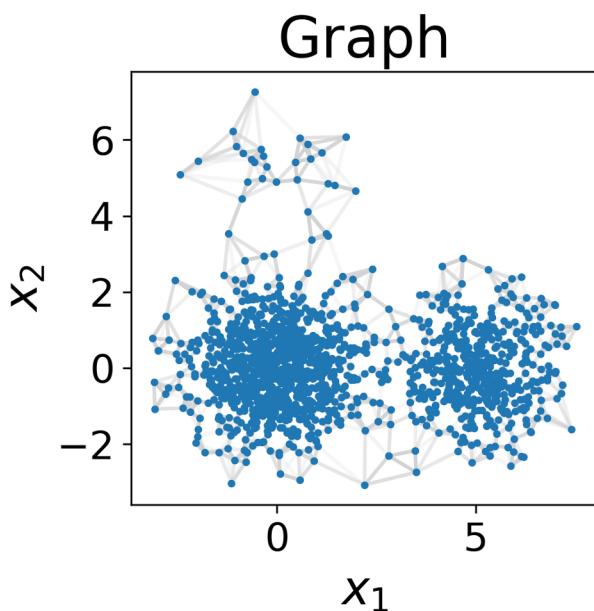
Spectral clustering uses eigenvectors of the graph Laplacian for clustering



Spectral clustering uses eigenvectors of the graph Laplacian for clustering



Spectral clustering uses eigenvectors of the graph Laplacian for clustering



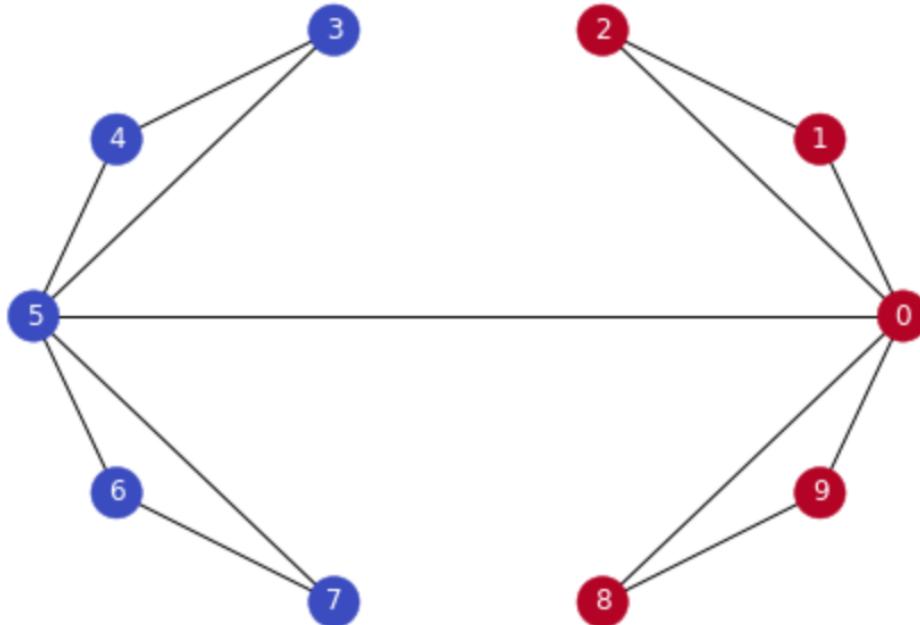
\mathbf{A} = adjacency matrix

$\mathbf{A}_{i,j}$ = edge weight between nodes
and i j

\mathbf{D} = degree matrix

\mathcal{L} = Laplacian matrix = $\mathbf{D} - \mathbf{A}$

Fiedler Vector

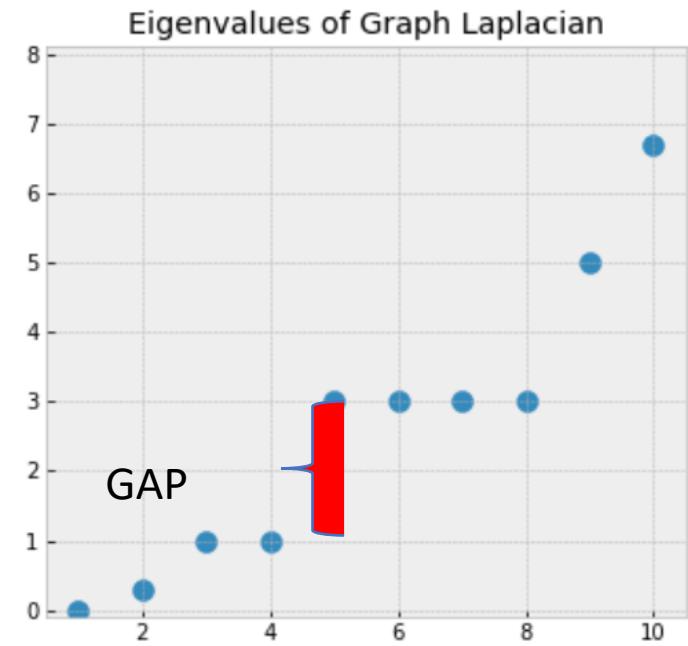
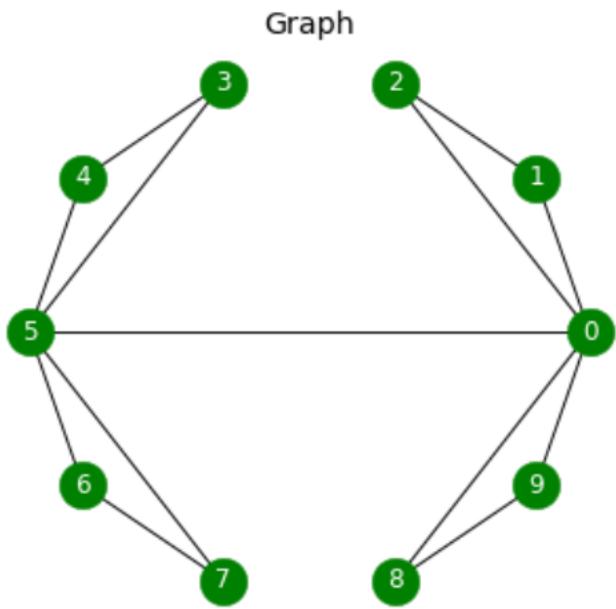


Second-smallest Eigenvalue is called Fiedler value, corresponding vector is the Fiedler vector

Fiedler value := approximates minimum cut needed to partition graph

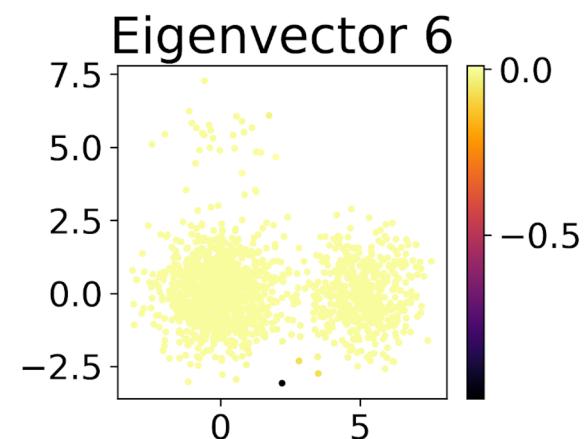
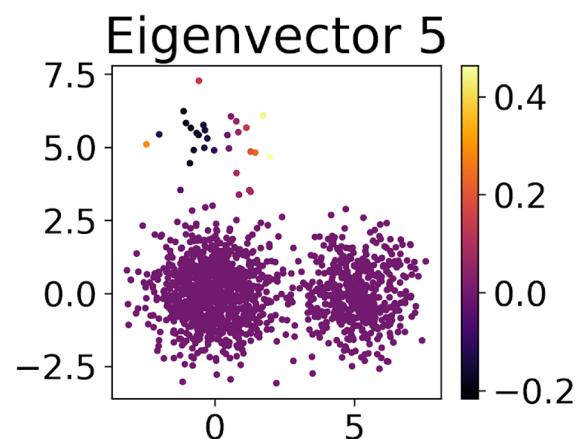
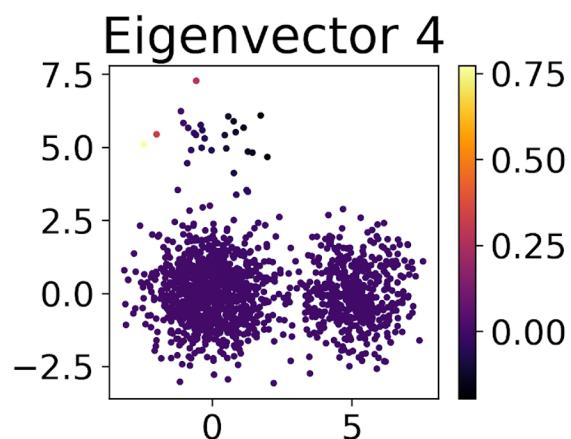
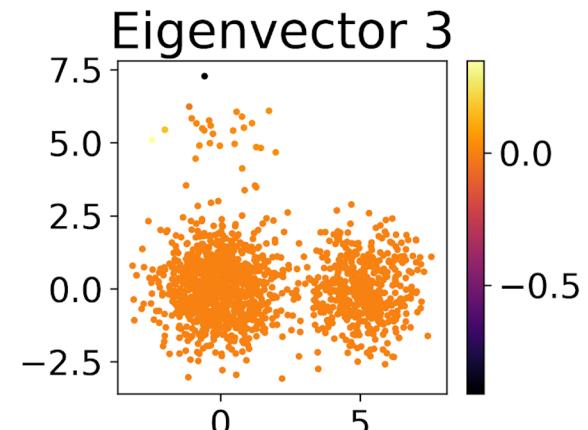
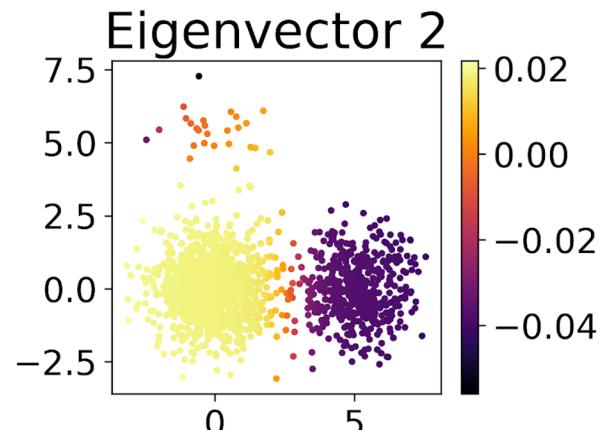
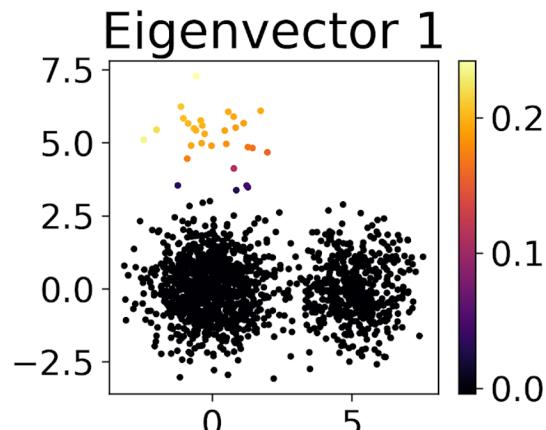
Value if graph was already in two components?

Vector can be used for partitioning, positive in one partition, negative in another

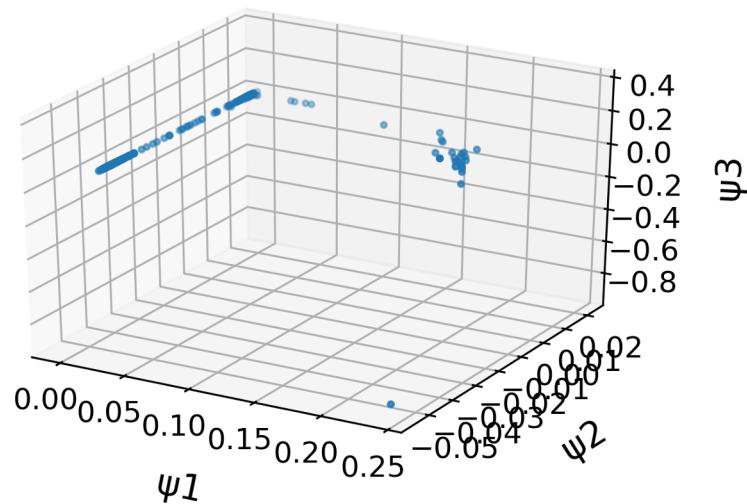
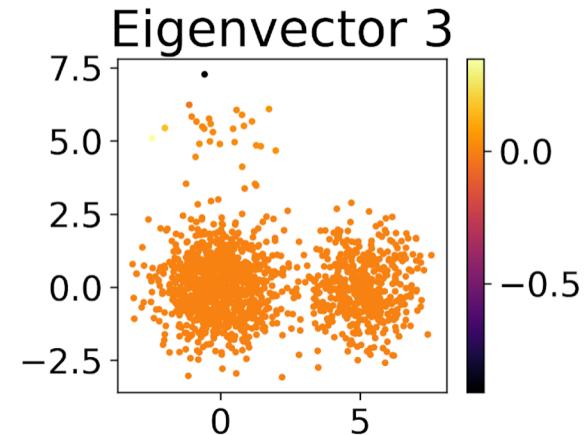
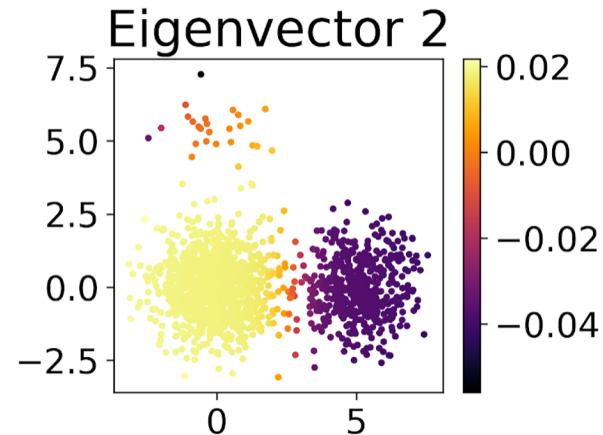
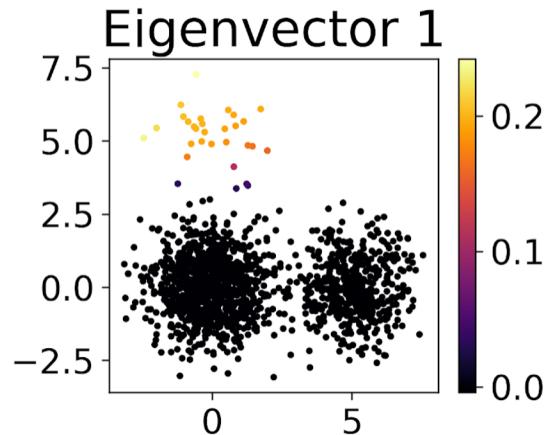


There is a gap between 4th and 5th values, increases suddenly, indicates that there are 4 clusters in the graph roughly.

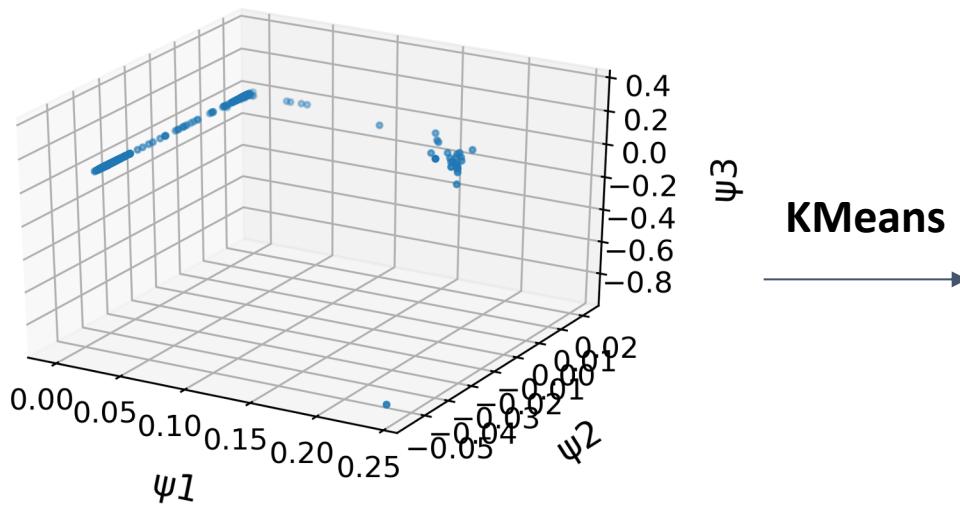
Spectral clustering uses eigenvectors of the graph Laplacian for clustering



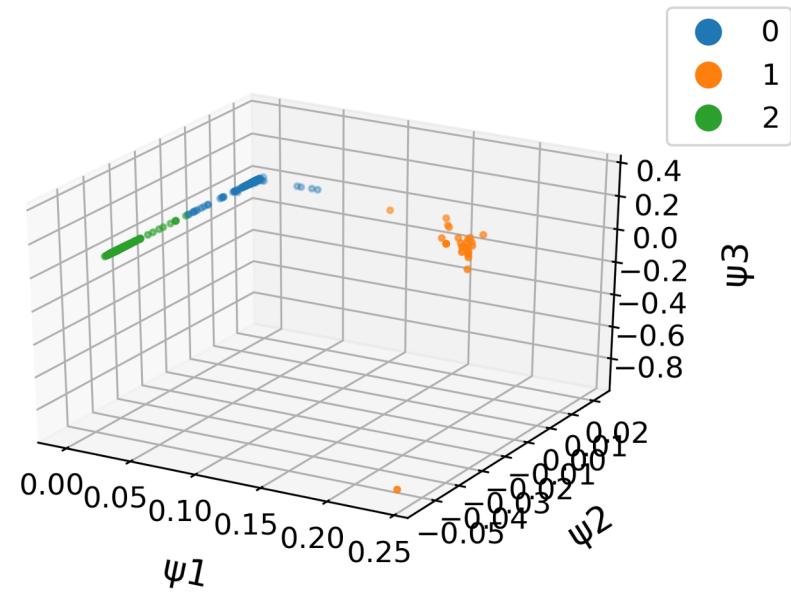
Spectral clustering uses eigenvectors of the graph Laplacian for clustering



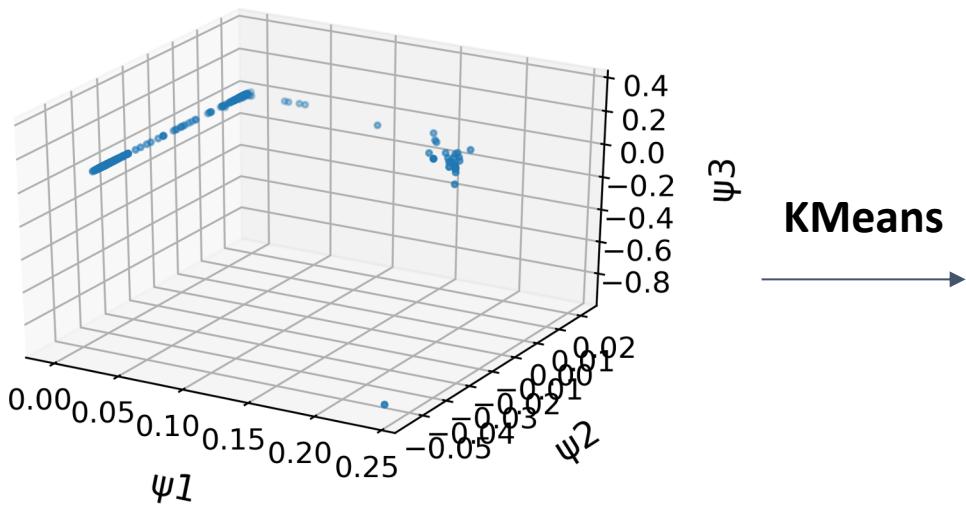
Spectral clustering uses eigenvectors of the graph Laplacian for clustering



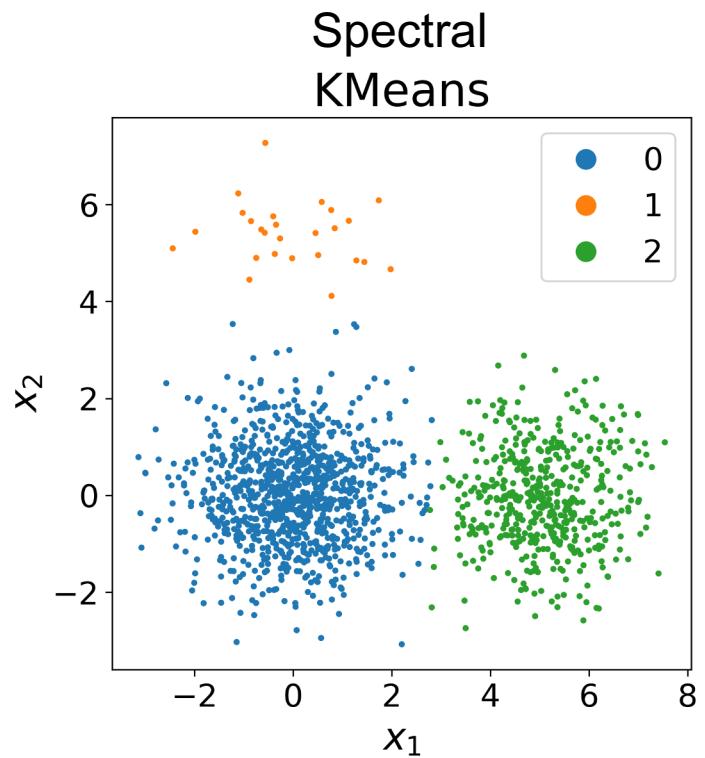
KMeans



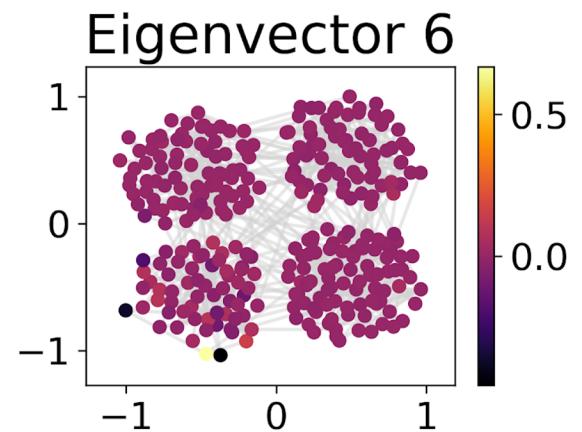
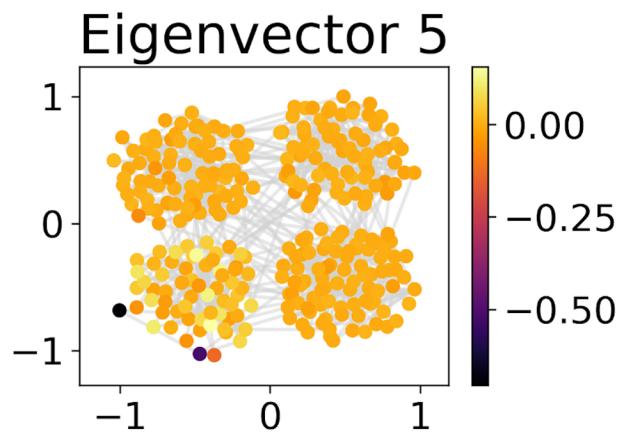
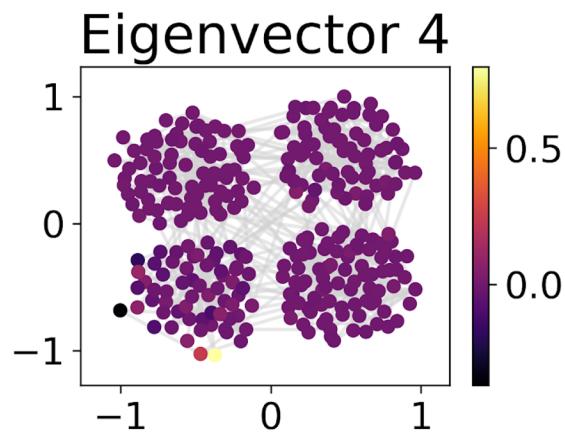
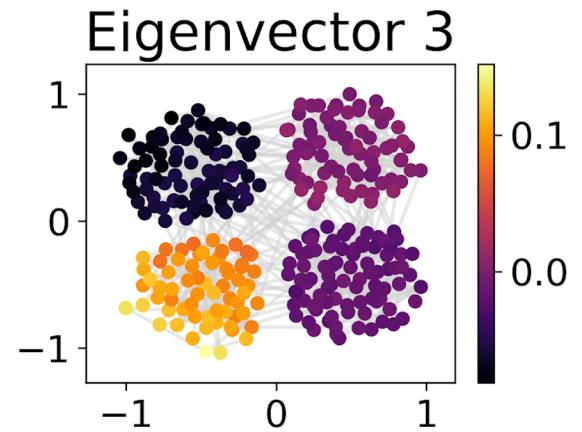
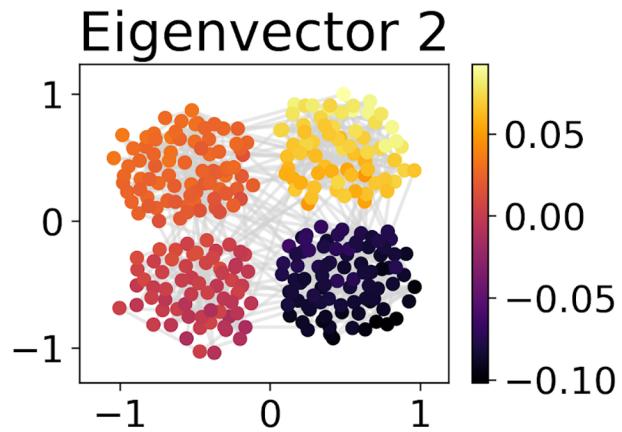
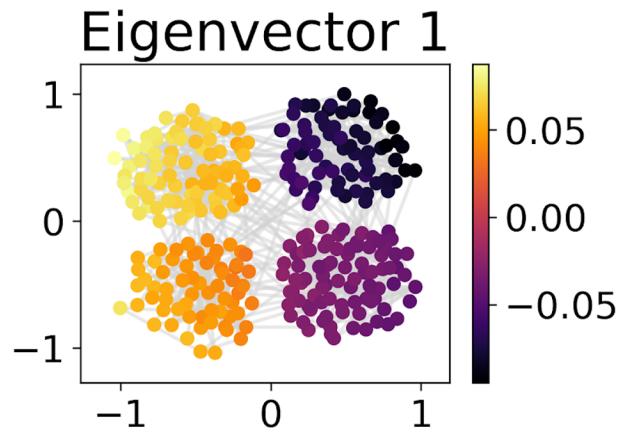
Spectral clustering uses eigenvectors of the graph Laplacian for clustering



KMeans

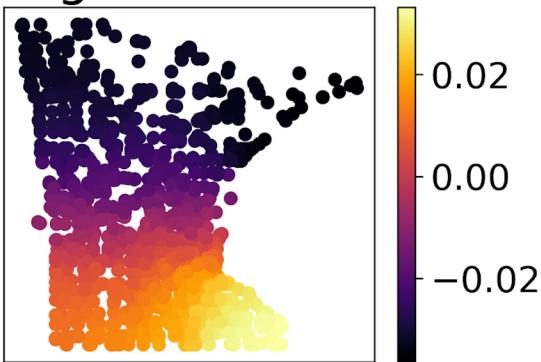


Spectra of various graphs

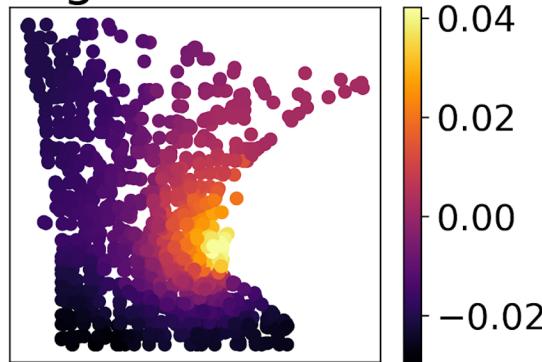


Spectra of various graphs

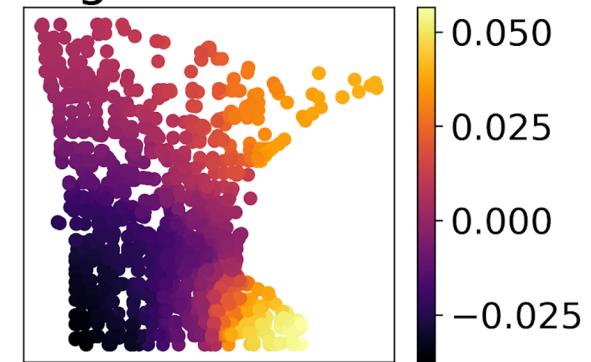
Eigenvector 1



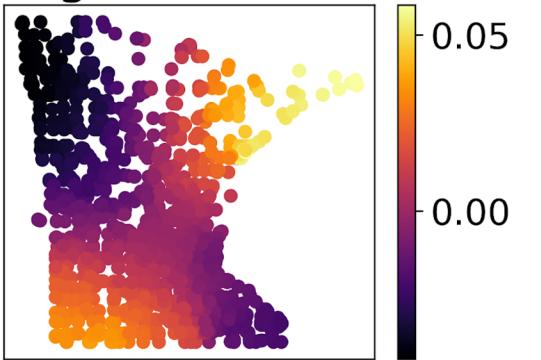
Eigenvector 2



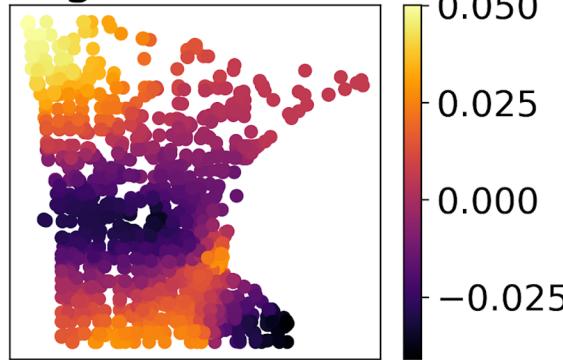
Eigenvector 3



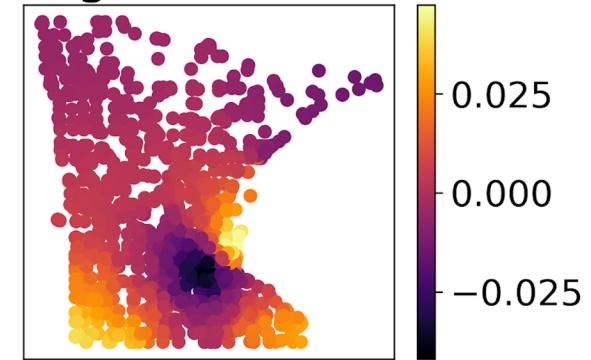
Eigenvector 4



Eigenvector 5



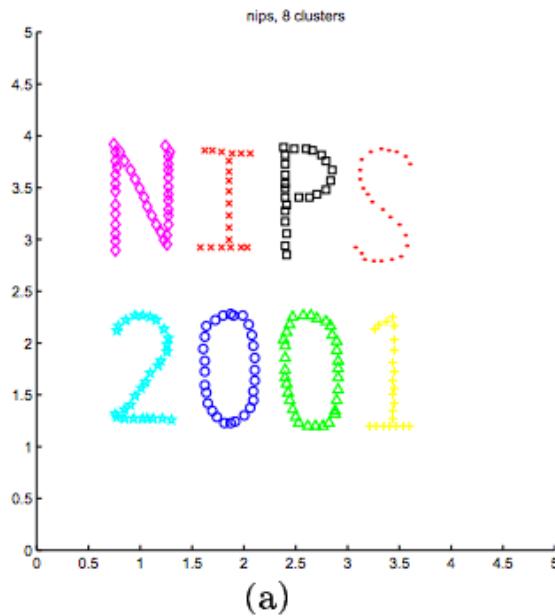
Eigenvector 6



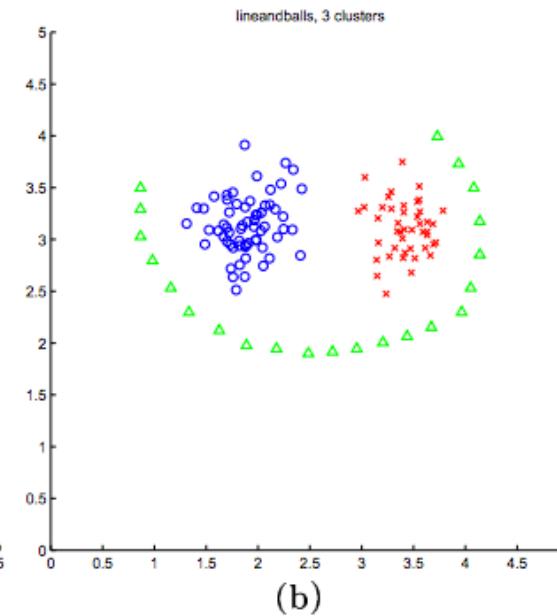
What are advantages of using the graph spectrum for k-Means?

What are advantages of using the graph spectrum for k-Means?

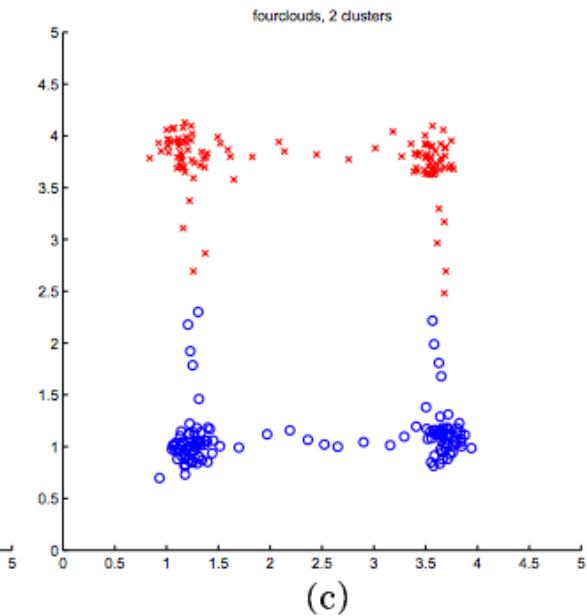
- Good for clusters of arbitrary shape
- Good for data that is just a graph
- Only need connectivity information!



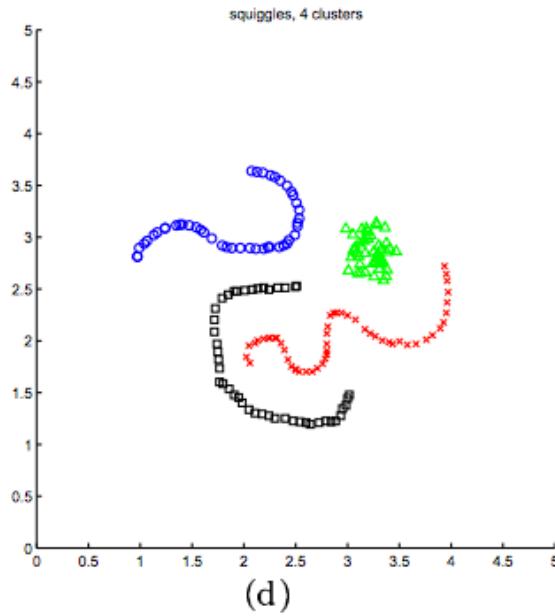
(a)



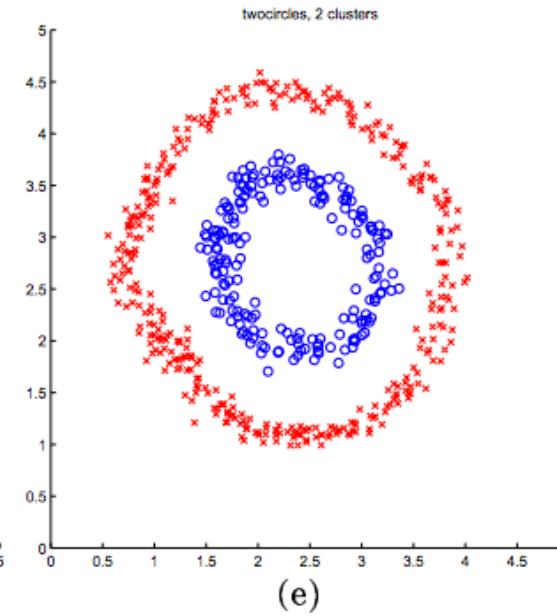
(b)



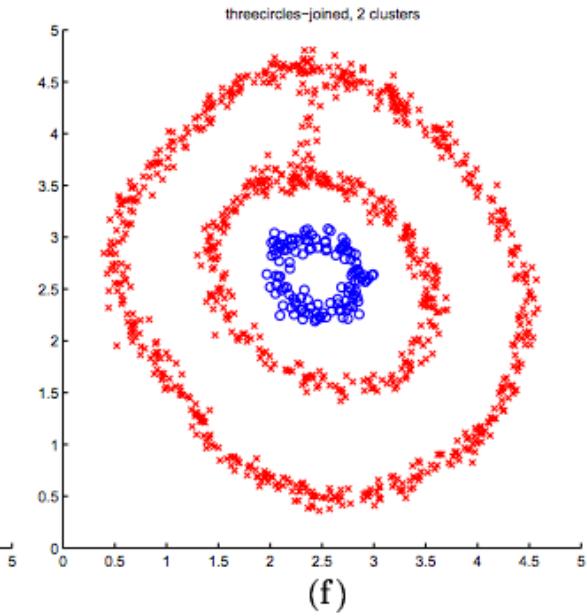
(c)



(d)

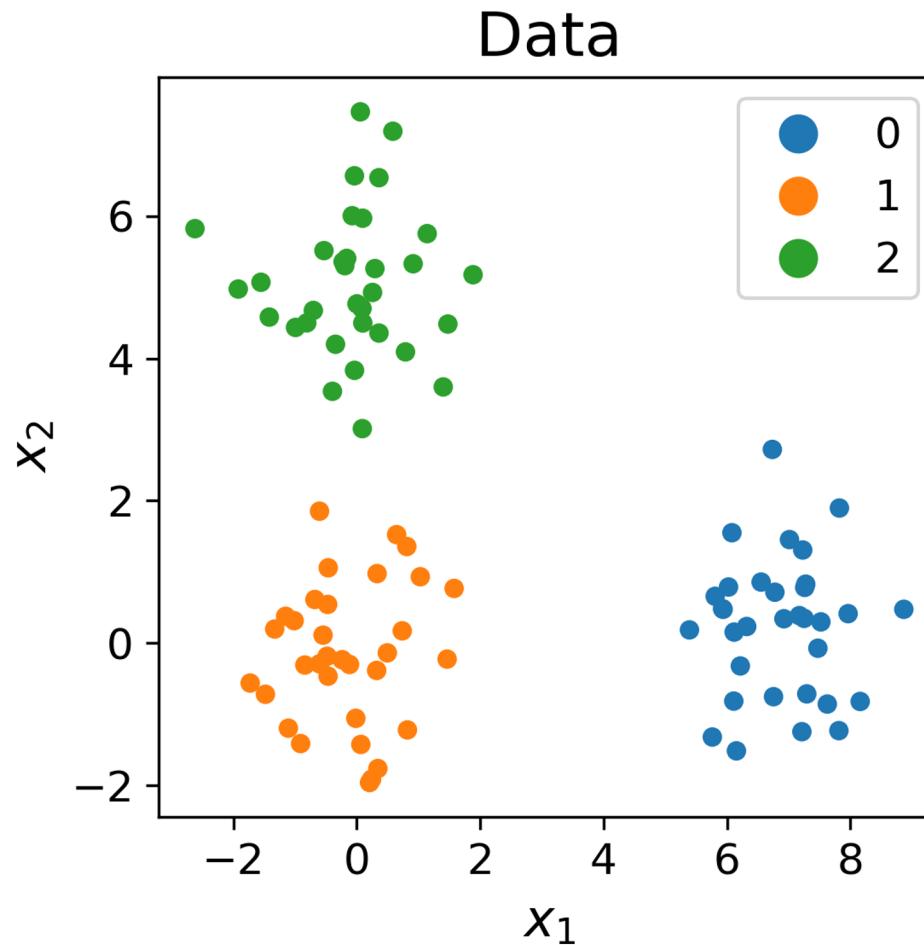


(e)

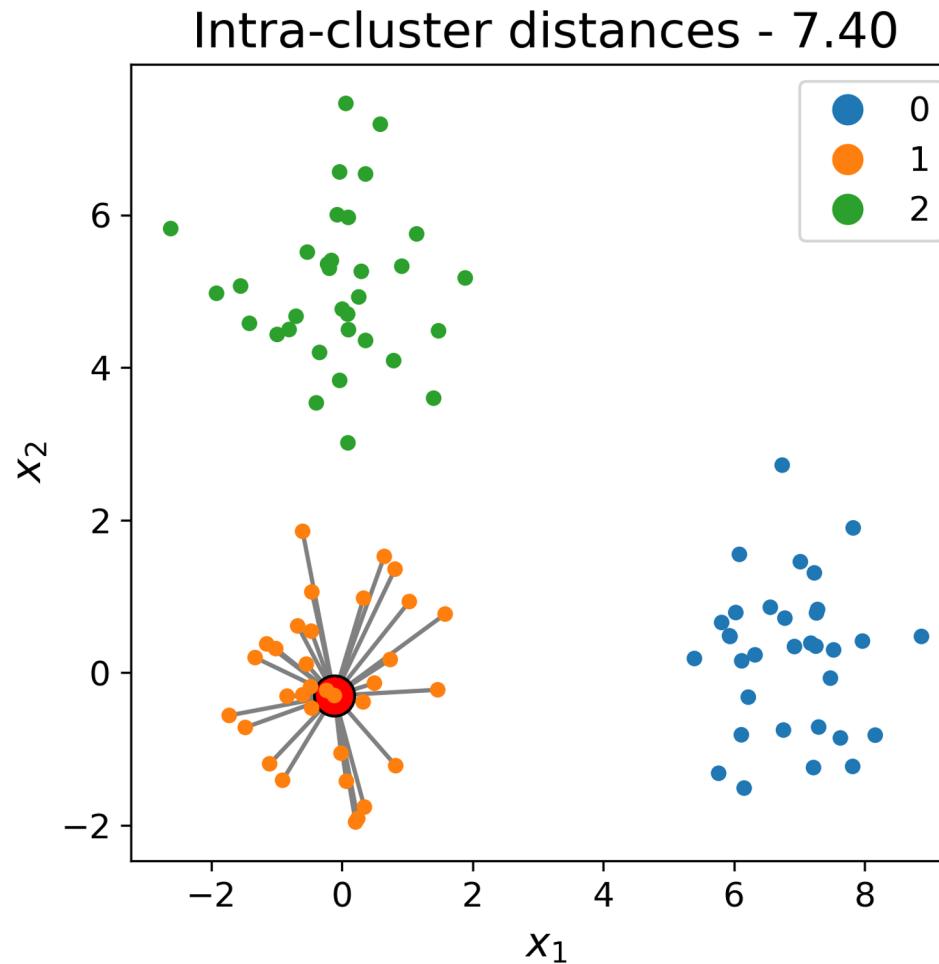


(f)

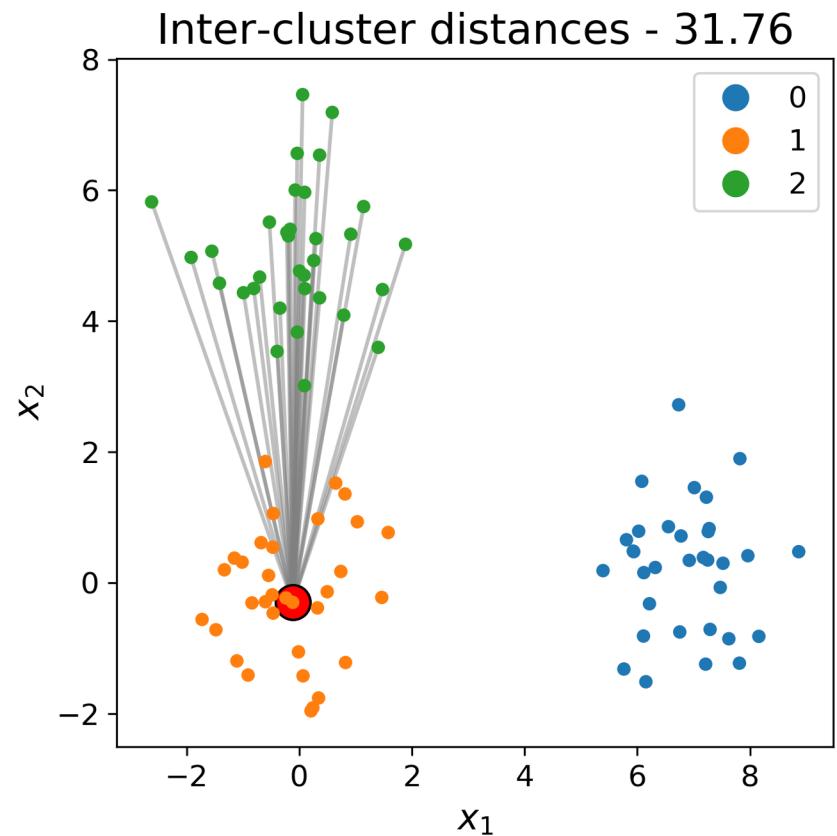
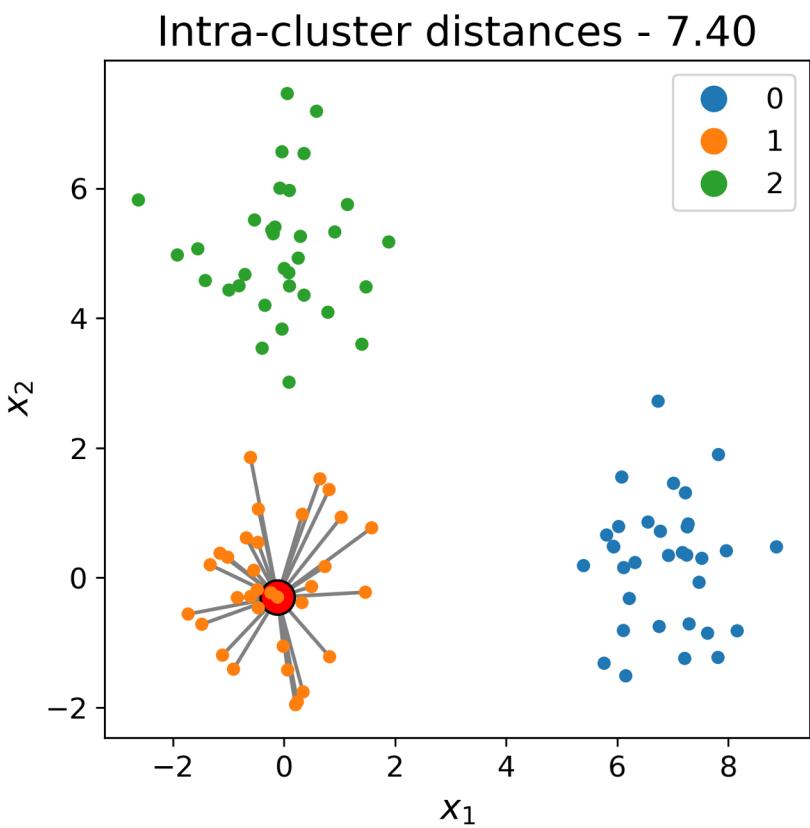
How do we decide the correct number of clusters?



How do we decide the correct number of clusters?

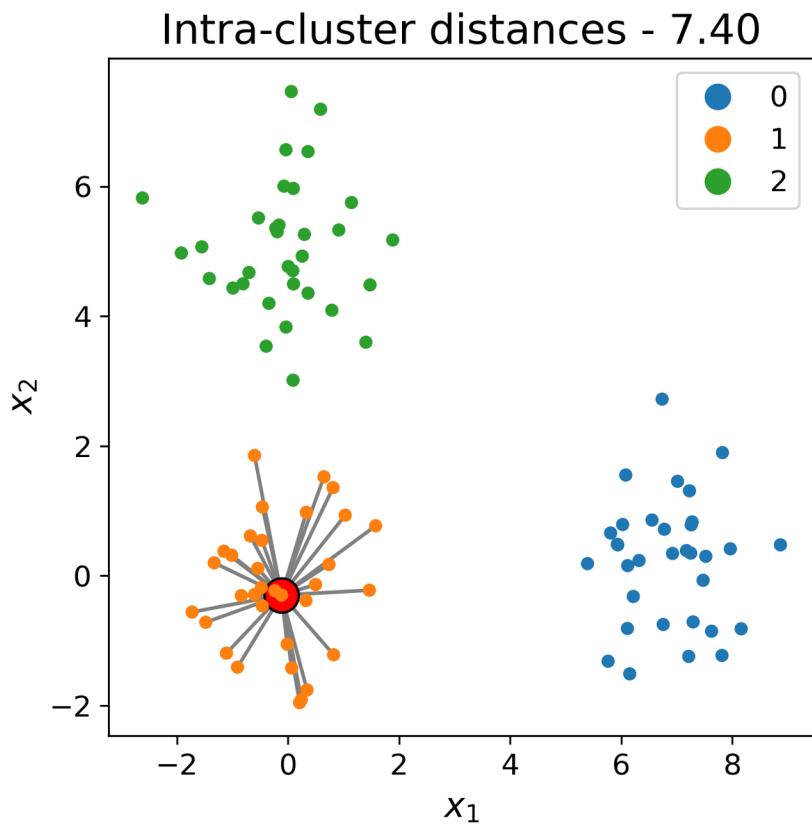


How do we decide the correct number of clusters?

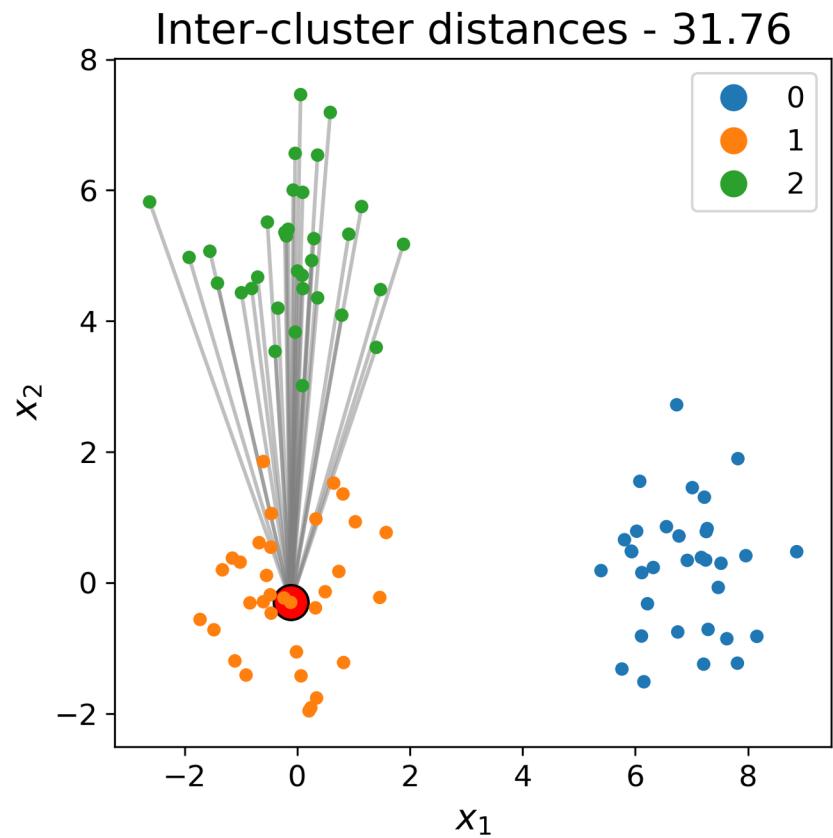


How do we decide the correct number of clusters?

A

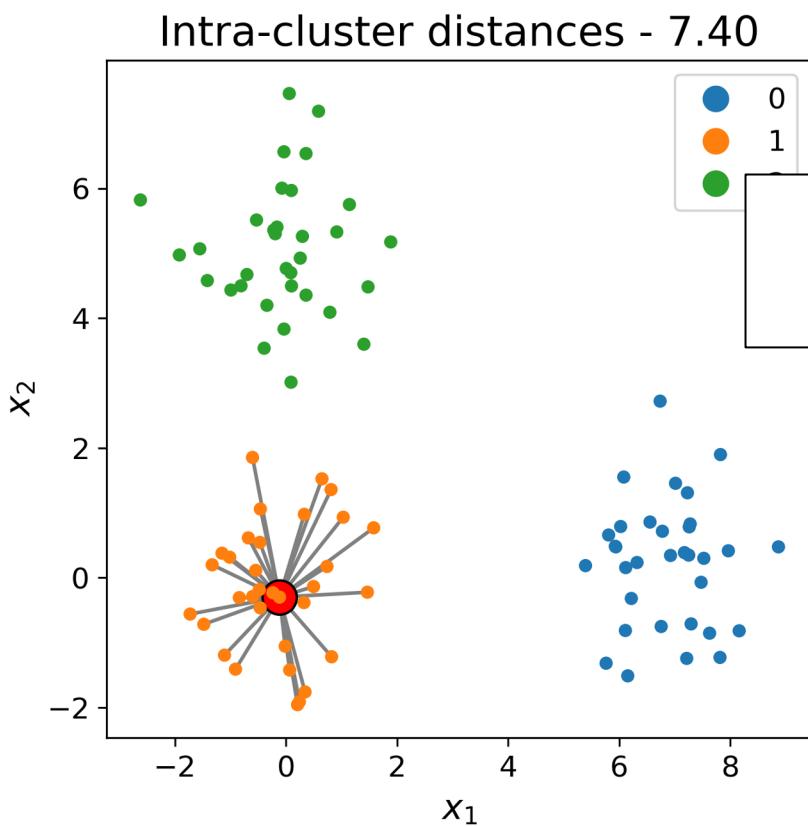


B

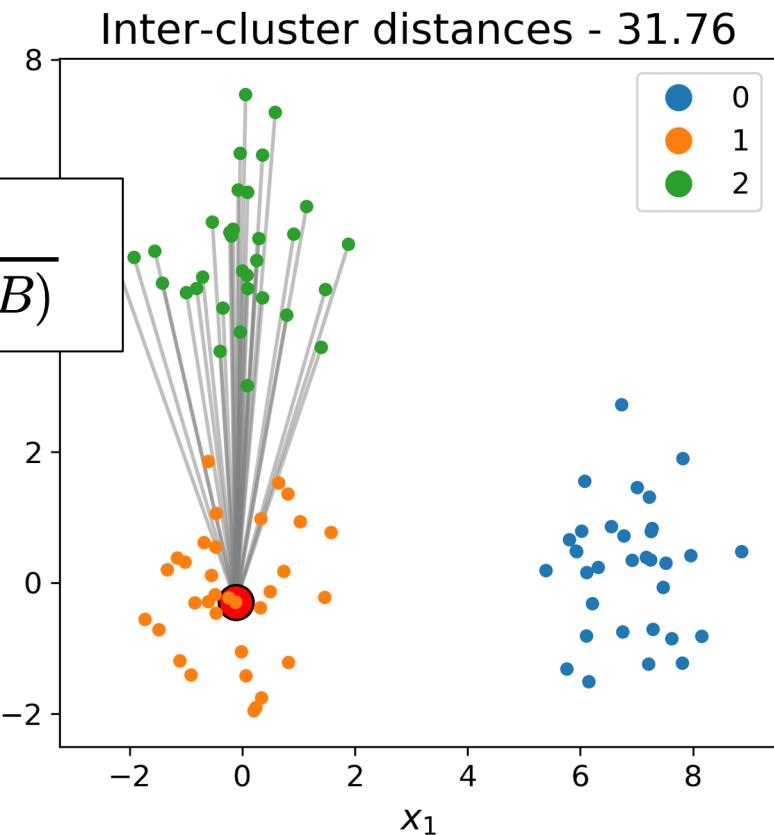


How do we decide the correct number of clusters?

A

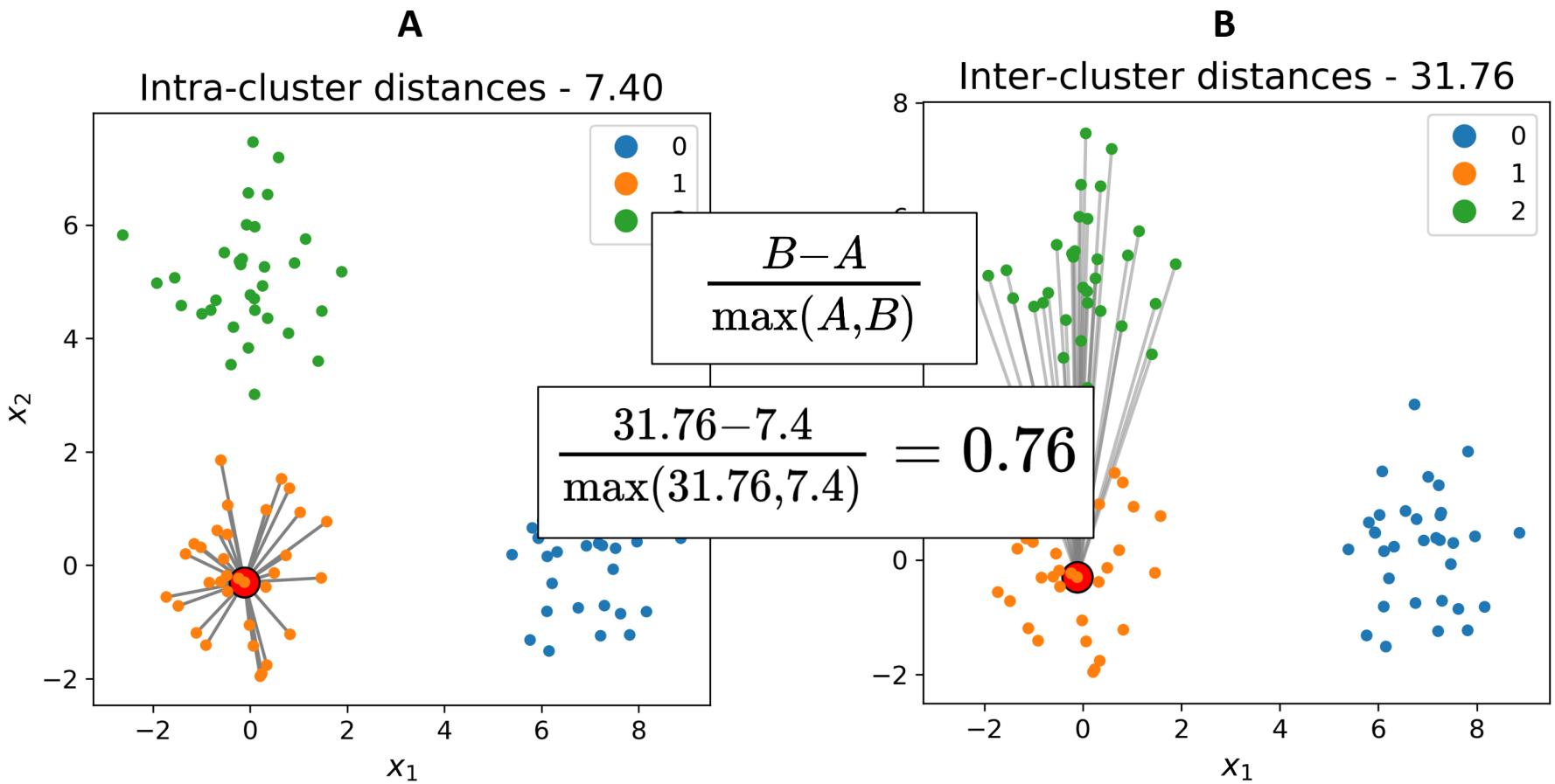


B

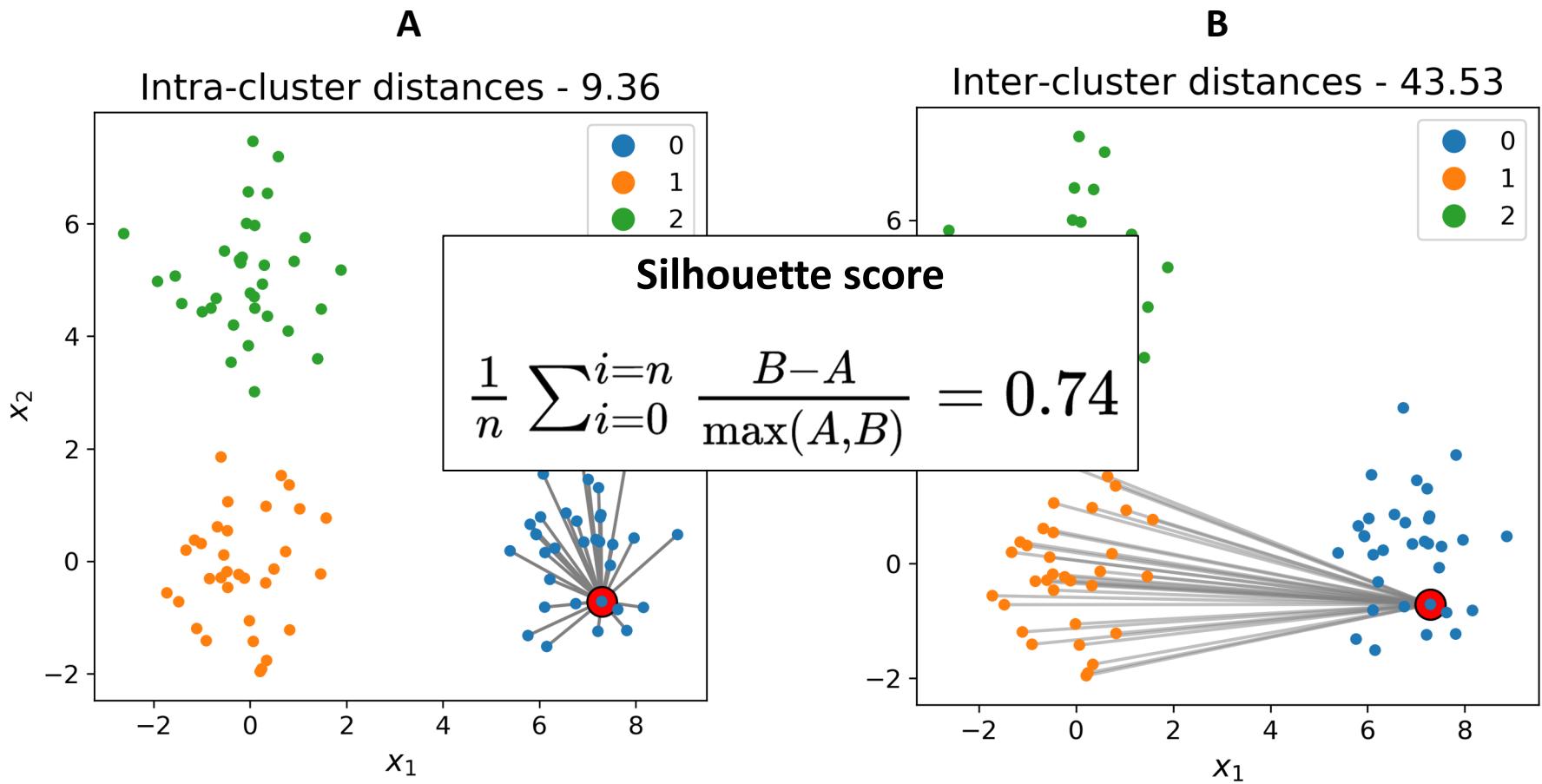


$$\frac{B-A}{\max(A,B)}$$

How do we decide the correct number of clusters?



How do we decide the correct number of clusters?



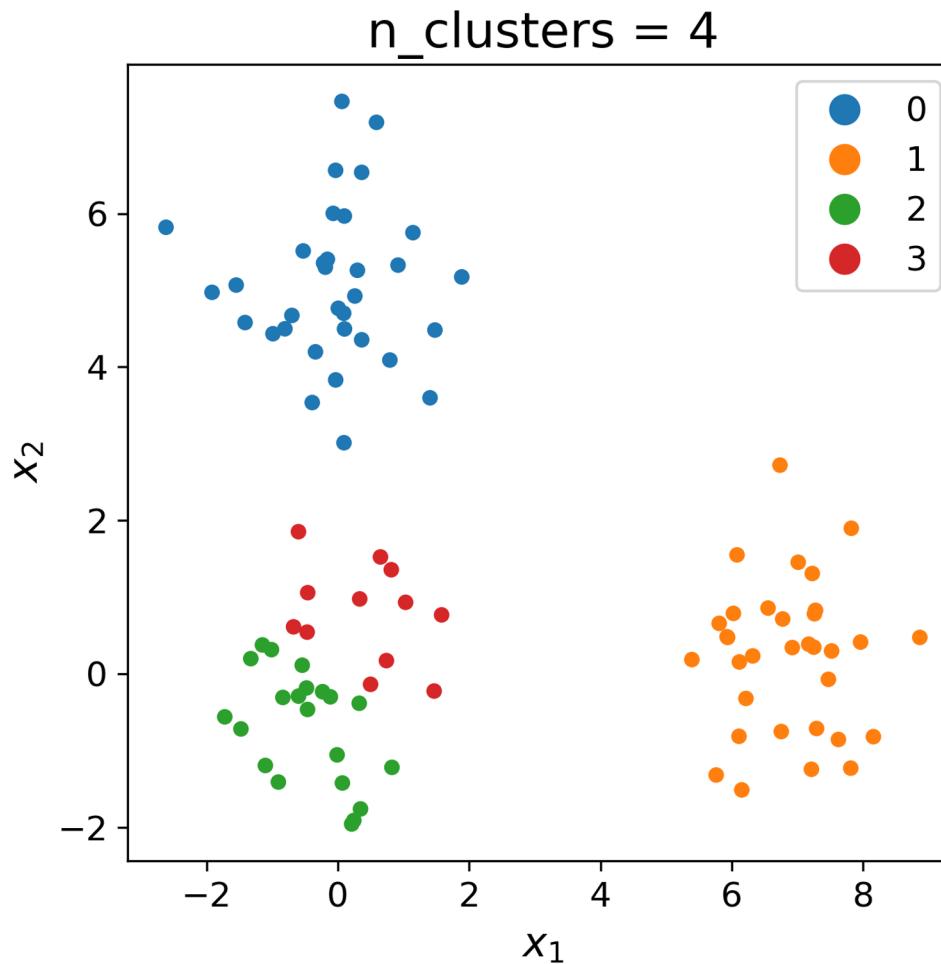
Silhouette score is a measure of fit for cluster assignments

The Silhouette Score is calculated using the mean intra-cluster distance (a) and the mean nearest-cluster distance (b) for each sample.

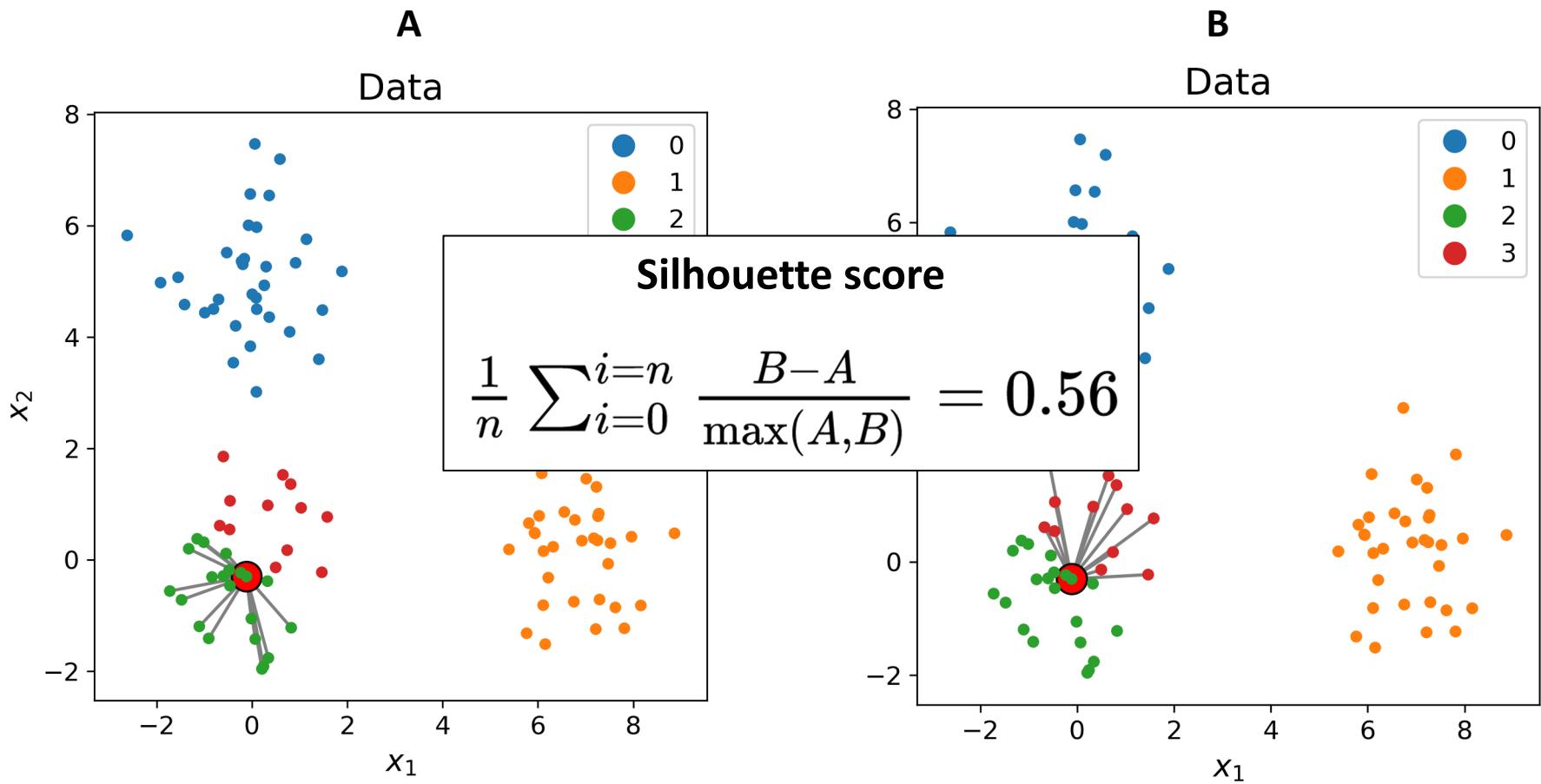
The Silhouette Score for a sample is $(b - a) / \max(a, b)$.

The best value is 1 and the worst value is -1. Values near 0 indicate overlapping clusters. Negative values generally indicate that a sample has been assigned to the wrong cluster, as a different cluster is more similar.

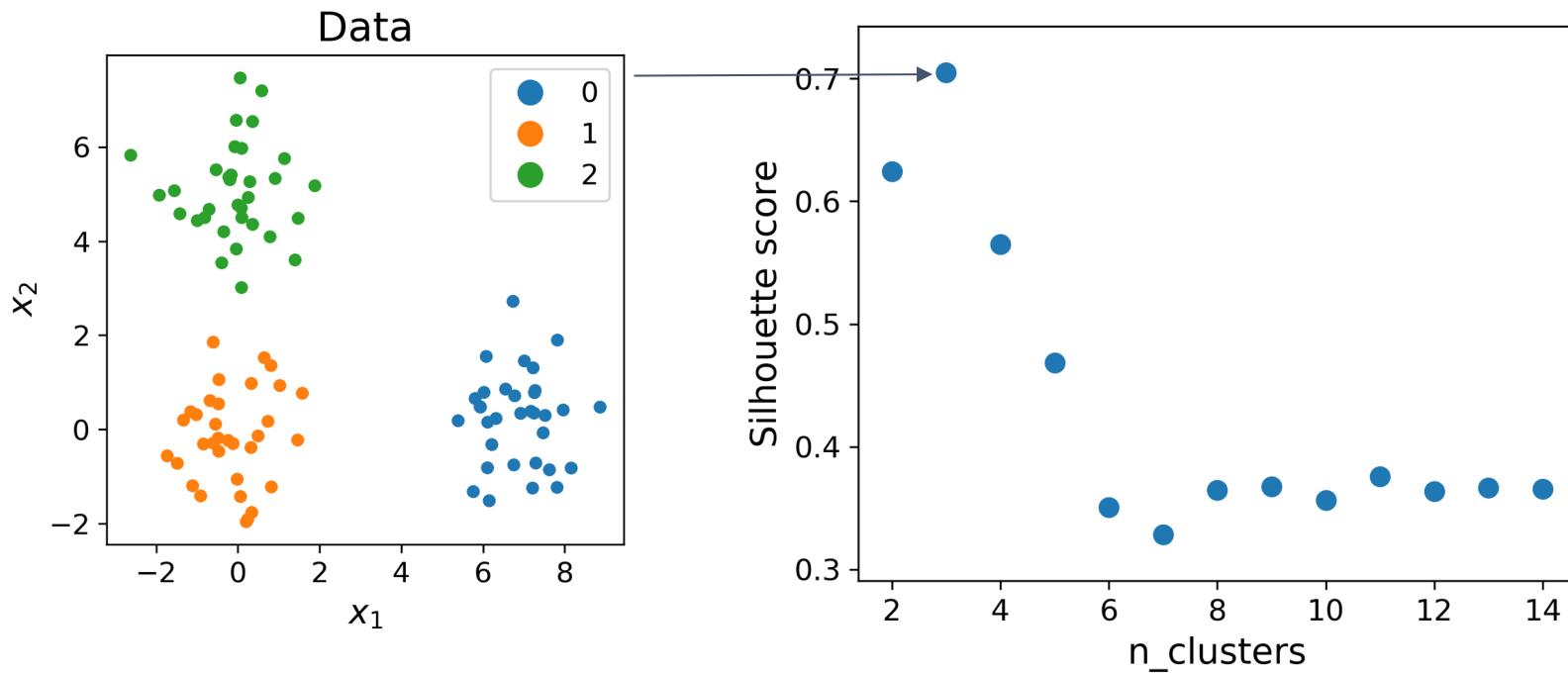
What happens when we change k?



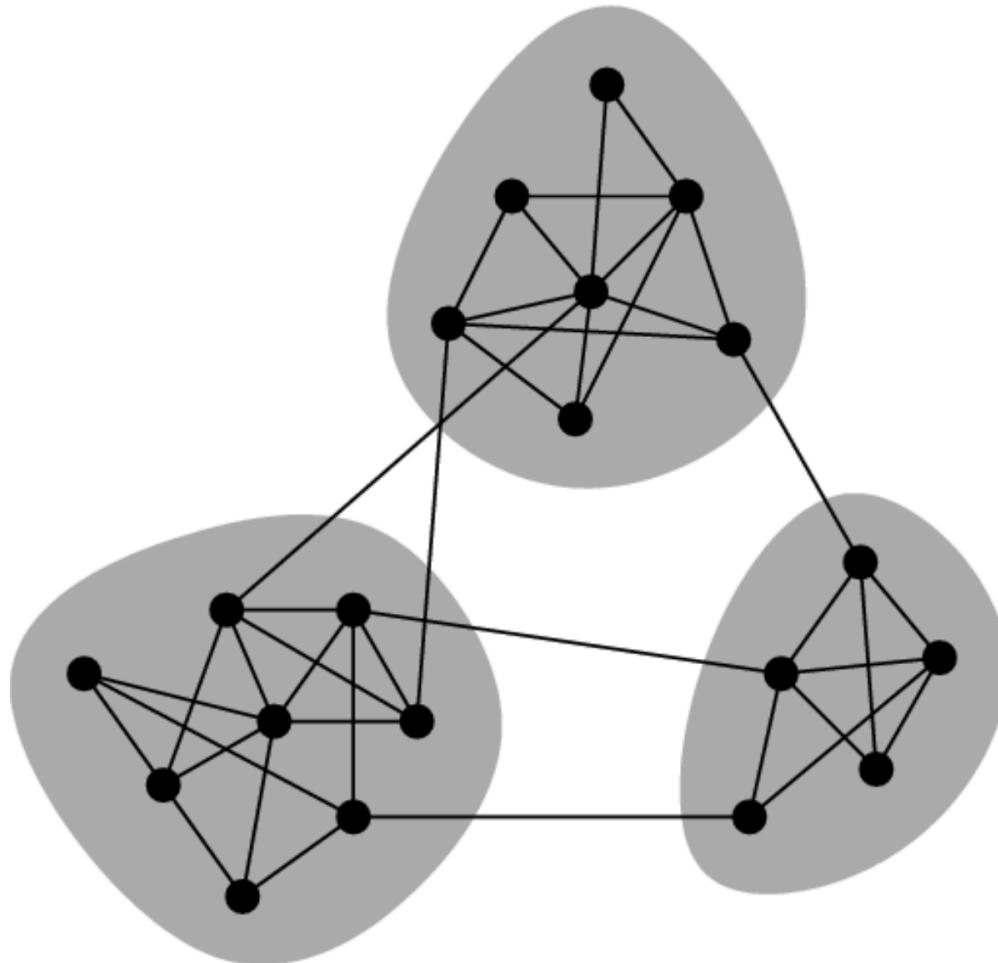
How do we decide the correct number of clusters?



Examining Silhouette score across n_clusters indicates ideal 'k'

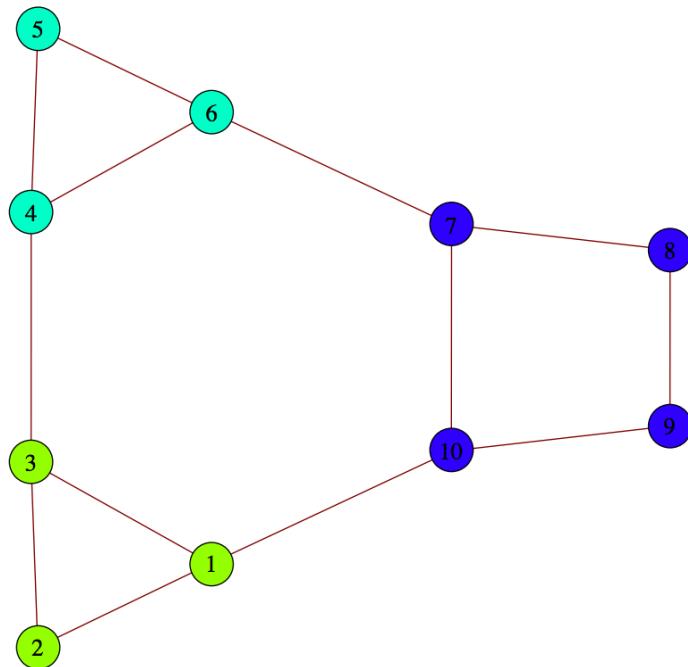


Graph-based Clusters

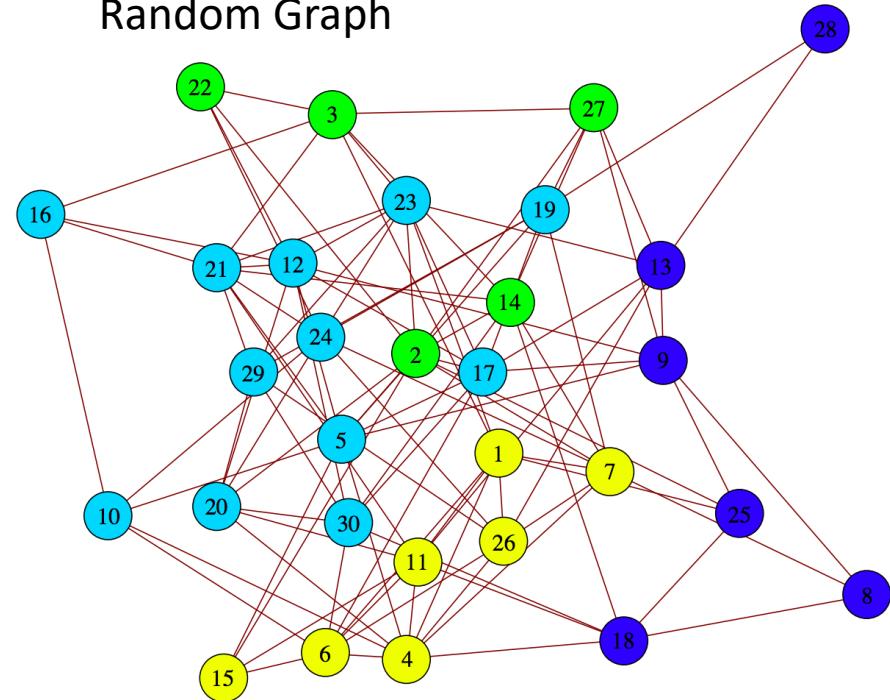


Clustering by “modularity”

Modular Graph



Random Graph



Louvain clustering maximizes modularity of clusters on a graph

$$\text{Modularity} = Q = \frac{1}{2m} \sum_{ij} \left[A_{ij} - \frac{k_i k_j}{2m} \right] \delta(c_i, c_j)$$

m is the sum of all edge weights in the graph

A_{ij} is the edge weight between nodes i and j

k_i is the sum of all edges around node i

δ is 1 if i and j are in the same community and 0 otherwise

Louvain clustering maximizes modularity of clusters on a graph

$$\text{Modularity} = Q = \frac{1}{2m} \sum_{ij} \left[A_{ij} - \frac{k_i k_j}{2m} \right] \delta(c_i, c_j)$$

constant
Adjusts for # edges per node

m is the sum of all edge weights in the graph

A_{ij} is the edge weight between nodes i and j

k_i is the sum of all edges around node i

δ is 1 if i and j are in the same community and 0 otherwise

Louvain clustering maximizes modularity of clusters on a graph

$$\text{Modularity} = Q = \frac{1}{2m} \sum_{ij} \left[A_{ij} - \frac{k_i k_j}{2m} \right] \delta(c_i, c_j)$$

constant
Connectivity between nodes
Adjusts for # edges per node
Only consider connections **within** communities

m is the sum of all edge weights in the graph

A_{ij} is the edge weight between nodes i and j

k_i is the sum of all edges around node i

δ is 1 if i and j are in the same community and 0 otherwise

Louvain clustering maximizes modularity of clusters on a graph

$$\text{Modularity} = Q = \frac{1}{2m} \sum_{ij} \left[A_{ij} - \frac{k_i k_j}{2m} \right] \delta(c_i, c_j)$$

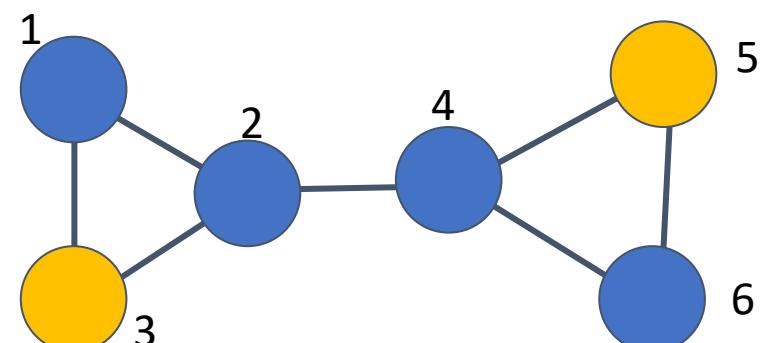
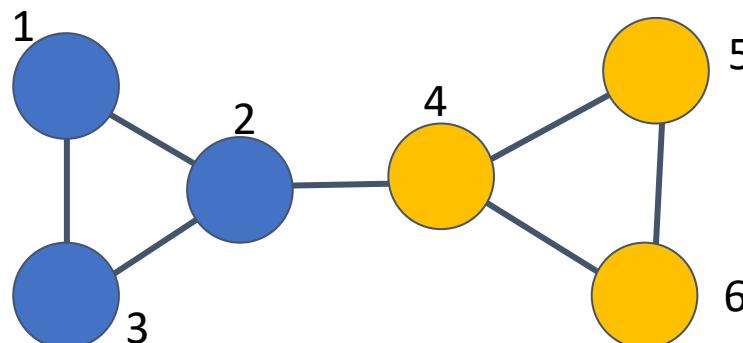
constant
Connectivity between nodes
Adjusts for # edges per node
Only consider connections **within** communities

m is the sum of all edge weights in the graph

A_{ij} is the edge weight between nodes i and j

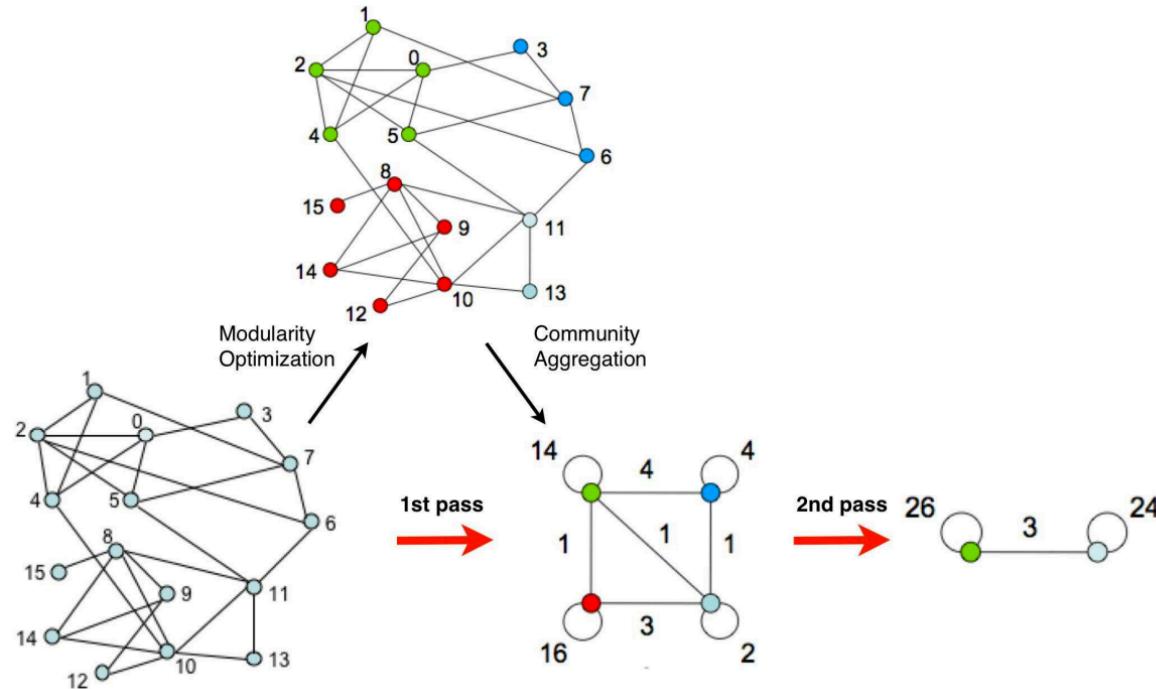
k_i is the sum of all edges around node i

δ is 1 if i and j are in the same community and 0 otherwise



Greedy Modularity Optimization (Newman 2003)

- Start with each point in its own cluster
- Merge two clusters that would increase modularity by the most
- Stop when all merges reduce modularity



s41598-019-41695-z.pdf (page 1 of 12)

www.nature.com/scientificreports/

SCIENTIFIC REPORTS



OPEN

From Louvain to Leiden: guaranteeing well-connected communities

V. A. Traag  & L. Waltman  & N. J. van Eck 

Community detection is often used to understand the structure of large and complex networks. One of the most popular algorithms for uncovering community structure is the so-called Louvain algorithm. We show that this algorithm has a major defect that largely went unnoticed until now: the Louvain algorithm may yield arbitrarily badly connected communities. In the worst case, communities may even be disconnected, especially when running the algorithm iteratively. In our experimental analysis, we observe that up to 25% of the communities are badly connected and up to 16% are disconnected. To address this problem, we introduce the Leiden algorithm. We prove that the Leiden algorithm yields communities that are guaranteed to be connected. In addition, we prove that, when the Leiden algorithm is applied iteratively, it converges to a partition in which all subsets of all communities are locally optimally assigned. Furthermore, by relying on a fast local move approach, the Leiden algorithm runs faster than the Louvain algorithm. We demonstrate the performance of the Leiden algorithm for several benchmark and real-world networks. We find that the Leiden algorithm is faster than the Louvain algorithm and uncovers better partitions, in addition to providing explicit guarantees.

In many complex networks, nodes cluster and form relatively dense groups—often called communities^{1,2}. Such a modular structure is usually not known beforehand. Detecting communities in a network is therefore an important problem. One of the best-known methods for community detection is called modularity³. This method tries to maximise the difference between the actual number of edges in a community and the expected number of such edges. We denote by e_c the actual number of edges in community c . The expected number of edges can be expressed as $\frac{K_c^2}{2m}$, where K_c is the sum of the degrees of the nodes in community c and m is the total number of edges in the network. This way of defining the expected number of edges is based on the so-called configuration model. Modularity is given by

$$\mathcal{H} = \frac{1}{2m} \sum_c \left[e_c - \gamma \frac{K_c^2}{2m} \right], \quad (1)$$

where $\gamma > 0$ is a resolution parameter⁴. Higher resolutions lead to more communities, while lower resolutions lead to fewer communities.

Optimising modularity is NP-hard⁵, and consequently many heuristic algorithms have been proposed, such as hierarchical agglomeration⁶, extremal optimisation⁷, simulated annealing⁸ and spectral⁹ algorithms. One of the most popular algorithms to optimise modularity is the so-called Louvain algorithm¹⁰, named after the location of its authors. It was found to be one of the fastest and best performing algorithms in comparative analyses^{11,12}, and it is one of the most-cited works in the community detection literature.

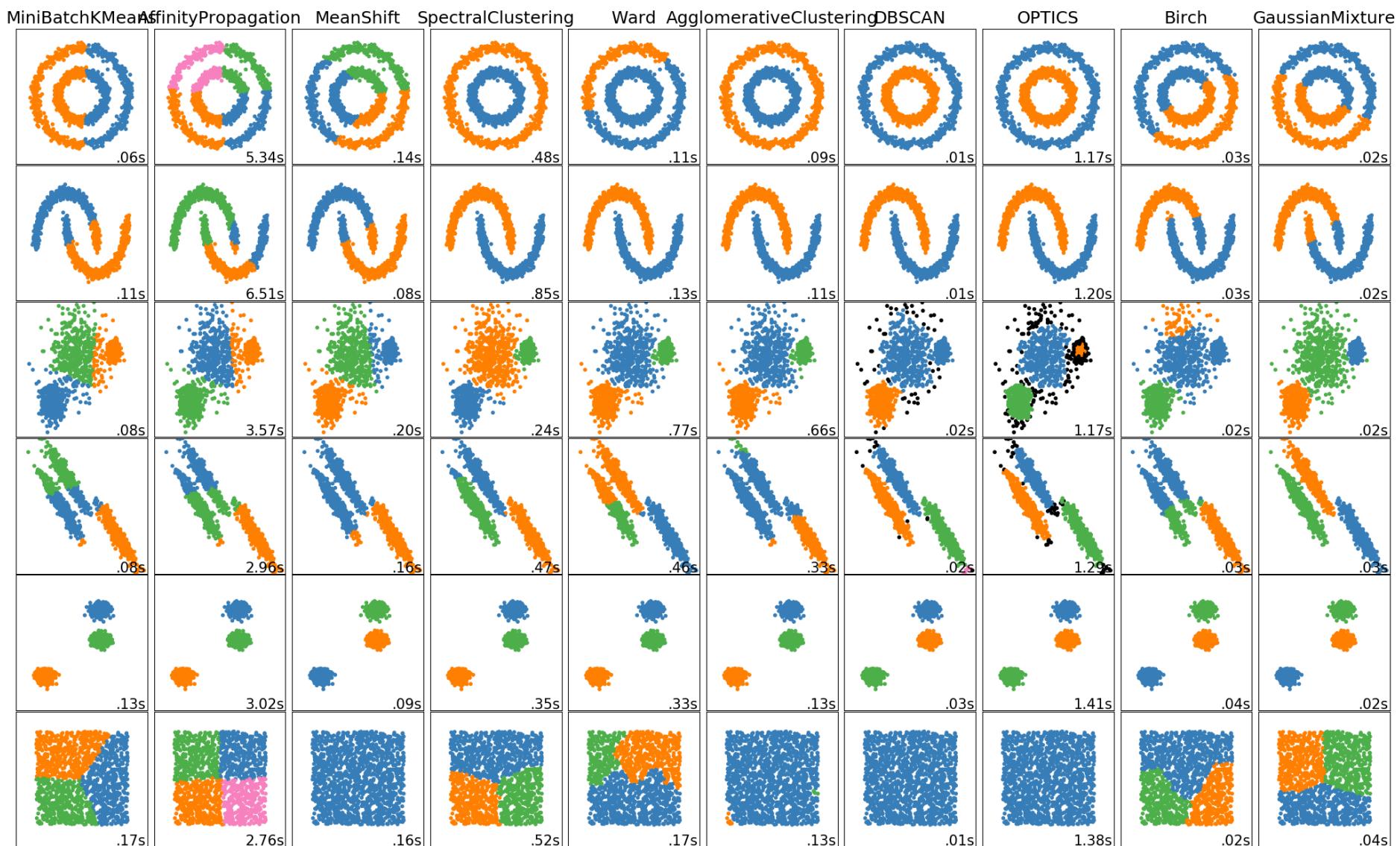
Although originally defined for modularity, the Louvain algorithm can also be used to optimise other quality functions. An alternative quality function is the Constant Potts Model (CPM)¹³, which overcomes some limitations of modularity. CPM is defined as

$$\mathcal{H} = \sum_c \left[e_c - \gamma \binom{n_c}{2} \right], \quad (2)$$

Centre for Science and Technology Studies, Leiden University, Leiden, The Netherlands. Correspondence and requests for materials should be addressed to V.A.T. (email: v.a.traag@cwts.leidenuniv.nl)

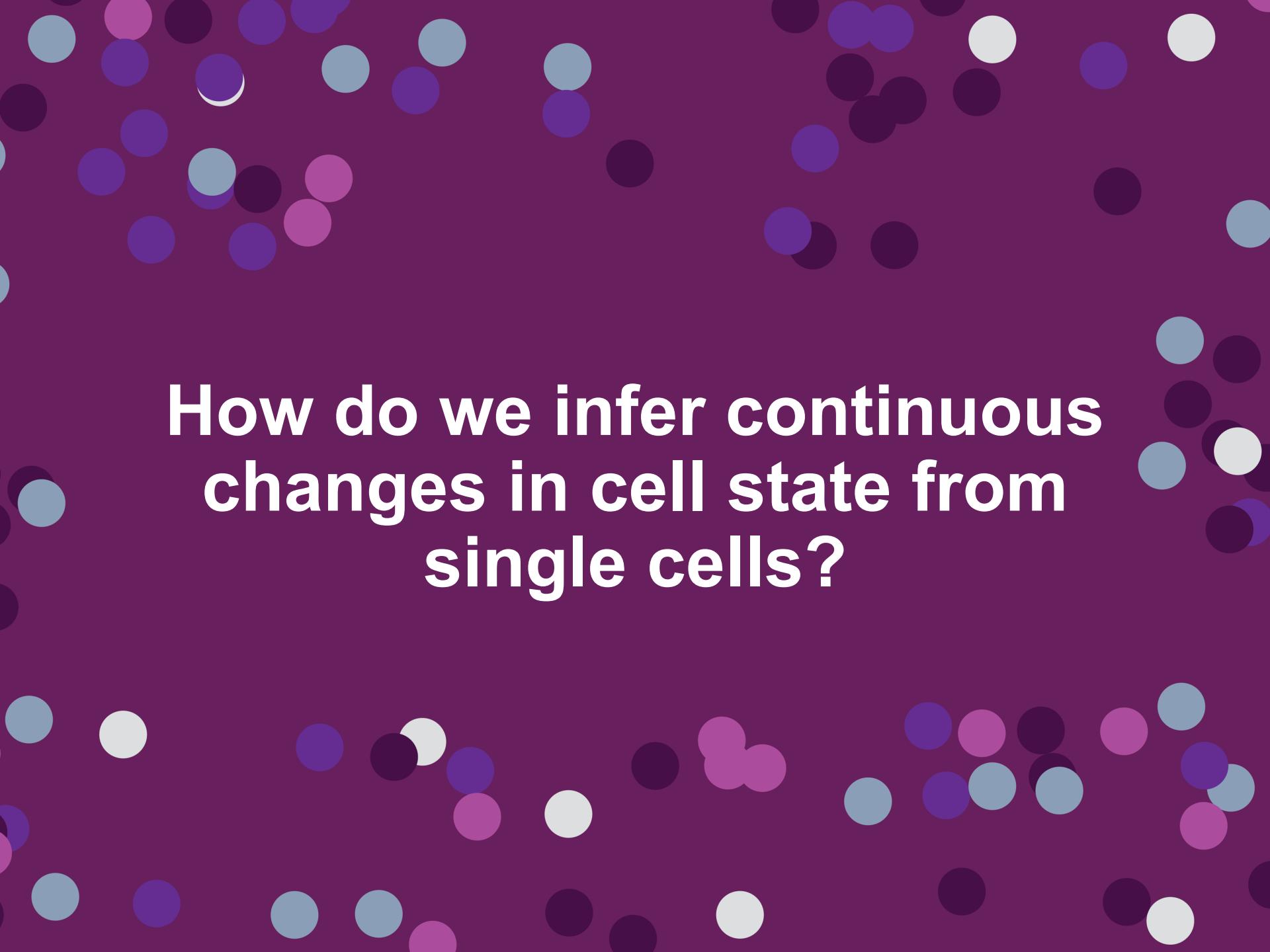
SCIENTIFIC REPORTS | (2019) 9:5233 | <https://doi.org/10.1038/s41598-019-41695-z> 1

<https://www.nature.com/articles/s41598-019-41695-z>



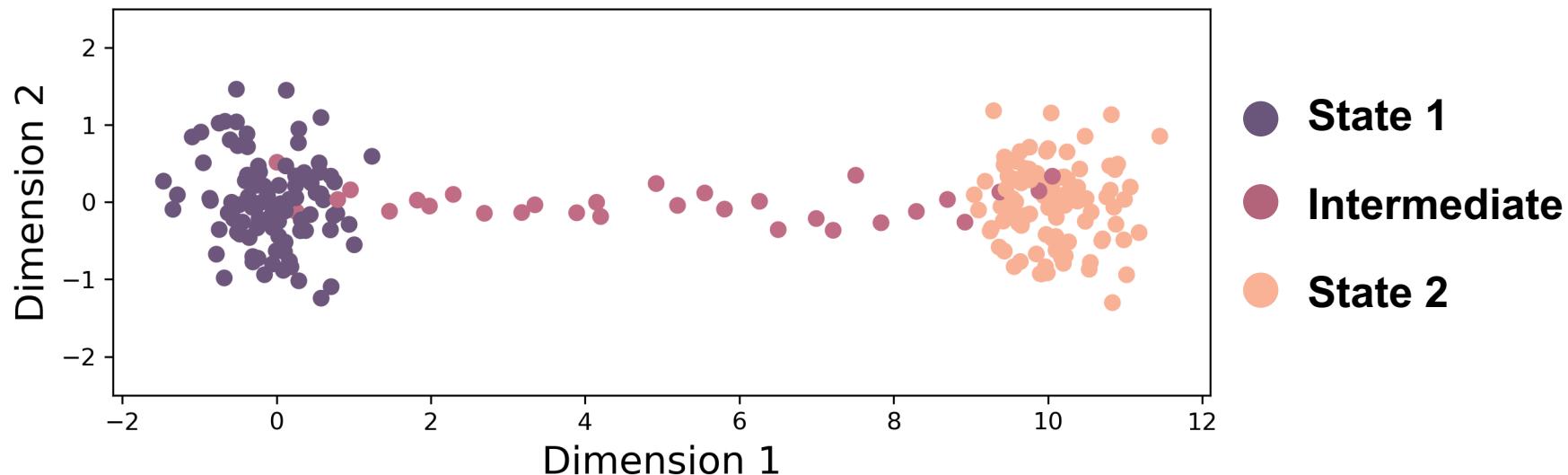
Conclusions

- Clustering algorithms partition data to identify groups of similar observations
- KMeans is an iterative algorithm that minimizes within-cluster distances in the data space
- Spectral clustering minimizes within-cluster distances using eigenvectors of the laplacian
- Louvain / Leiden iteratively aggregate clusters on a graph to increase a modularity heuristic



How do we infer continuous changes in cell state from single cells?

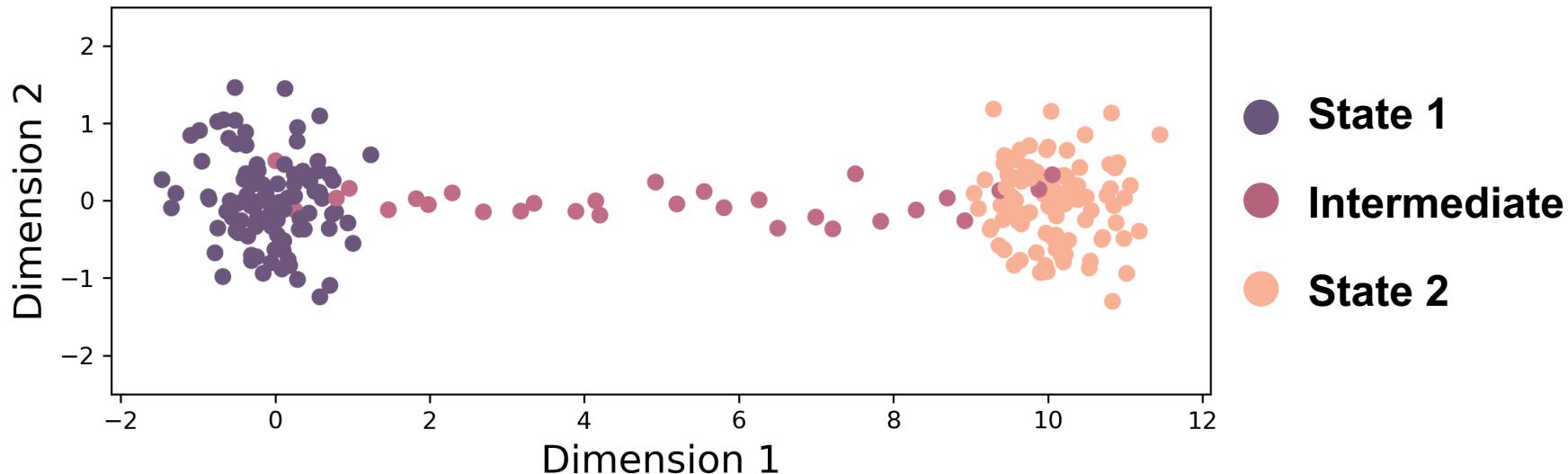
A simple developmental system



Dimension 1: “Genes that change from State 1 to State 2”

Dimension 2: “Noise genes”

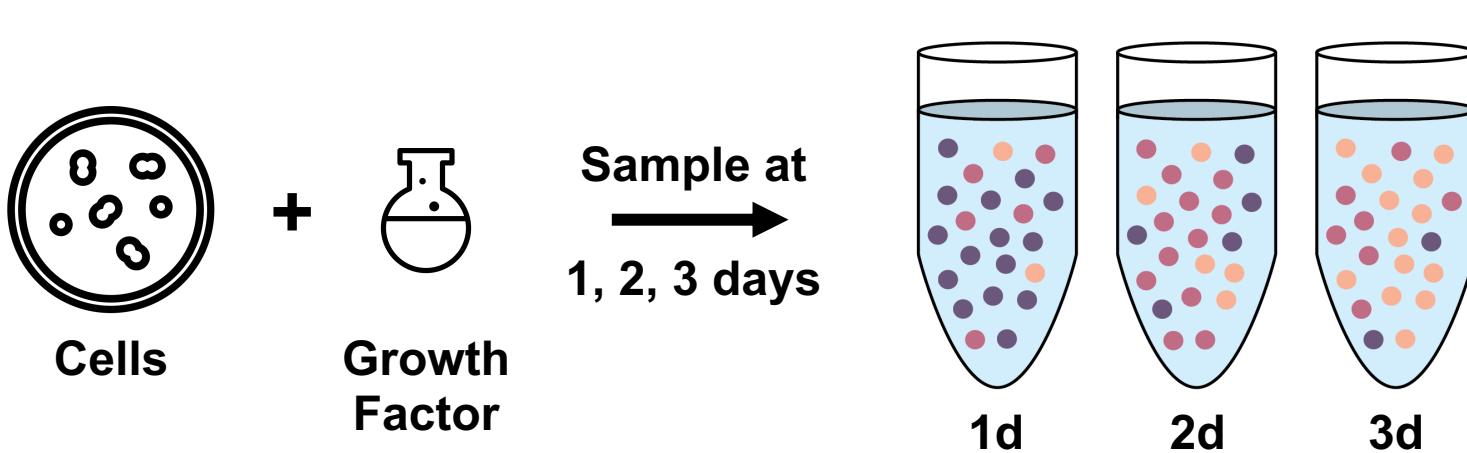
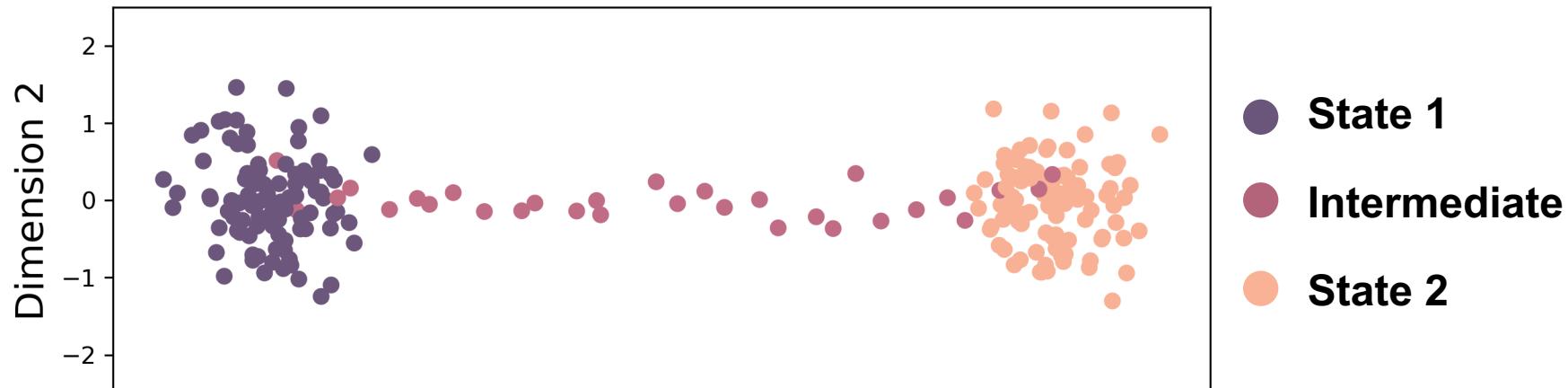
A simple developmental system



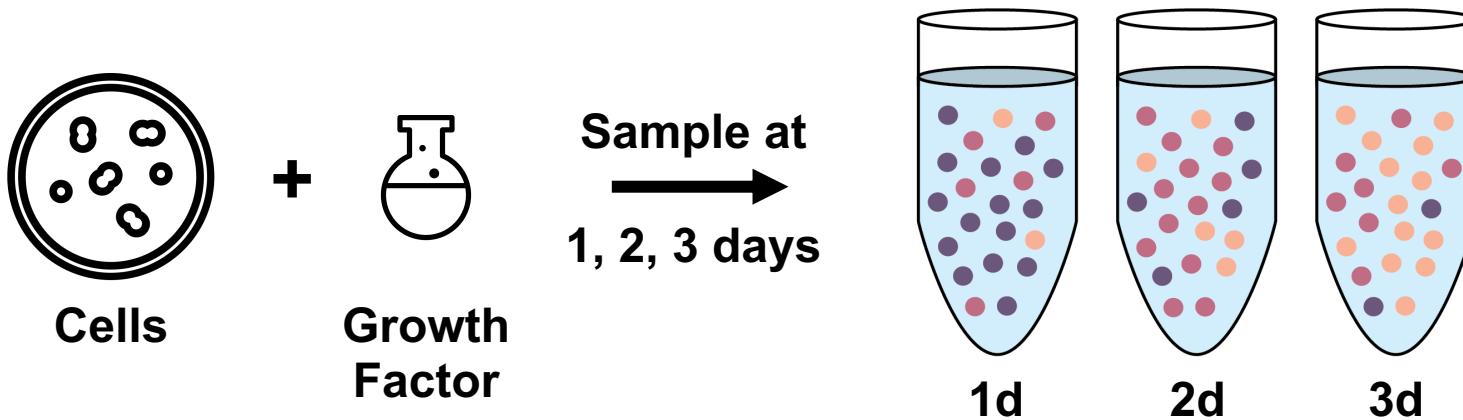
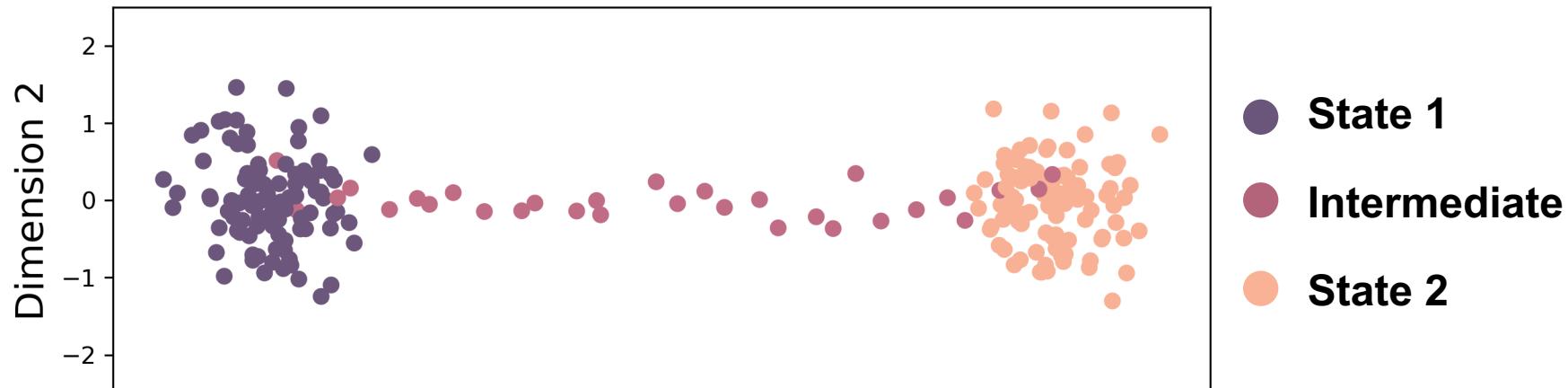
Things we might like to know:

- What genes are expressed in each population?
- What genes change the most from State 1 → State 2?
- What is the ordering of gene expression?
- How do gene-gene relationships change across states?

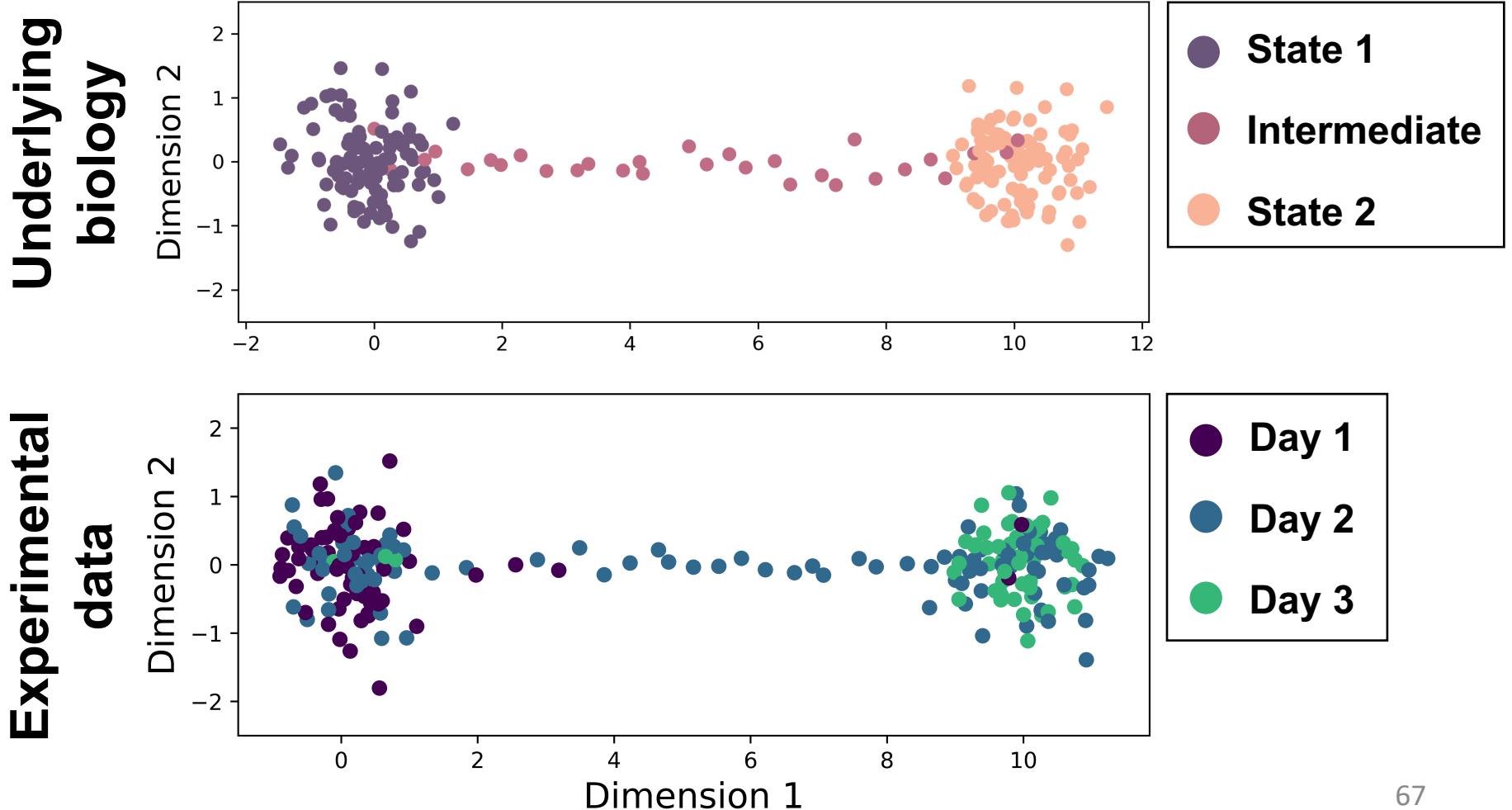
A simple developmental system



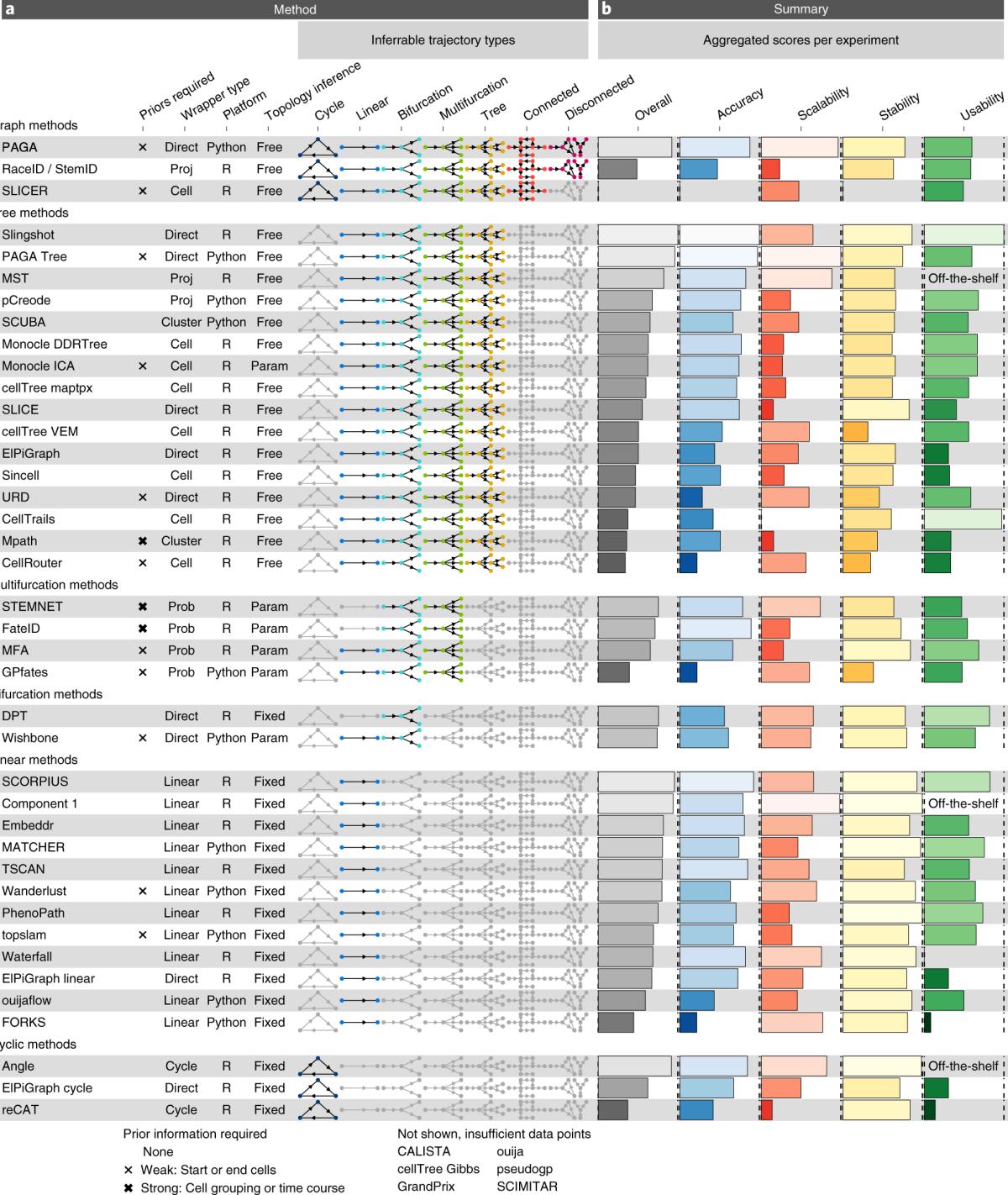
A simple developmental system



We must infer underlying biology from experimental observations

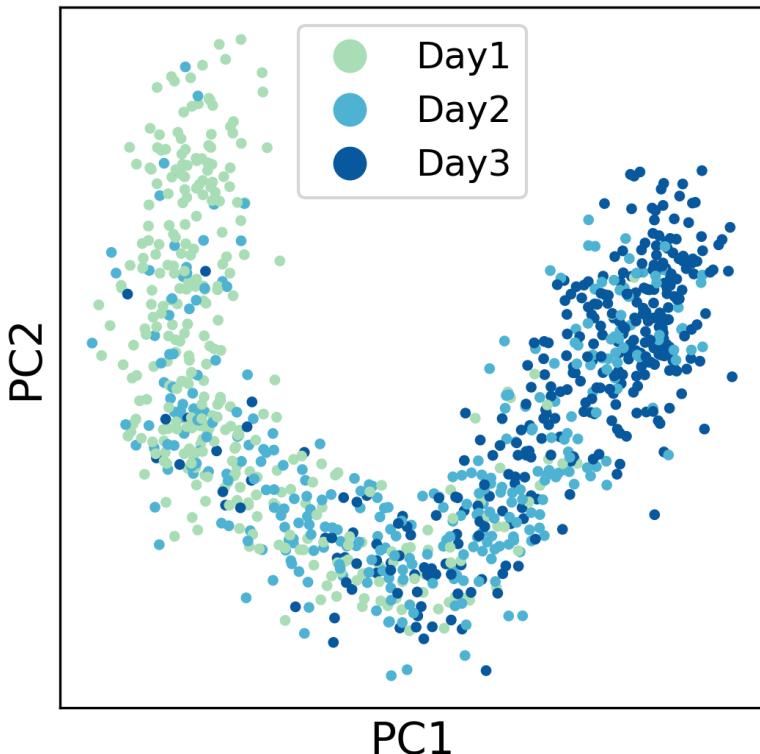


The goal of trajectory inference (TI) is to infer a temporal ordering of cells

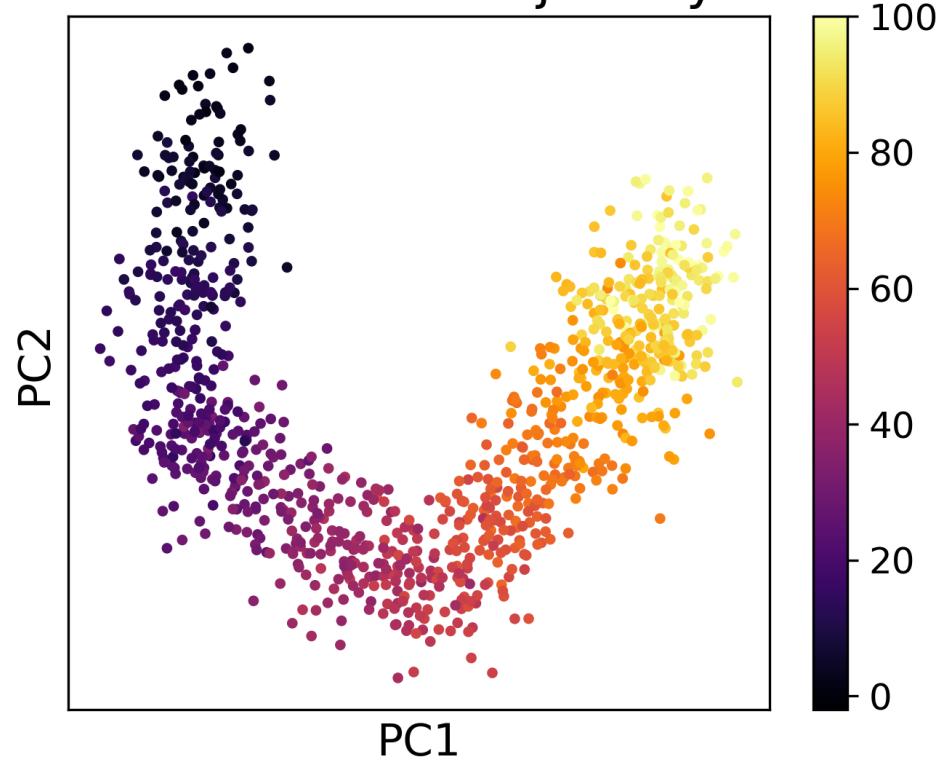


Learning pseudotime via graph walks

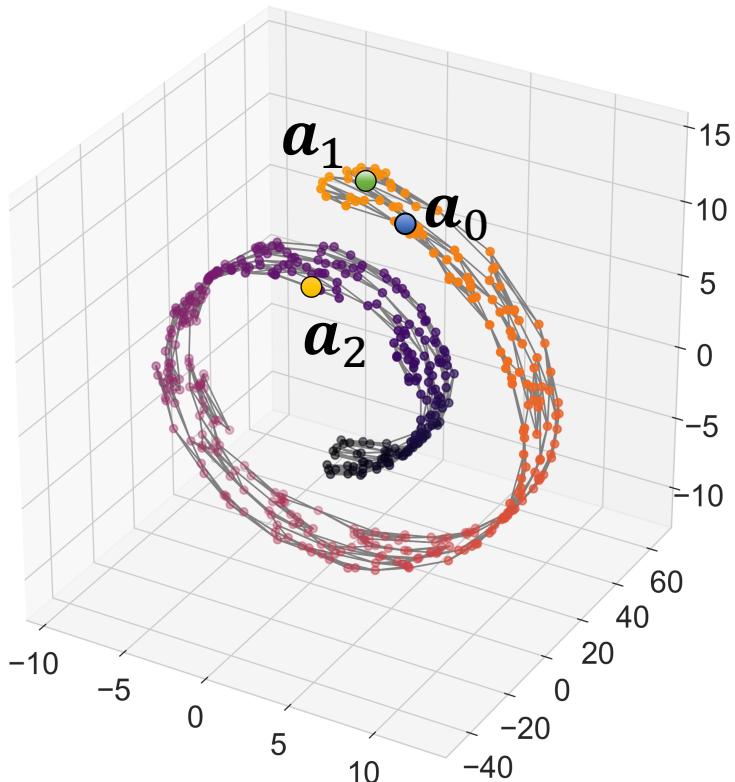
Day of sample collection



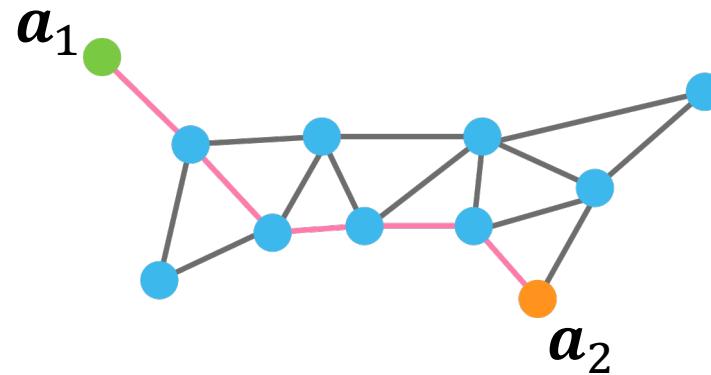
Growth truth trajectory



Graph walks approximate manifold distances



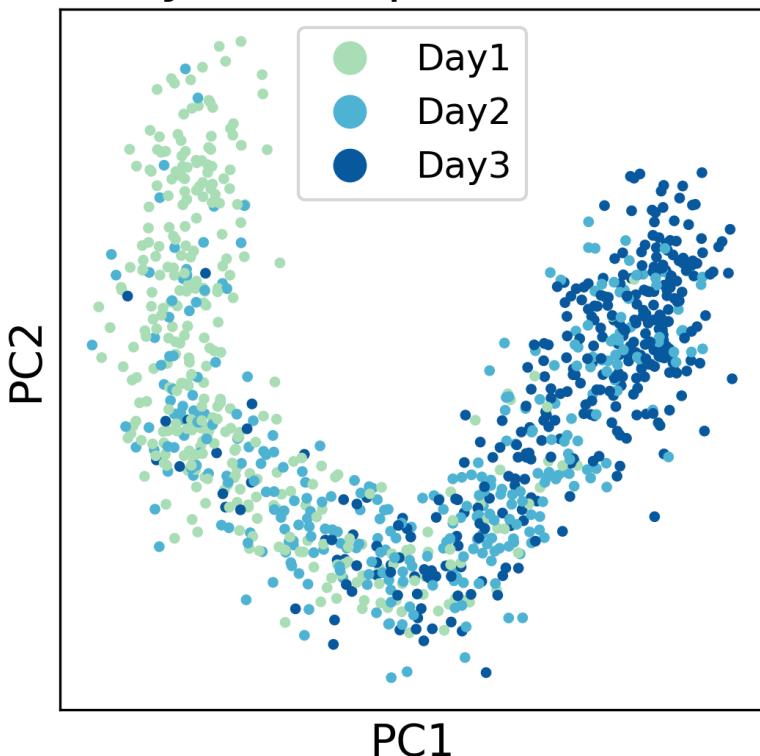
Shortest path between points



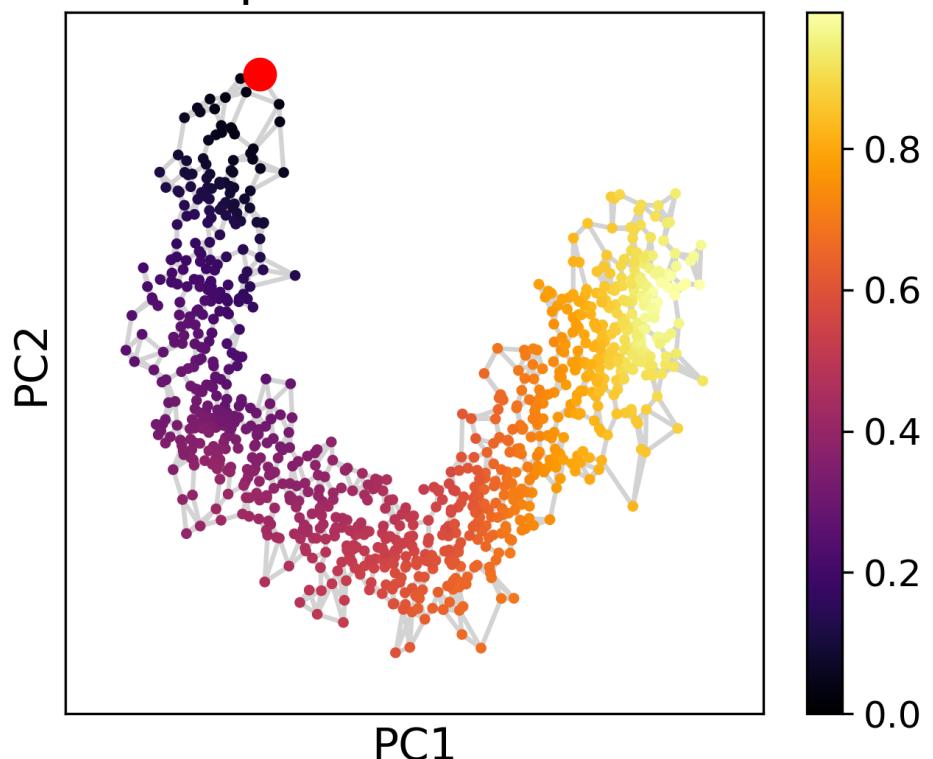
$$\begin{aligned} d_{geodesic}(a_1, a_2) &= \text{shortestpath}(a_1, a_2) \\ &= 5 \end{aligned}$$

Learning pseudotime via graph walks

Day of sample collection



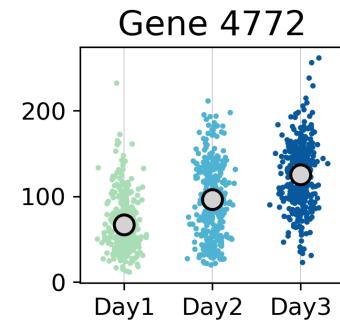
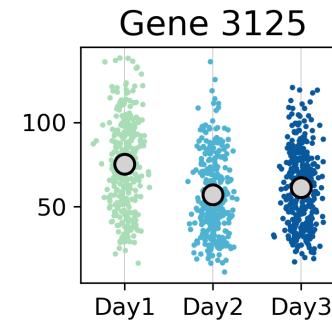
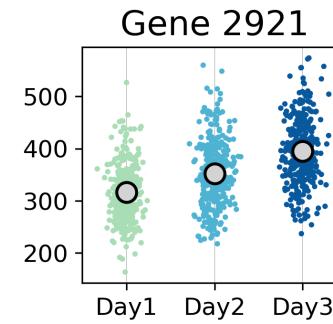
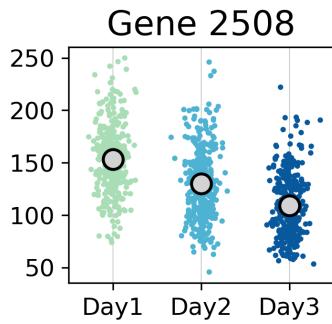
Graph walk distance



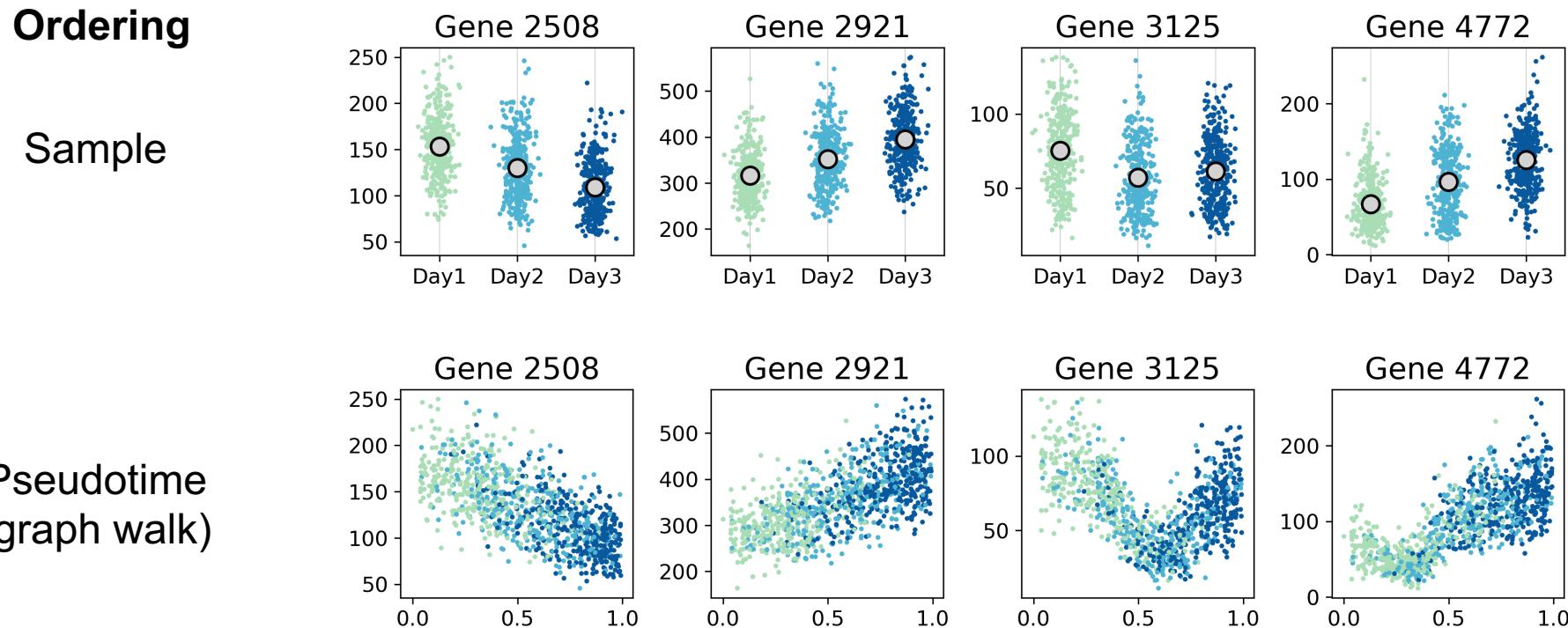
Pseudotime facilitates analysis of genes driving cellular progression

Ordering

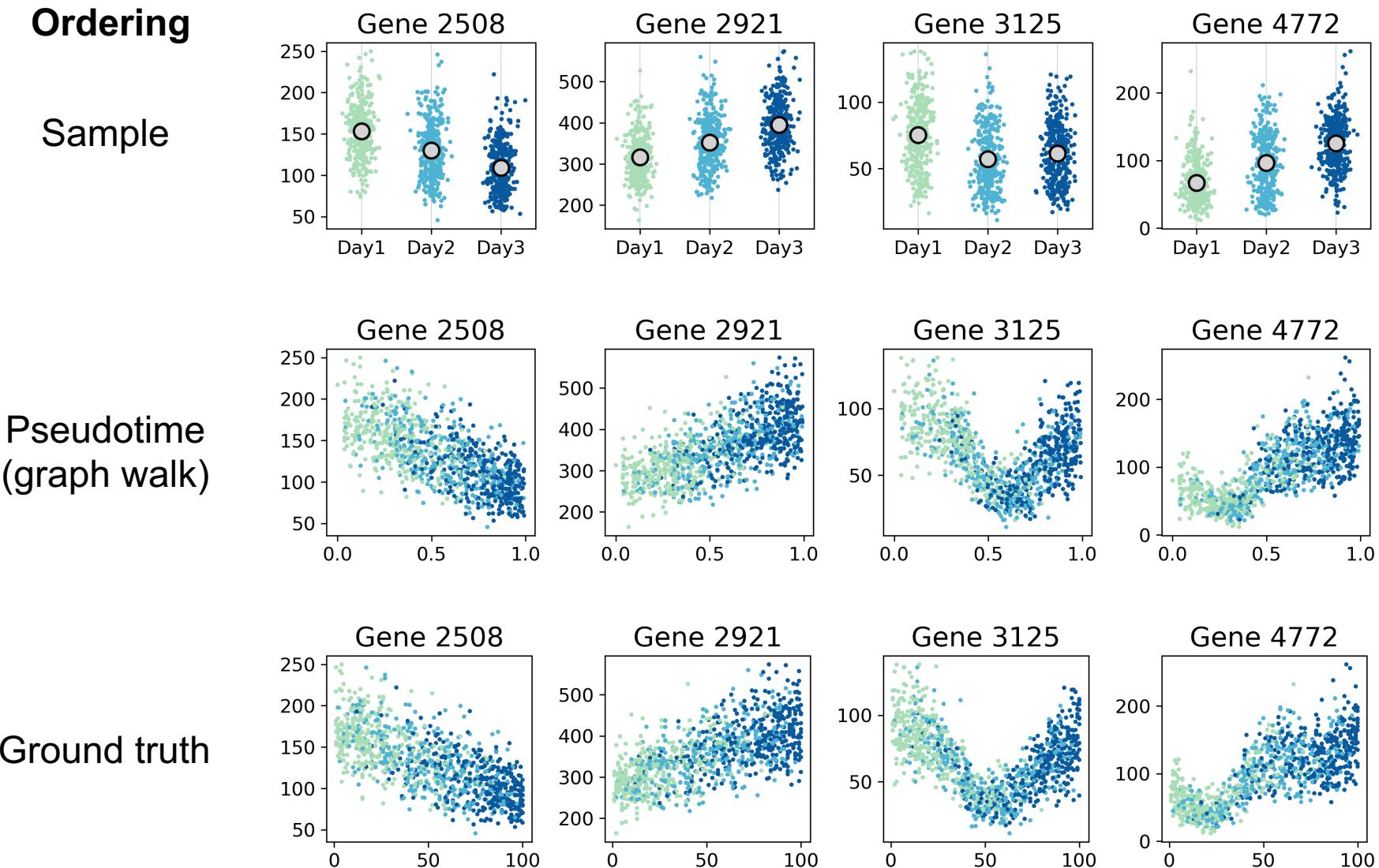
Sample



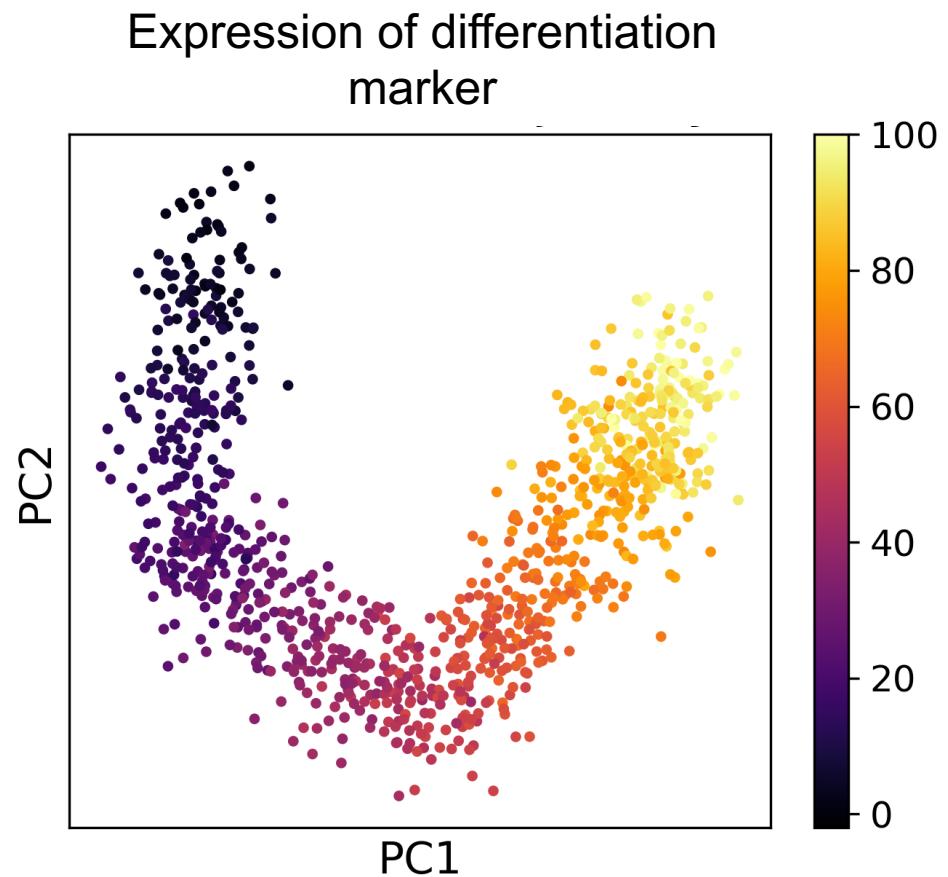
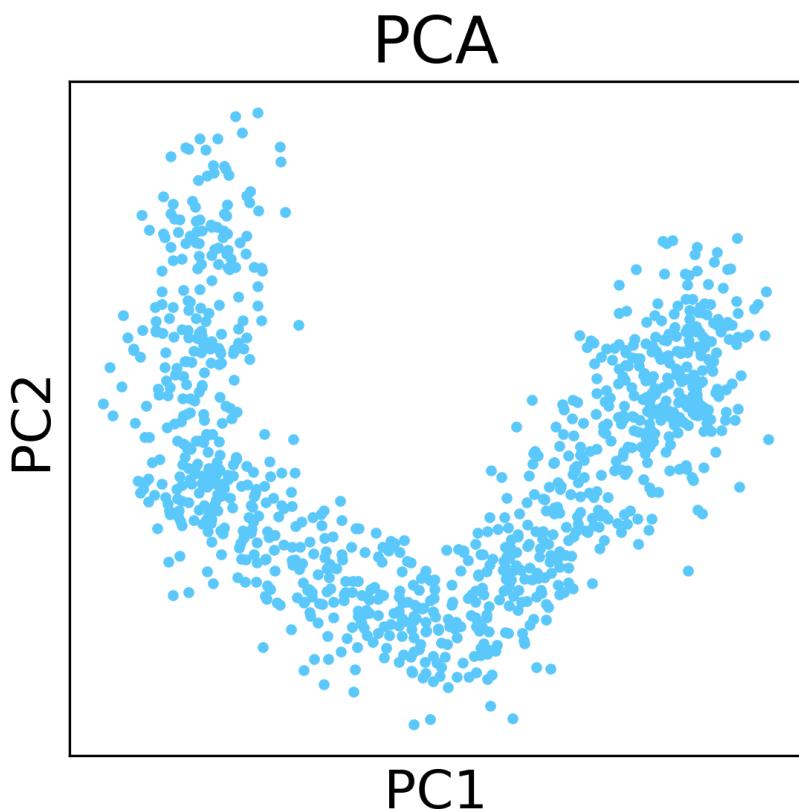
Pseudotime facilitates analysis of genes driving cellular progression



Pseudotime facilitates analysis of genes driving cellular progression



Note: you do not need multiple time points for pseudotime analysis



Diffusion pseudotime – from graph distance to random walks

Euclidean distance
between two vectors

$$\text{dpt}(x, y) = \| M(x, \cdot) - M(y, \cdot) \|, \quad M = \sum_{t=1}^{\infty} \tilde{T}^t.$$

Sum over
values of t

$$M = \sum_{t=1}^{\infty} \tilde{T}^t = (I - \tilde{T})^{-1} - I \text{ where } \tilde{T} = T - \psi_0 \psi_0^T,$$

Provides a
calculatable solution
to the infinite series

T is the random
walk matrix

ψ is used for eigenvectors
 ψ_0 represents starting
probabilities for T

I is the identity
matrix with 1's
on the diagonal

$$\begin{bmatrix} 1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & 1 \end{bmatrix}$$

Diffusion pseudotime – from graph distance to random walks

Euclidean distance between two vectors

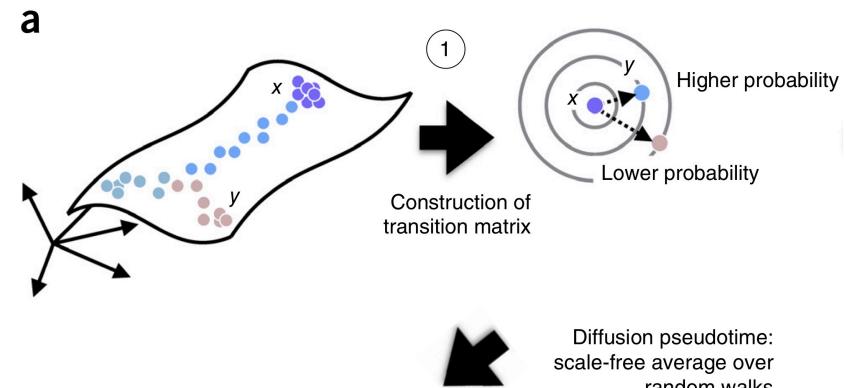
$$\text{dpt}(x, y) = \| M(x, \cdot) - M(y, \cdot) \|, \quad M = \sum_{t=1}^{\infty} \tilde{T}^t.$$

$$M = \sum_{t=1}^{\infty} \tilde{T}^t = (I - \tilde{T})^{-1} - I \text{ where } \tilde{T} = T - \Psi_0 \Psi_0^T,$$

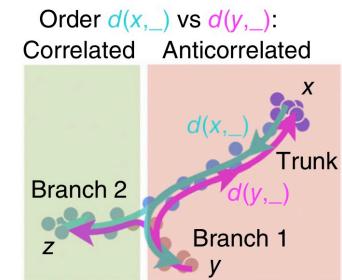
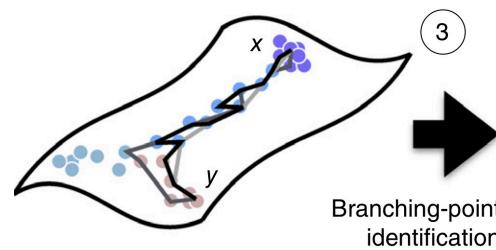
Provides a calculatable solution to the infinite series

Sum over values of t

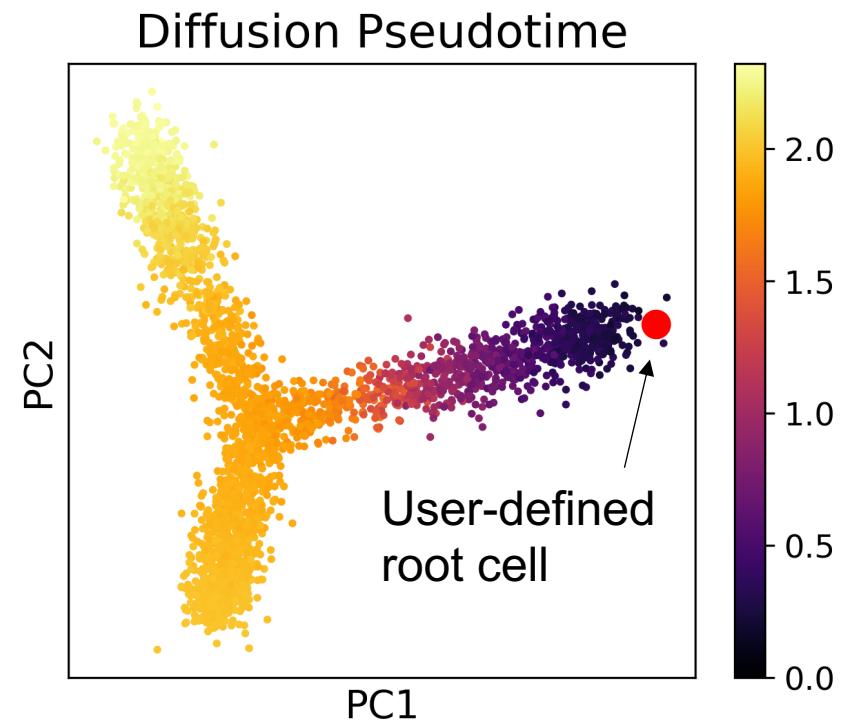
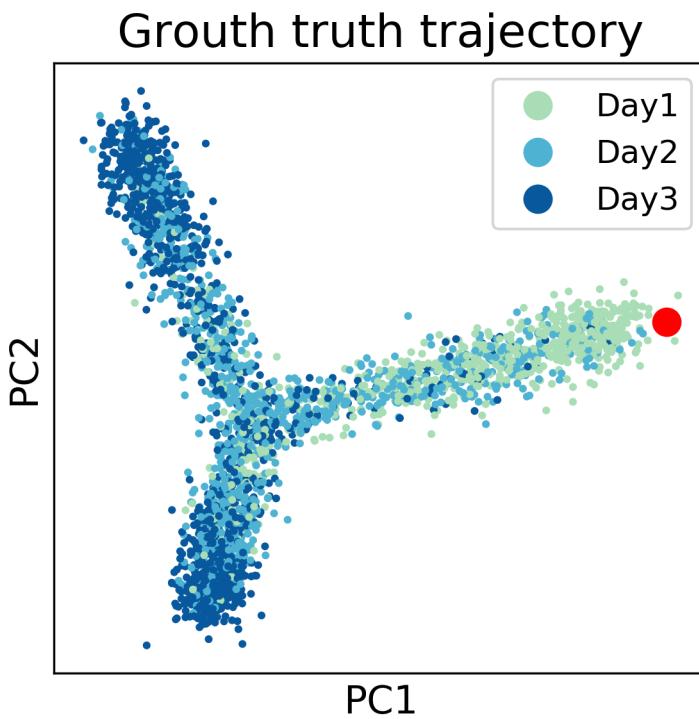
Transition probabilities give data geometry



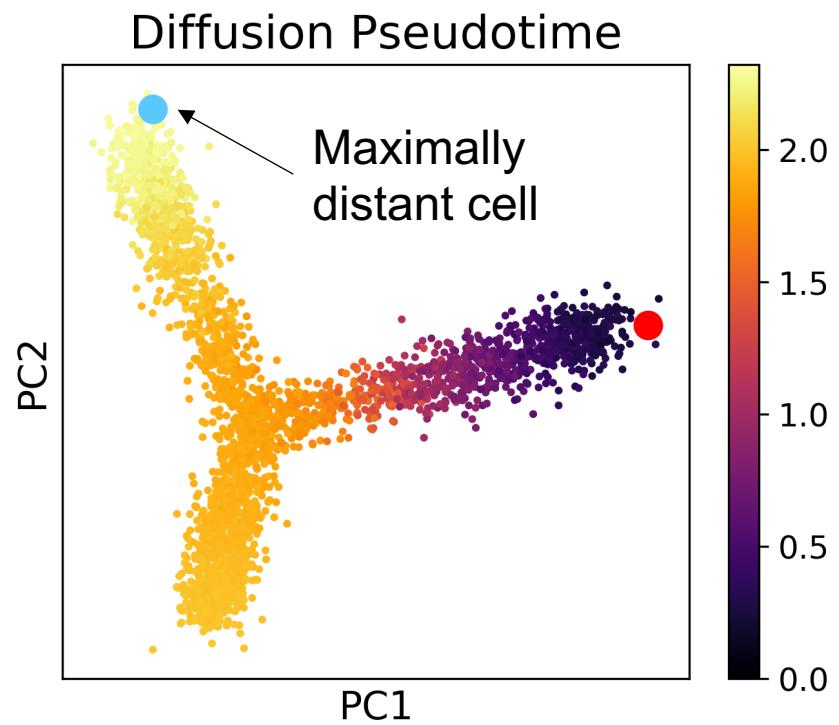
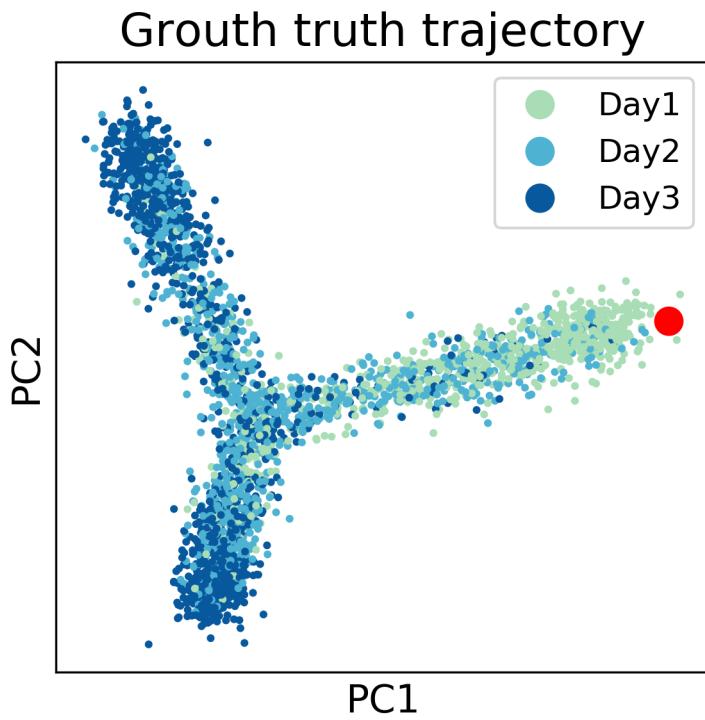
Correlation reveals branches



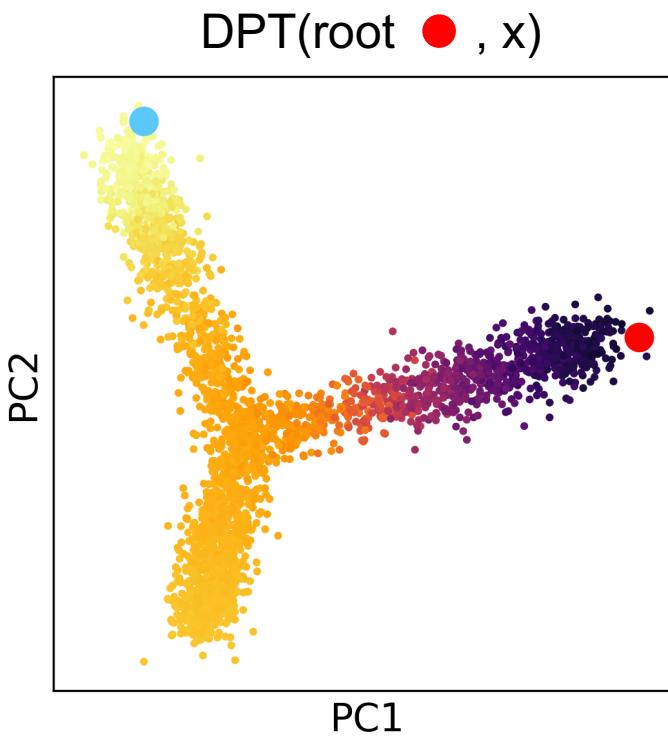
Using DPT to detect branches



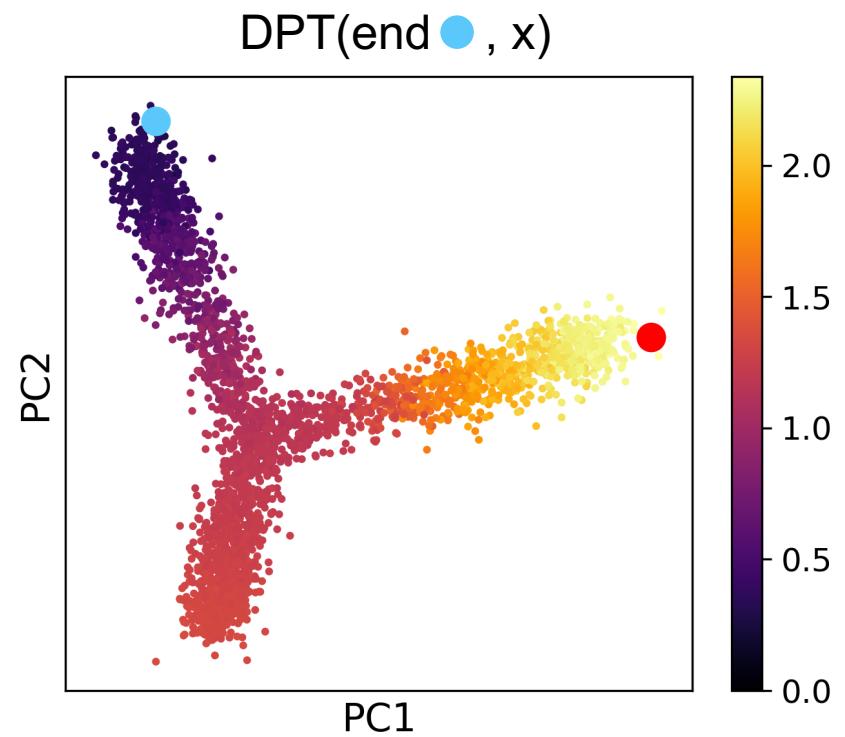
Using DPT to detect branches



Using DPT to detect branches



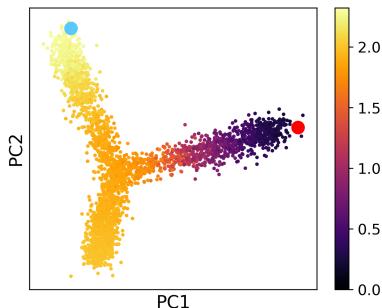
“Forward pseudotime”



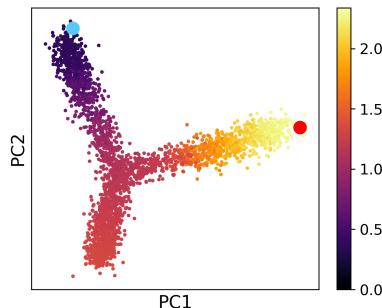
“Backward pseudotime”

Using DPT to detect branches

DPT(root ● , x)



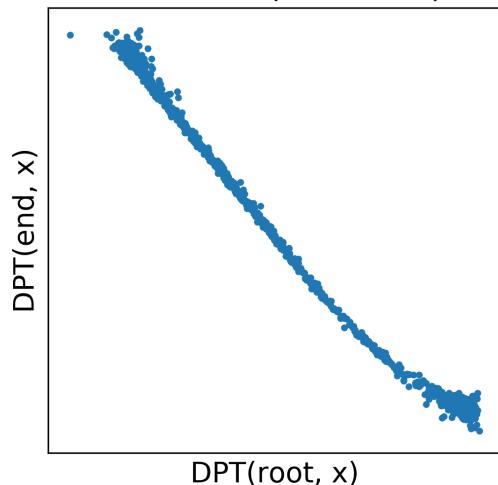
DPT(end ● , x)



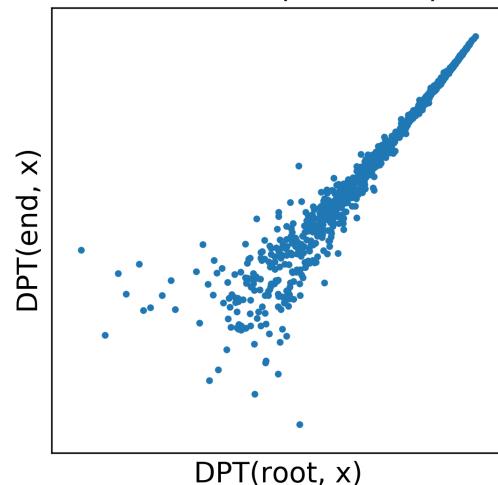
Cells on the same “branch” as the root and end cell have negative DPT correlation.

Cells on different branches have positive DPT correlation

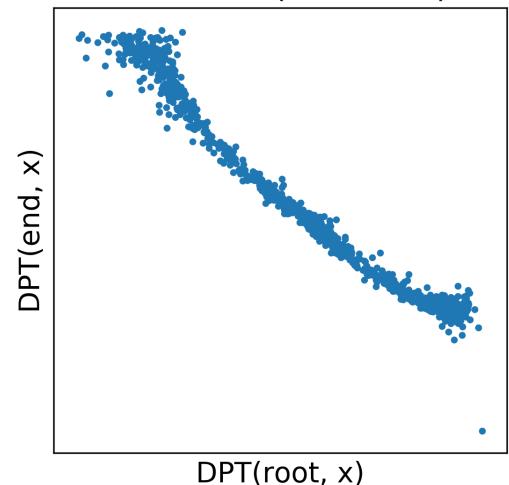
Branch 1 ($r = -0.99$)



Branch 2 ($r = 0.96$)

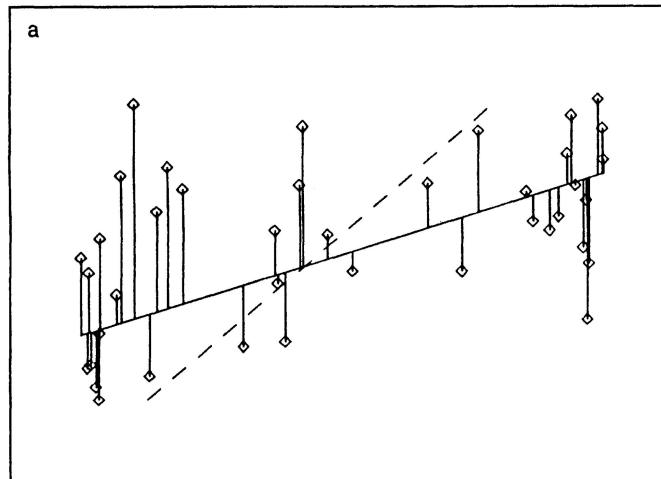


Branch 3 ($r = -0.99$)

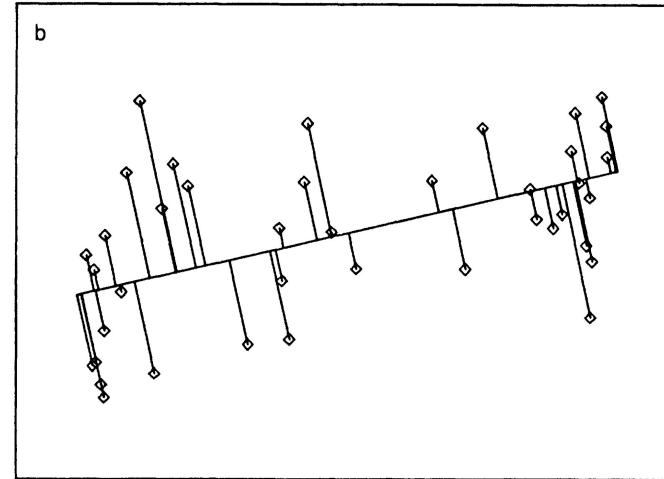


Principal curves – non-linear PCA

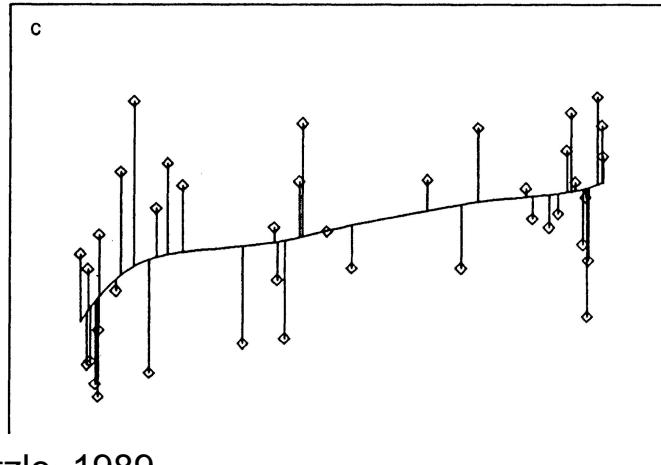
Linear Regression



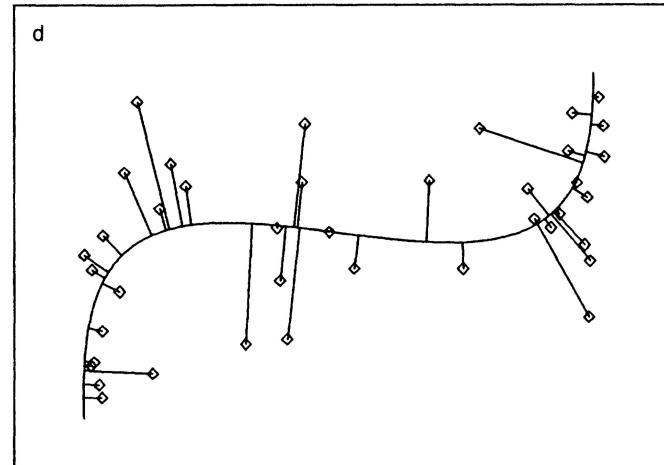
PCA



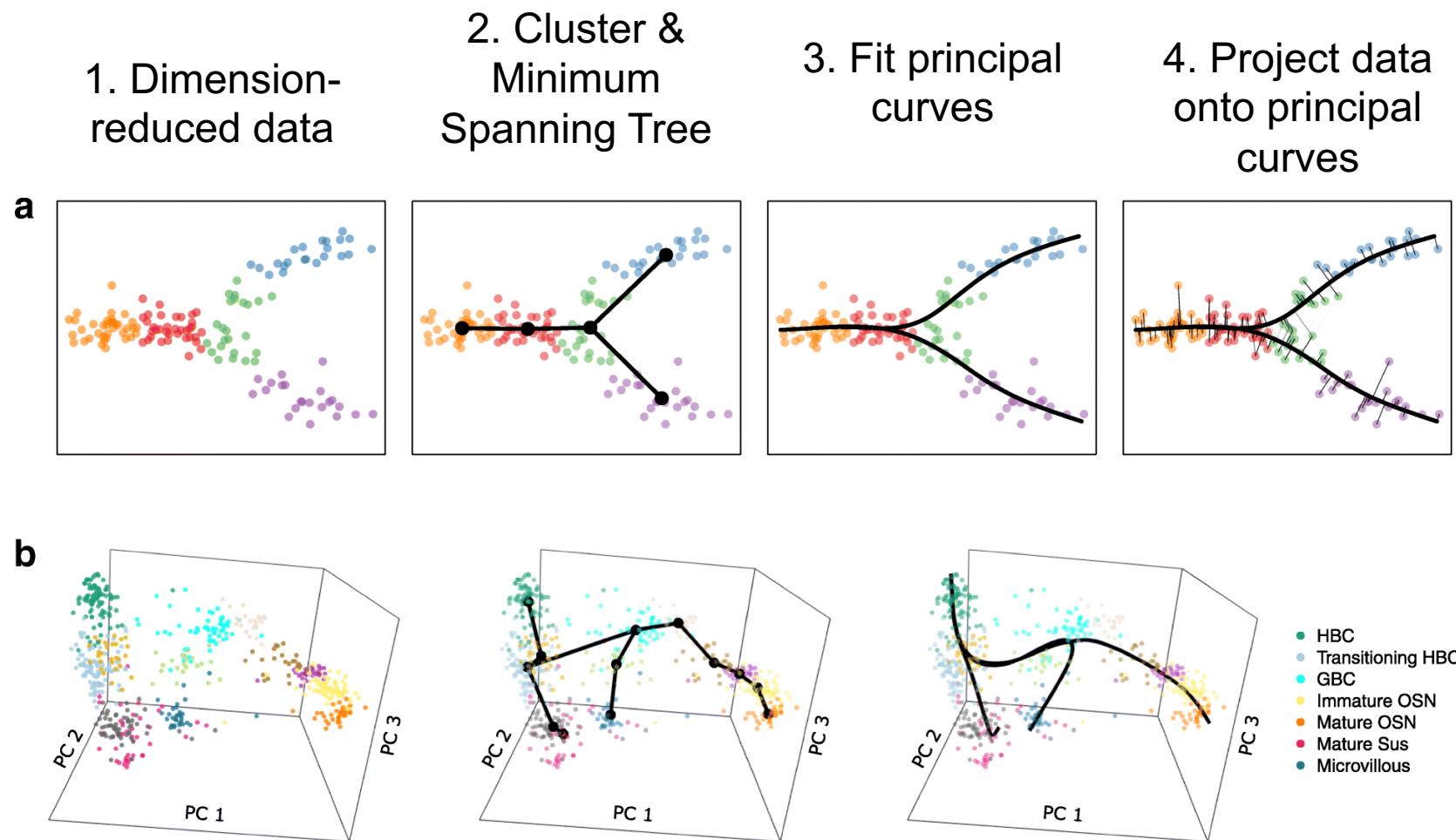
Non-linear Regression



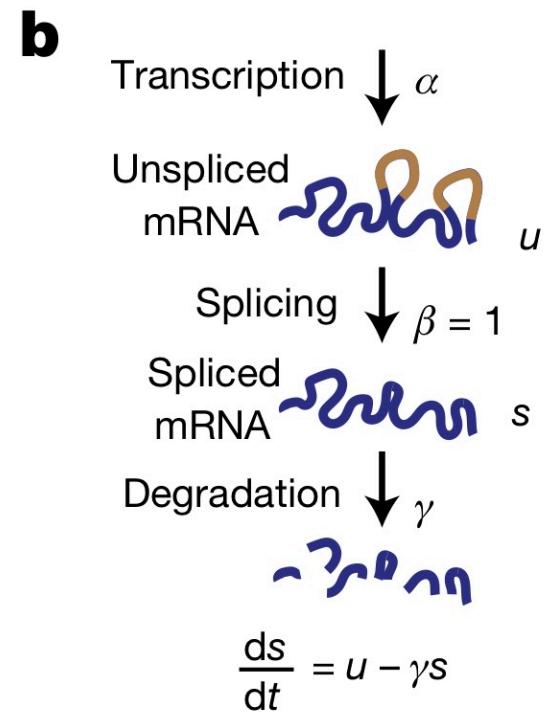
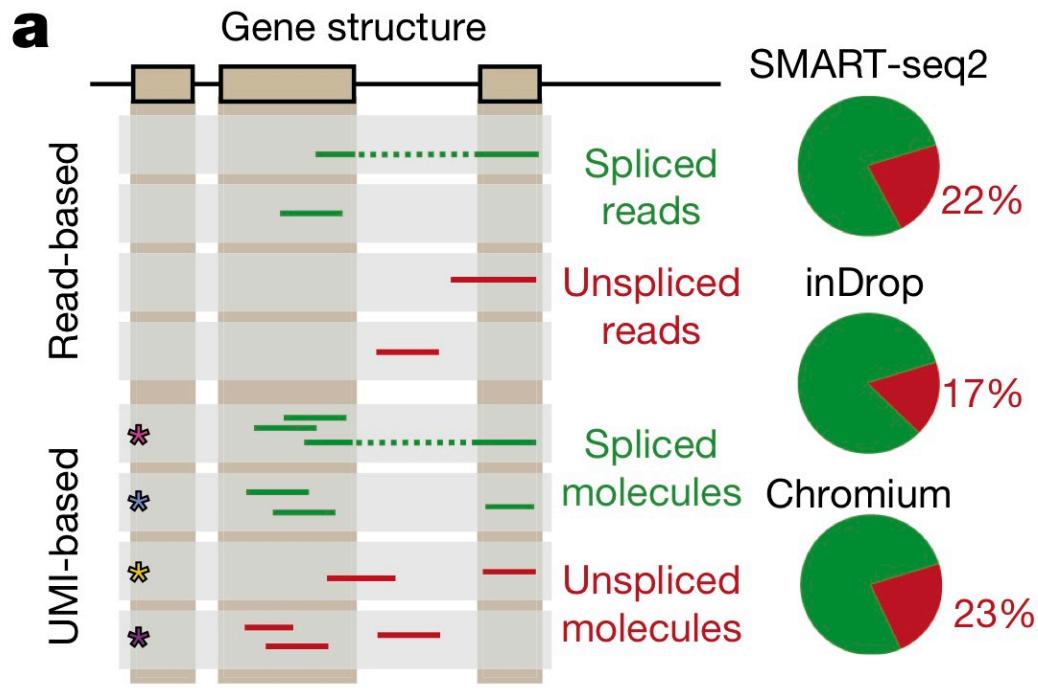
Principal Curves



Slingshot - Street et al. 2018

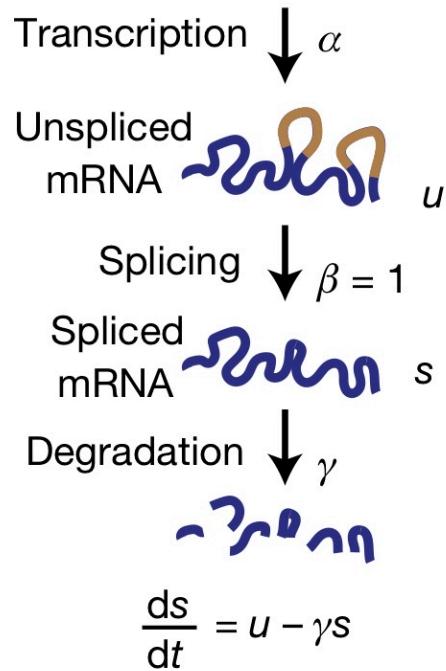


RNA Velocity: using gene splicing genes to infer future cell state

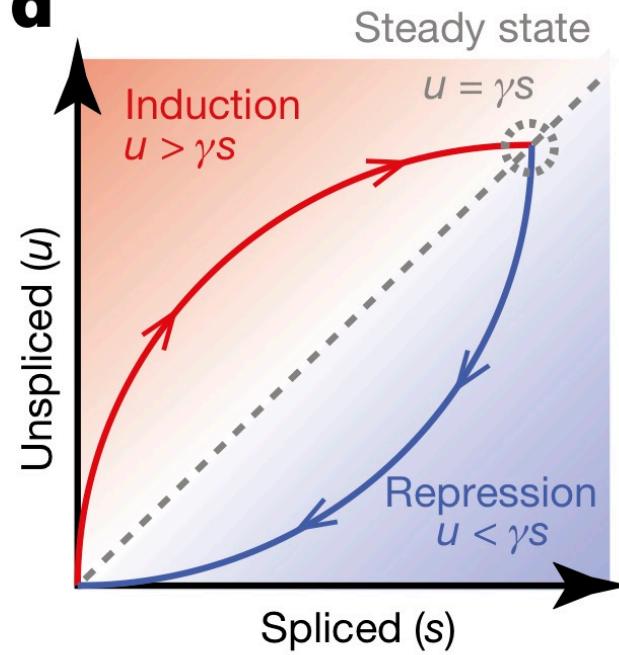


RNA Velocity: using gene splicing genes to infer future cell state

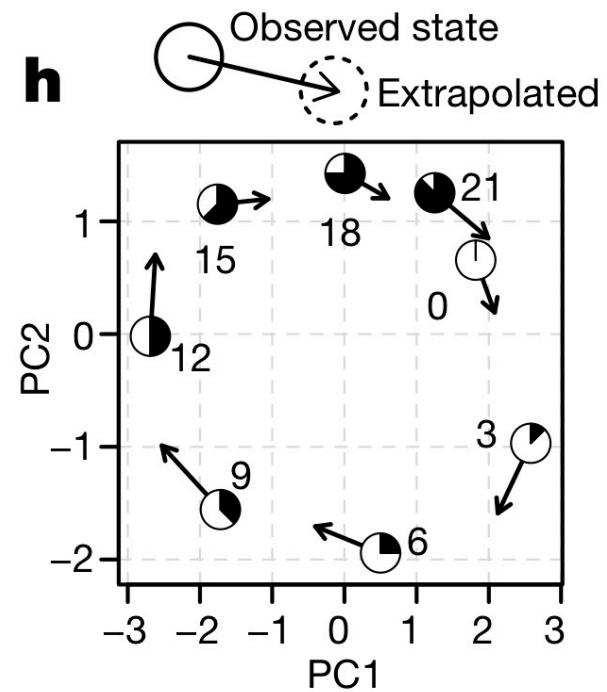
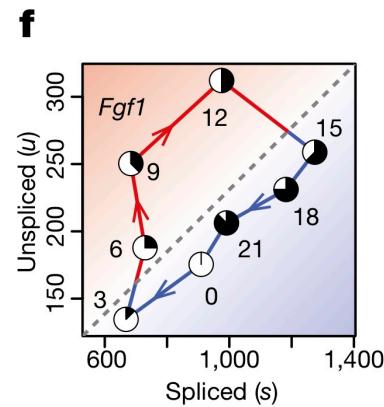
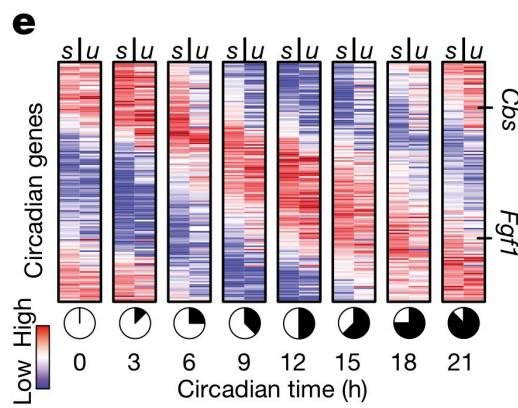
b



d

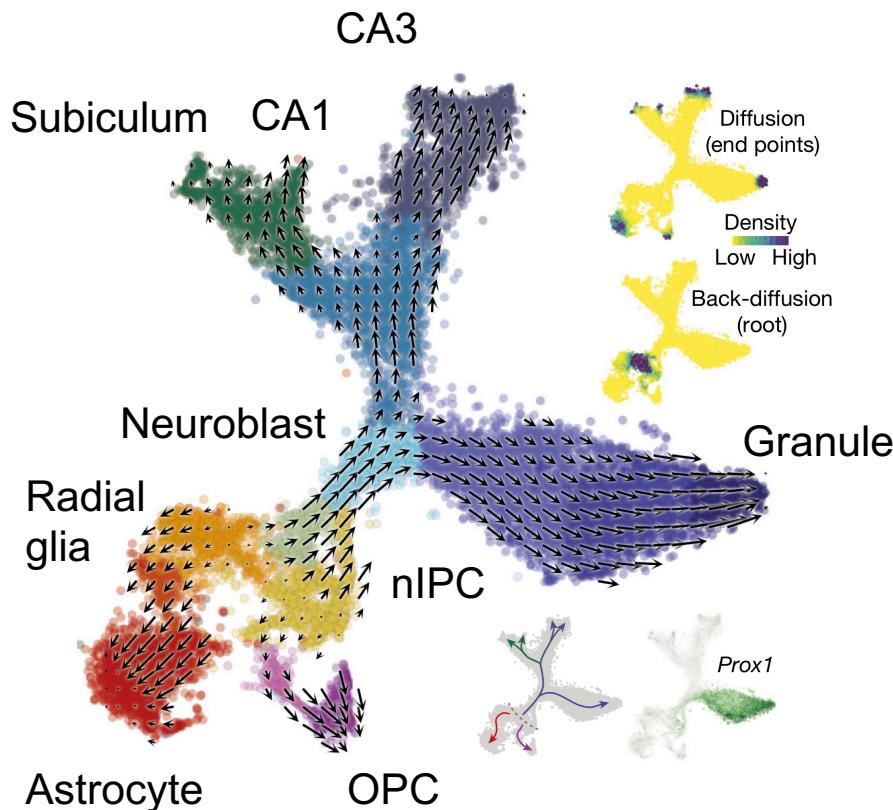


RNA velocity correctly infers circadian rhythm in mouse liver

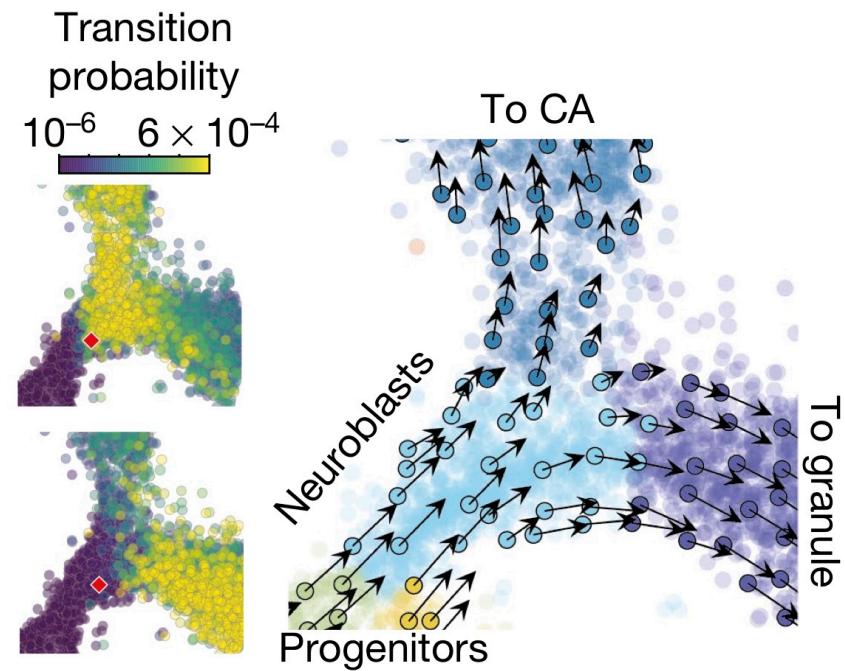
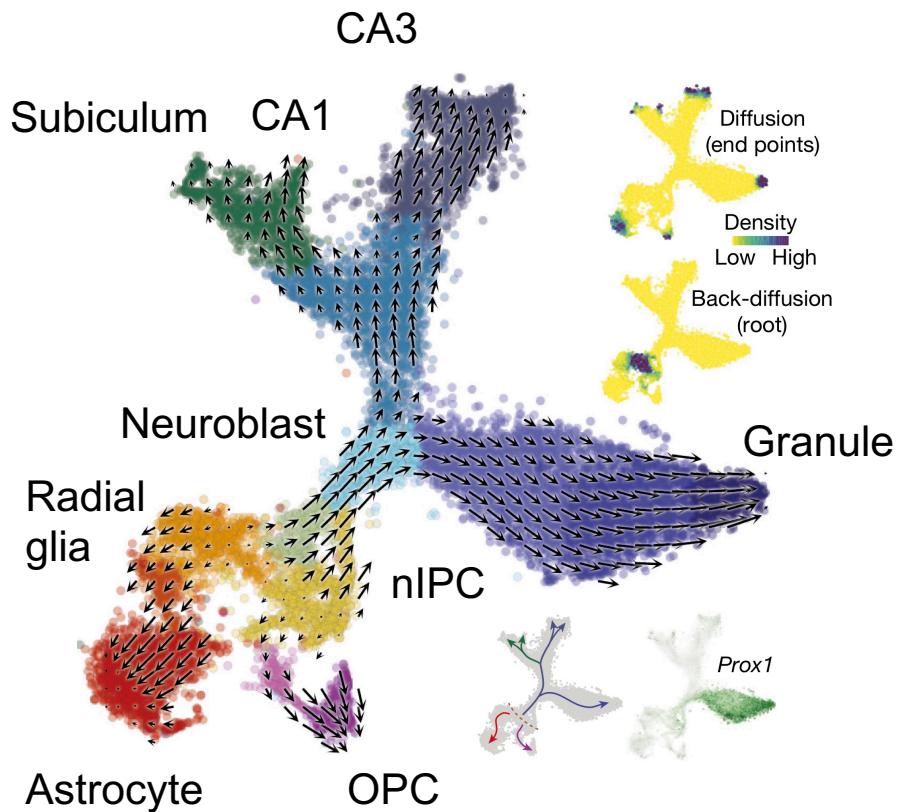


mRNA was measured in bulk over 24 hours in the mouse liver

Neural fate decisions in the mouse hippocampus



Neural fate decisions in the mouse hippocampus

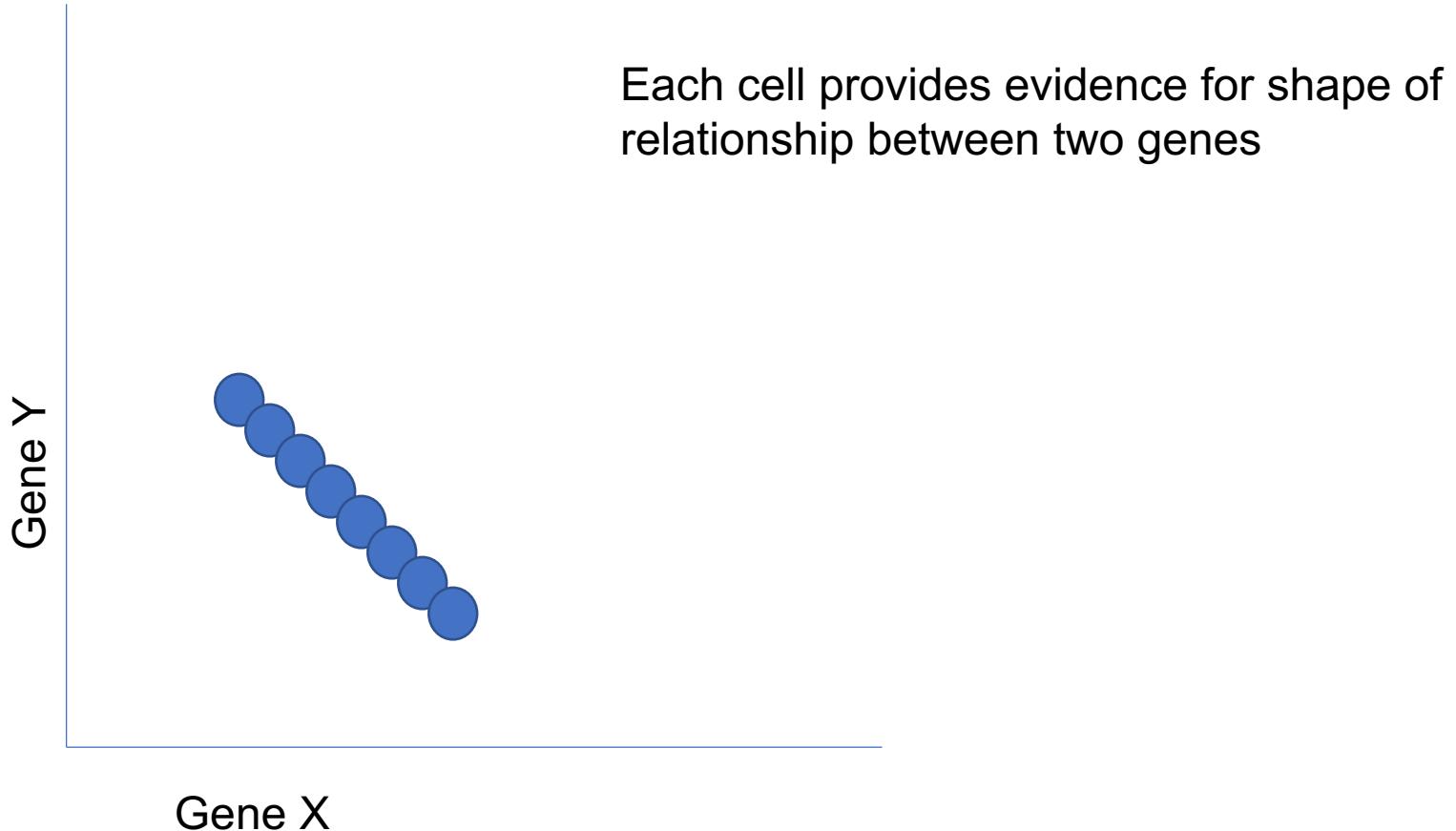


Conclusions

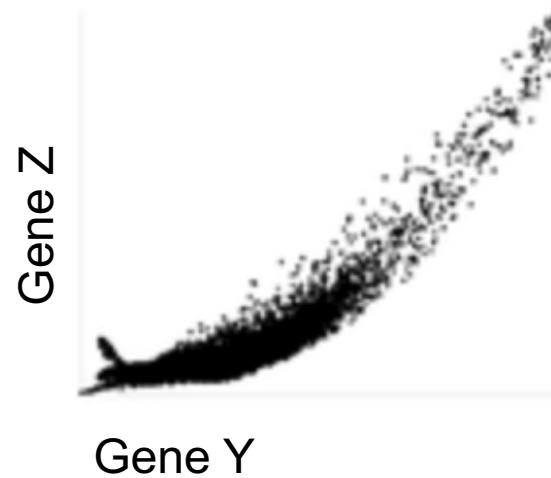
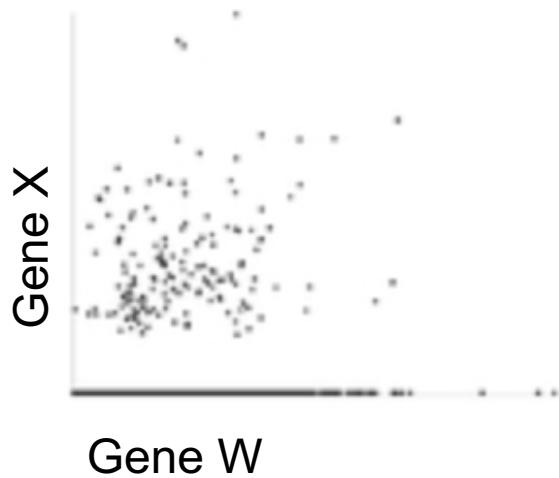
1. Pseudotime identifies an ordering of cells that match the developmental progression of gene expression
2. Pseudotime can be identified from a single timepoint
3. Methods for inferring pseudotime use graph walks or identify axes through the data
4. A comprehensive comparison of trajectory inference methods is available as dynverse.org

Learning gene-gene relationships

Single Cell Data



Gene-gene Relationships



When poll is active, respond at **PollEv.com/yaleml**

Text **YALEML** to **22333** once to join

How do you know that the second pair might have a relationship (interacting, coregulatory), but not the first?

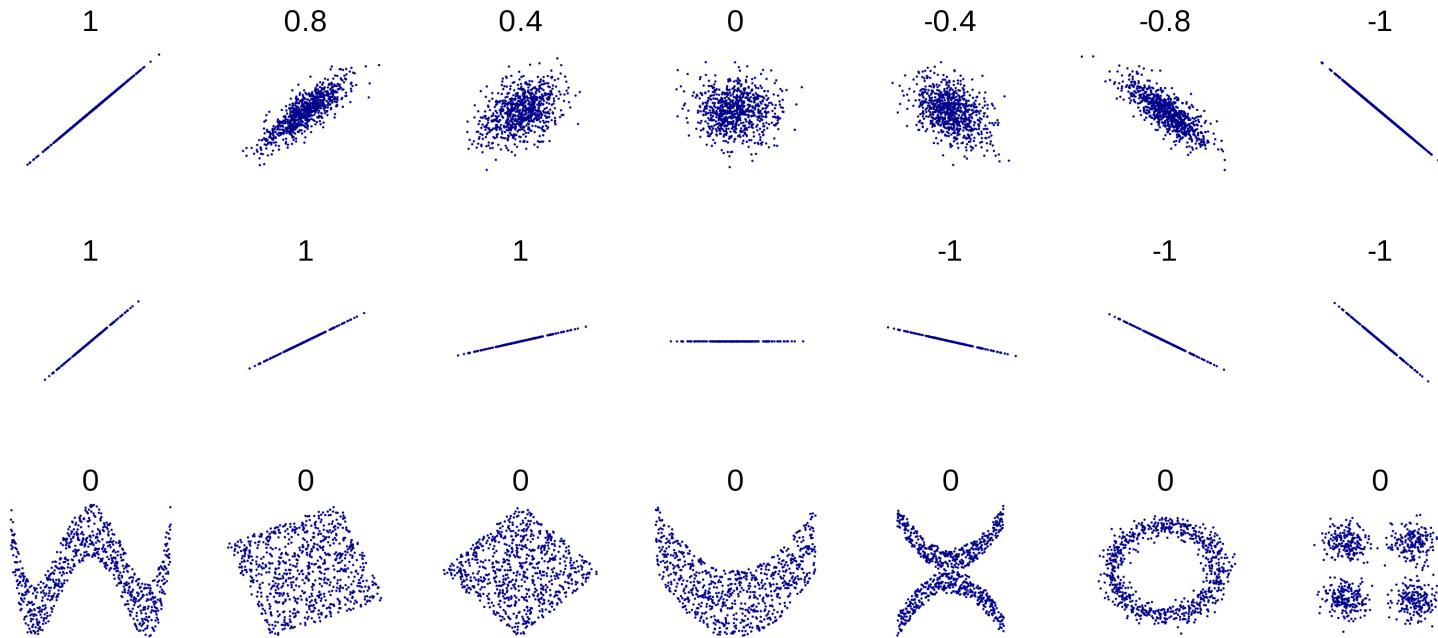
Intuition Behind Relationships

- When X changes Y should change
- Does the change have to be proportional?
- Does the change have to be in the same direction?
- No



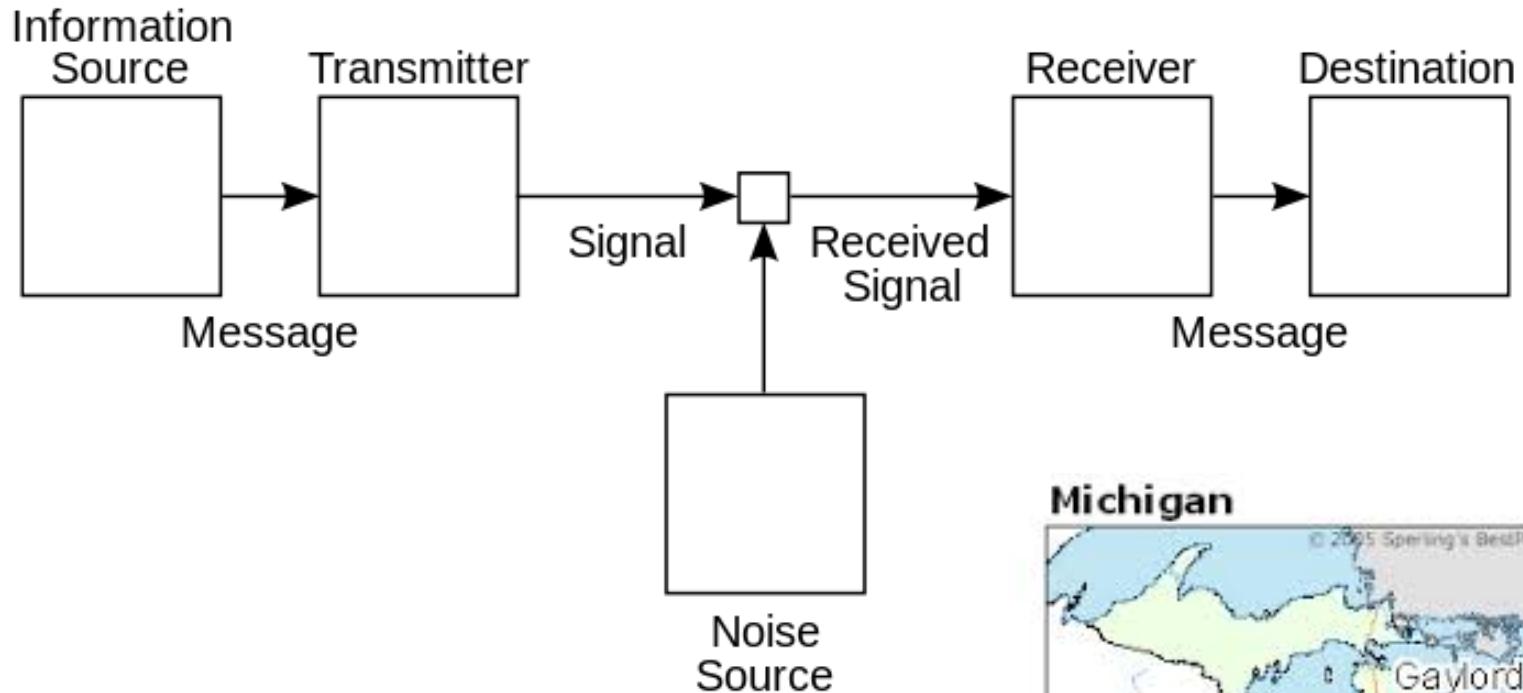
Correlation as a measure

$$\rho_{X,Y} = \text{corr}(X, Y) = \frac{\text{cov}(X, Y)}{\sigma_X \sigma_Y} = \frac{\mathbb{E}[(X - \mu_X)(Y - \mu_Y)]}{\sigma_X \sigma_Y}$$



Problem: checks only for linear relationships

Information Theory



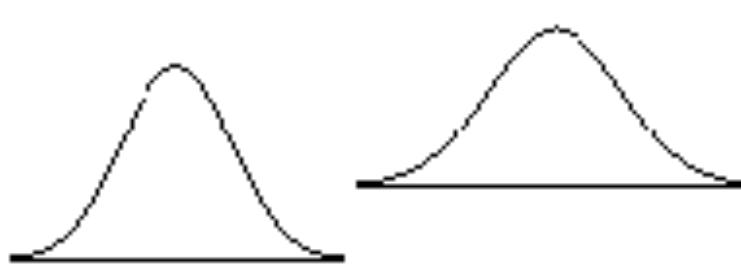
“A Mathematical Theory of Communication”



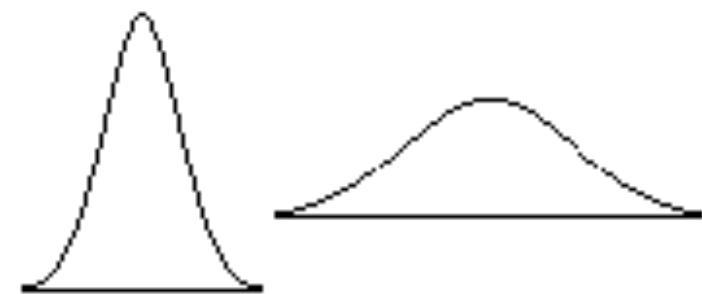
Mutual Information: Shape Agnostic

- Mutual information is a metric that comes from information theory
- Do two variables have “information” on each other?
- Information is defined as the opposite of uncertainty
- If a variable X has information on another variable Y then knowing X reduces *the uncertainty of Y*

Uncertainty measured by entropy



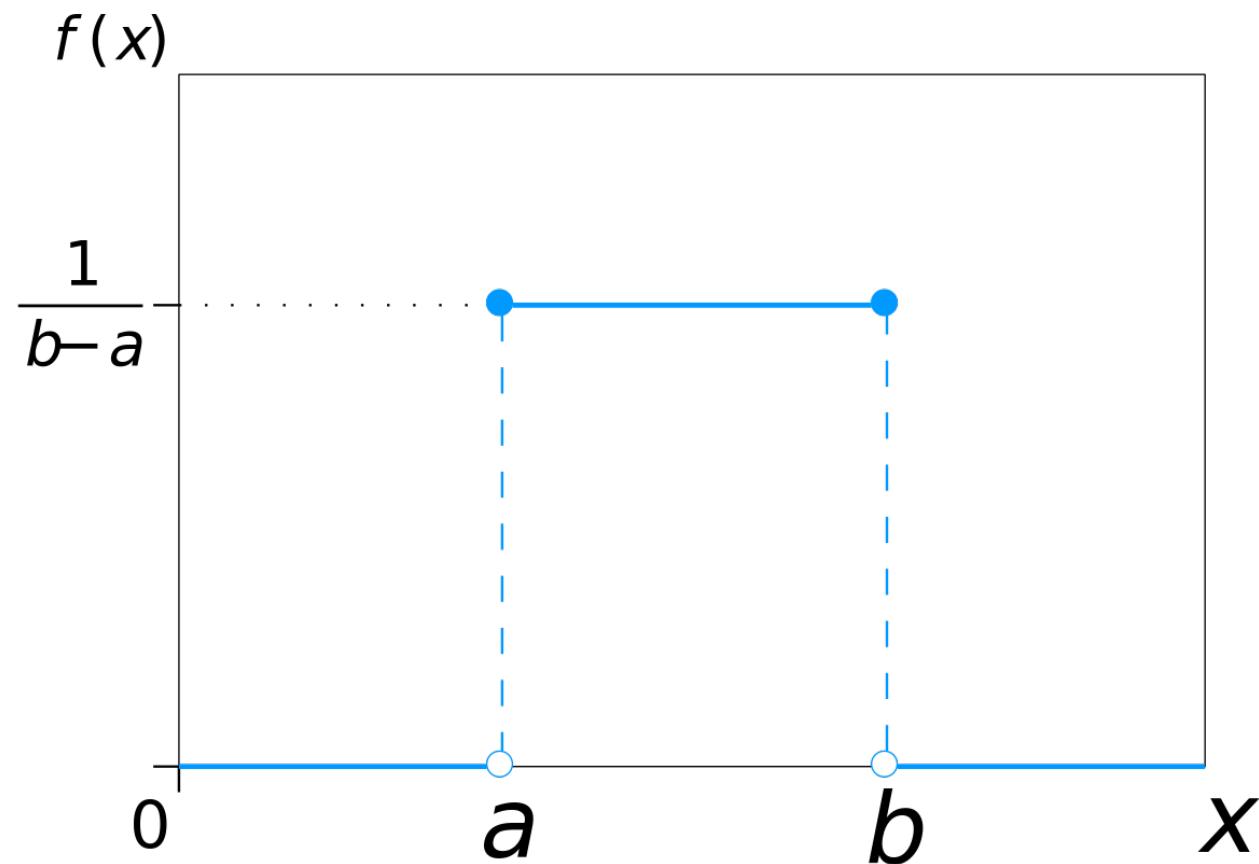
Peaky narrow ones have low entropy



Wide flat distributions have high entropy

Peaky narrow ones have low entropy

Uniform has highest entropy



Entropy Measured in Bits

- Fair coin has 1 bit of information



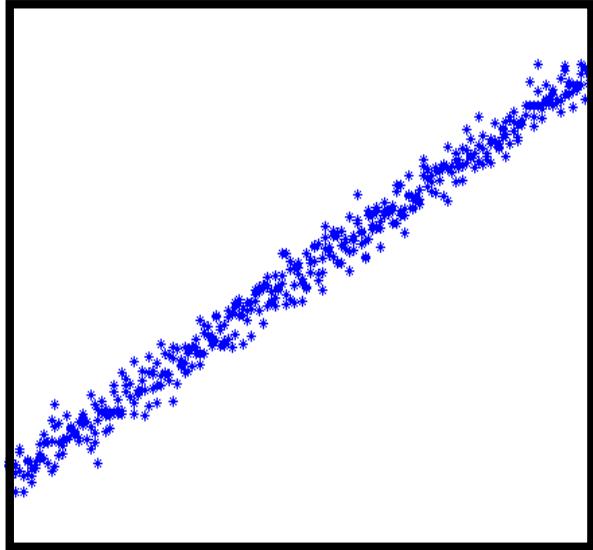
$$H(X) = - \sum_i p(x_i) \log(p(x_i))$$

$$\begin{aligned} P(\text{heads}) &= \frac{1}{2}, \quad P(\text{tails}) = \frac{1}{2} \\ H(\text{coin}) &= -\left(\frac{1}{2}\right) \log\left(\frac{1}{2}\right) - \left(\frac{1}{2}\right) \log\left(\frac{1}{2}\right) = 1 \end{aligned}$$

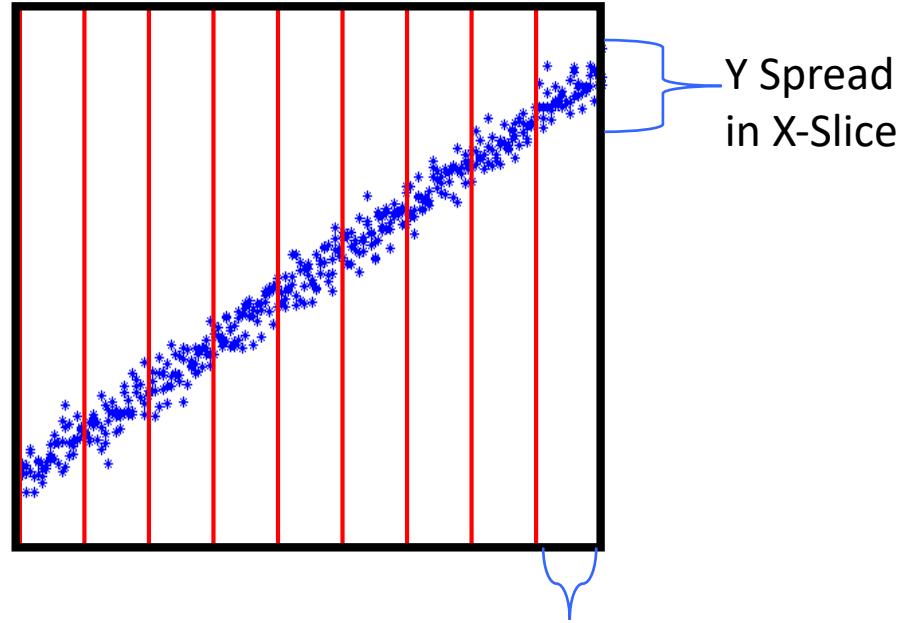
Entropy Properties

- Maximum at uniform distribution – because here each outcome is equally likely
- Symmetric with respect to probabilities – flipping probabilities of events should not change information
- Continuous – small change in probabilities should result in small change in entropy
- Additive – Independent sources of information should add

Entropy



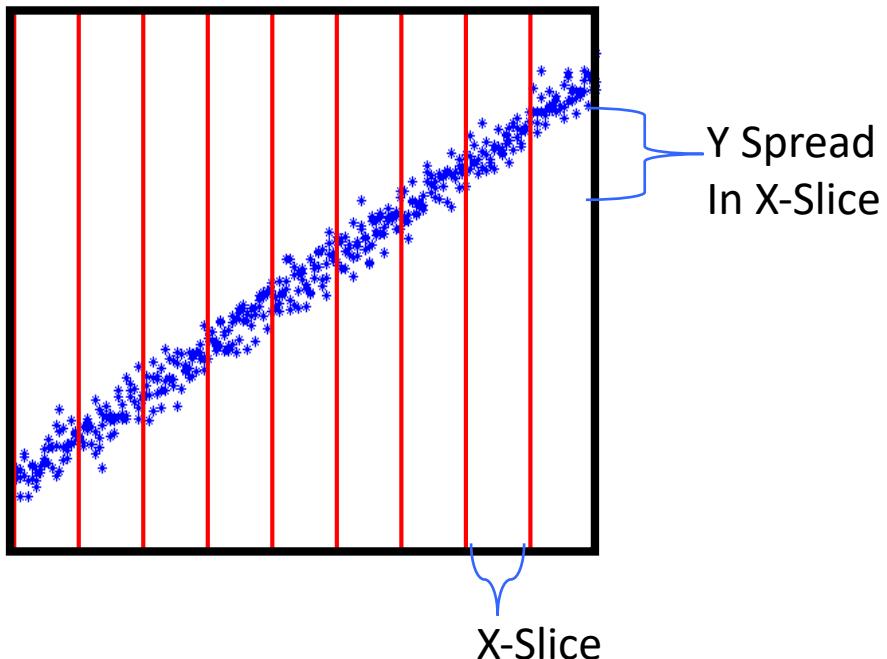
Y Spread



X-Slice

$$-\sum_{i=1}^n P(x_i) \log_b P(x_i),$$

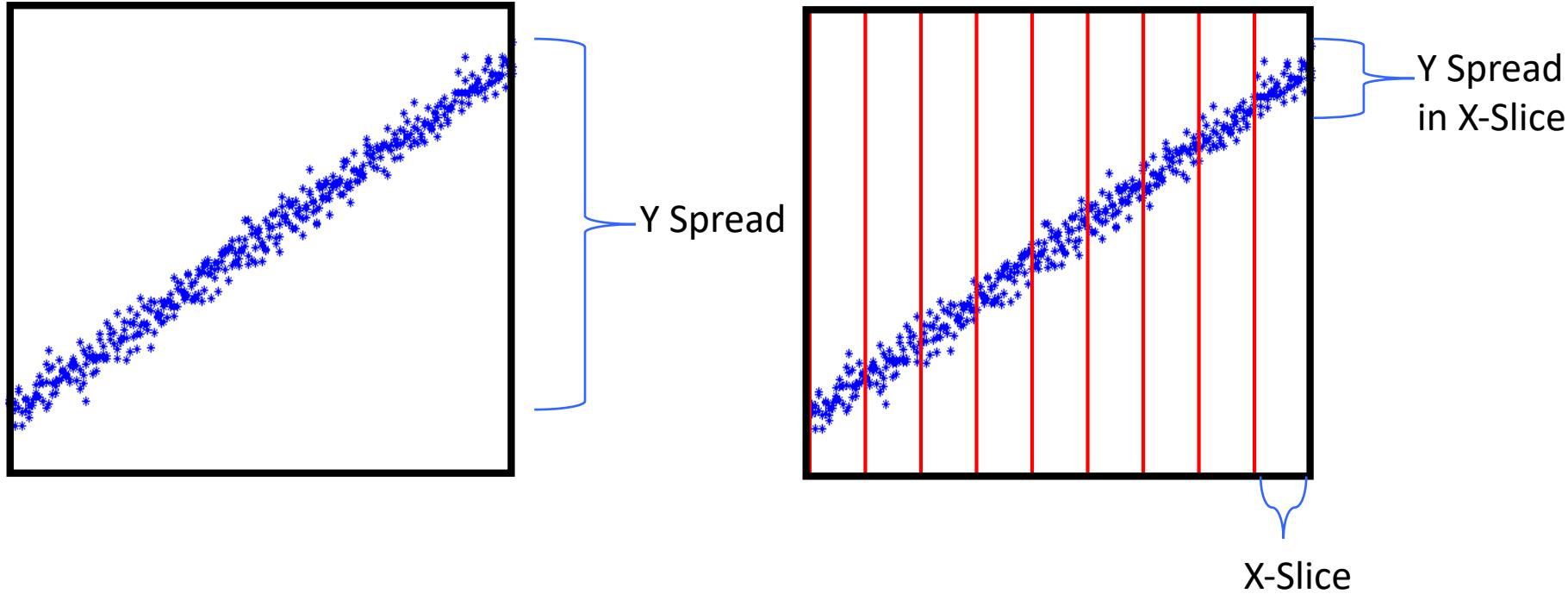
Conditional Entropy



$$H(Y|X) \leq H(Y)$$

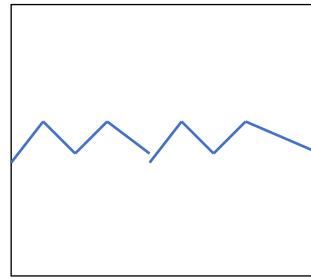
Measure of Uncertainty in
a random variable
given knowledge about
another variable

Mutual Information

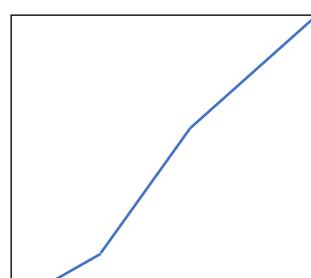
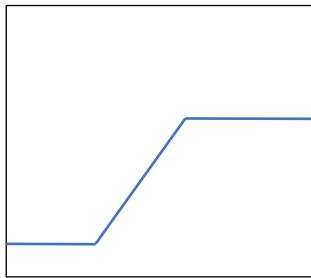


$$\text{Mutual Information: } I(X,Y) = H(Y) - H(Y|X)$$

Trends with High and Low MI



Low MI



High MI

Another Definition

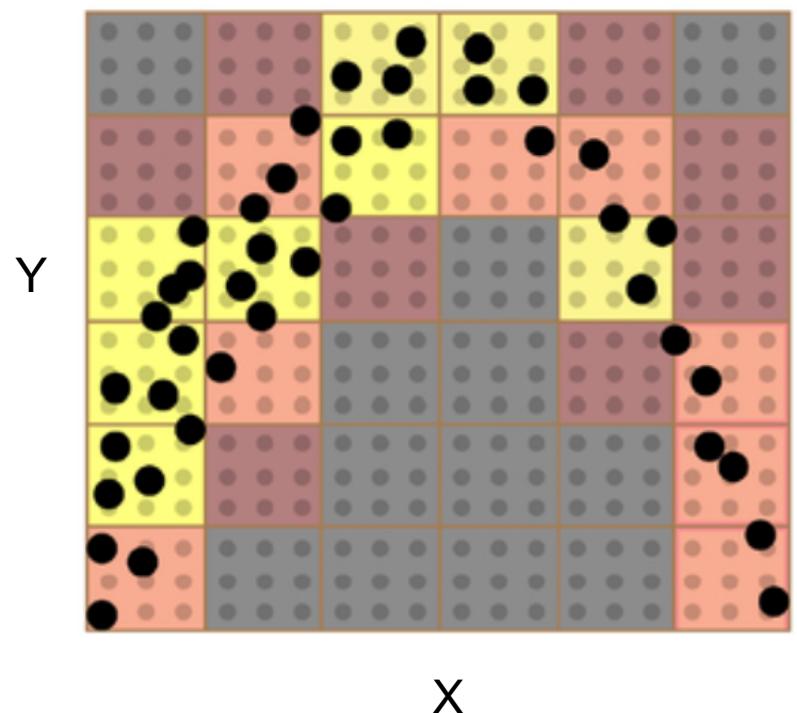
- Compares joint to marginal distributions

$$I(X; Y) = \sum_{y \in Y} \sum_{x \in X} p(x, y) \log \left(\frac{p(x, y)}{p(x) p(y)} \right)$$

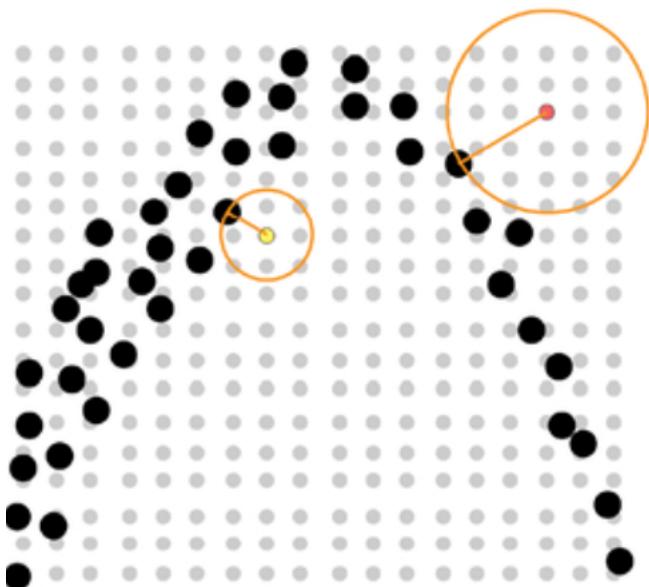
- Equivalent definitions

Computing MI ($X \rightarrow Y$) in Data

- Step 1: Compute probability density $P(Y)$ – Divide the data into bins and compute probability of datapoint falling into the bin



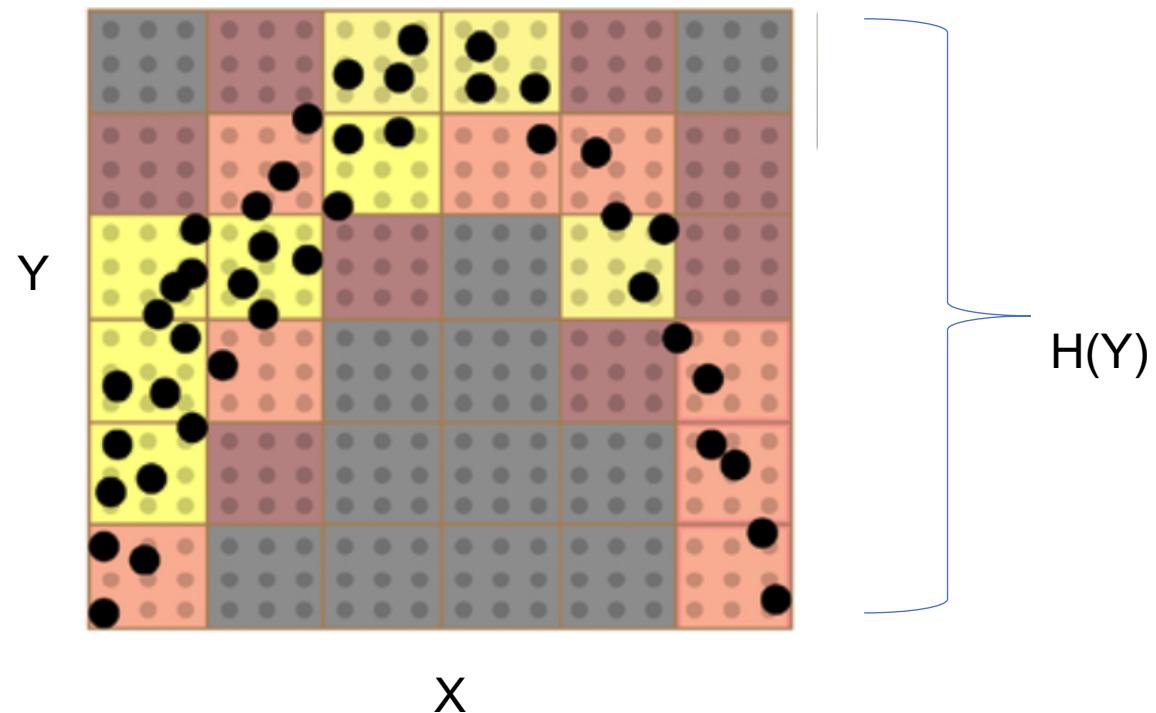
Density estimation



- Computing a probability density from data is called density estimation
- Kernel density estimation is a popular approach
- KNN based density estimation often done on data with affinity graph structure

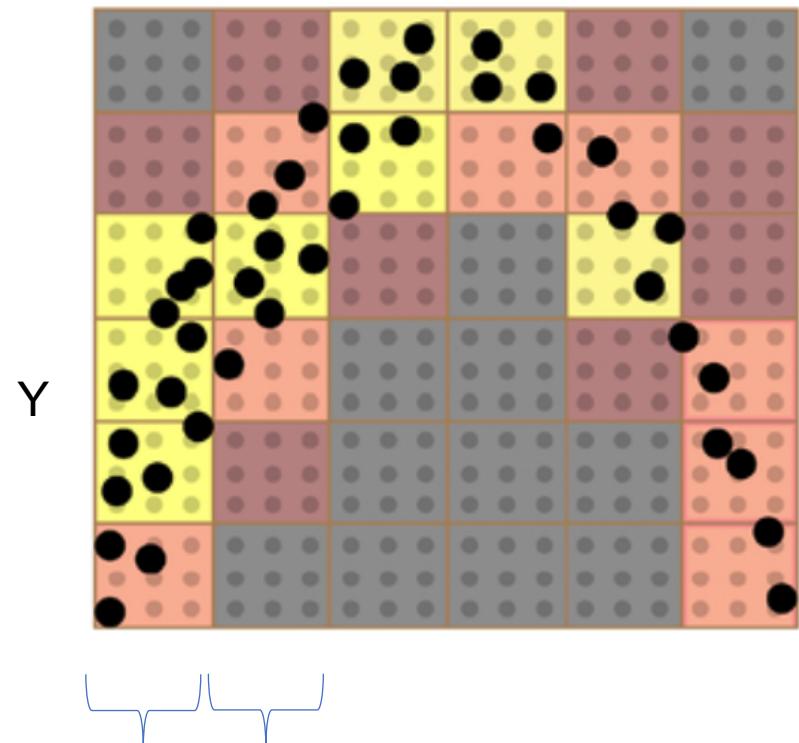
Computing MI ($X \rightarrow Y$) in Data

- Step 2: Compute entropy of Y (based on the Y bins)



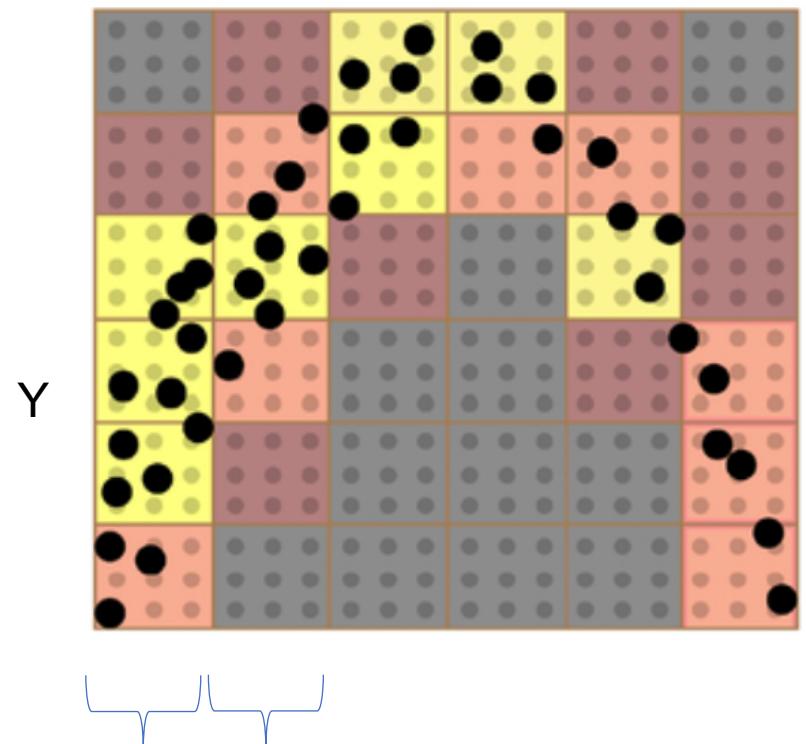
Computing MI ($X \rightarrow Y$) in Data

- Step 3: Compute entropy in each X column



Computing MI ($X \rightarrow Y$) in Data

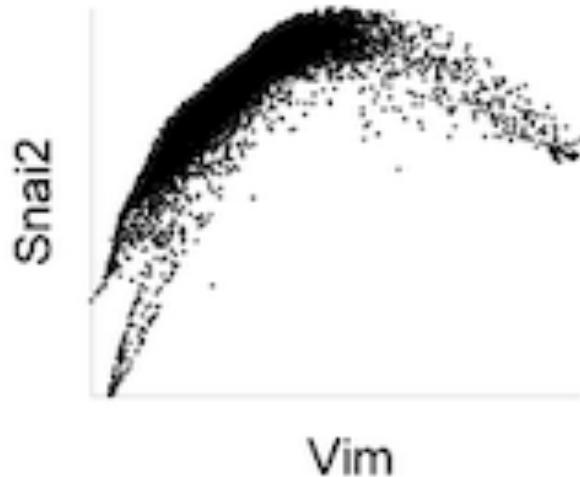
- Step 3: Compute $H(Y) - \sum_x (H(Y|X)) * p(x)$



Properties of Mutual Information

- $I(X;Y) \geq 0$
- Data processing inequality:
 - $I(X;Z) \leq I(X;Y)$ $X \rightarrow \boxed{\text{Channel}} \rightarrow Y \rightarrow \boxed{\text{Processing}} \rightarrow Z$
 - Conditioning on a third variable can increase or decrease MI between two variables
 - Why?

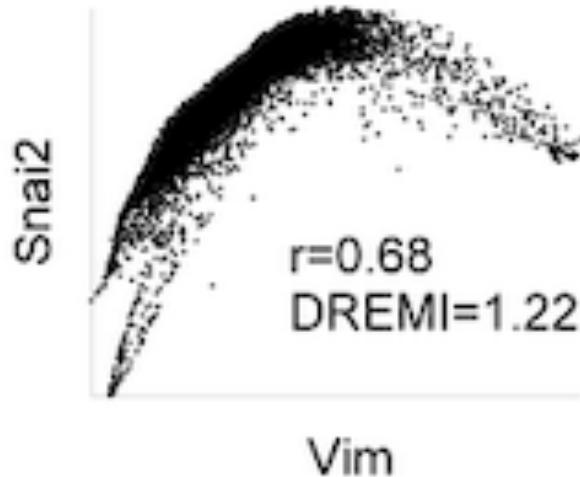
Imbalanced Data a Problem for MI



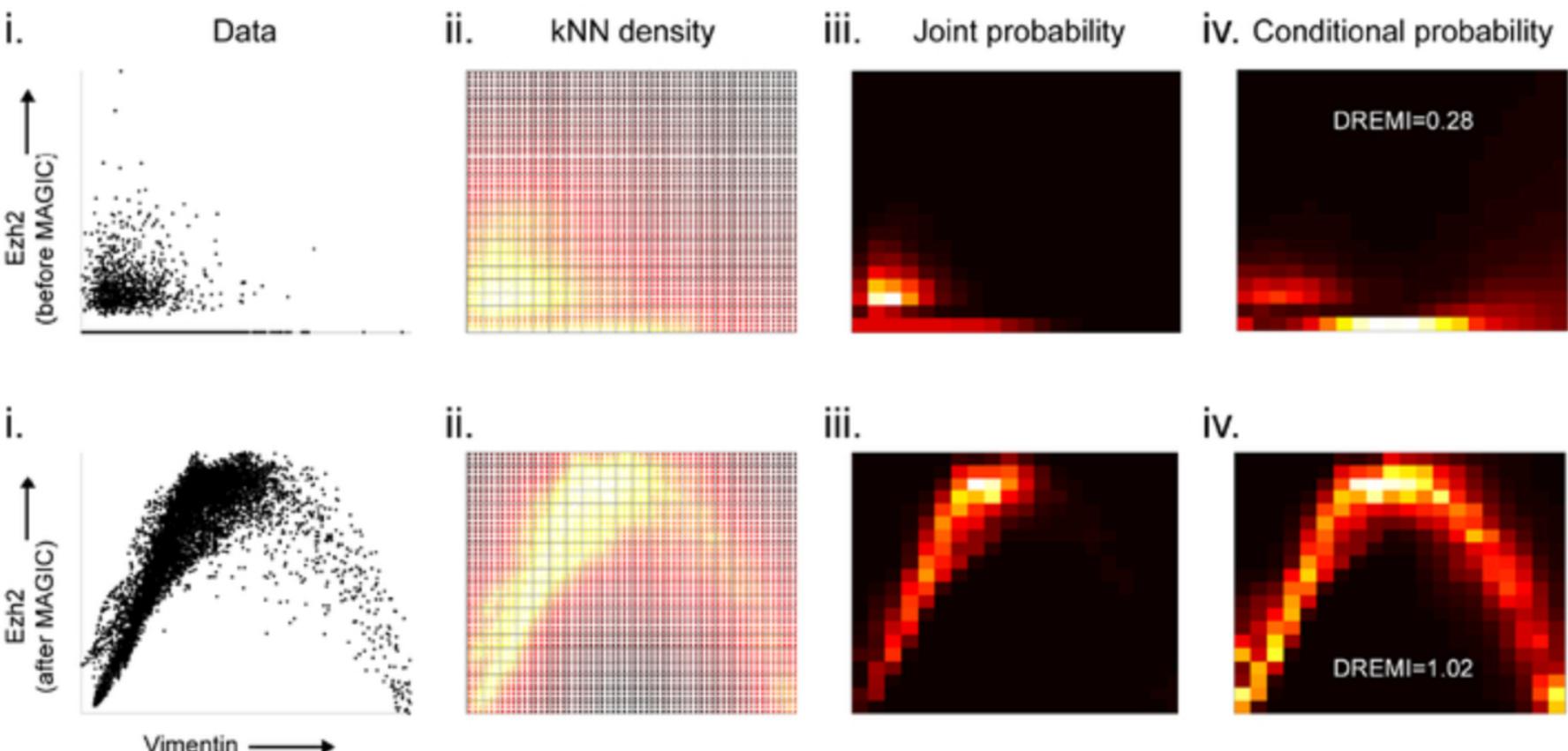
MI will just look at the dense blob for its computation and not the whole relationship

Density Resampled Estimate of Mutual Information (DREMI)

- Cells in lowly populated parts of relationship useful for unveiling shape and structure



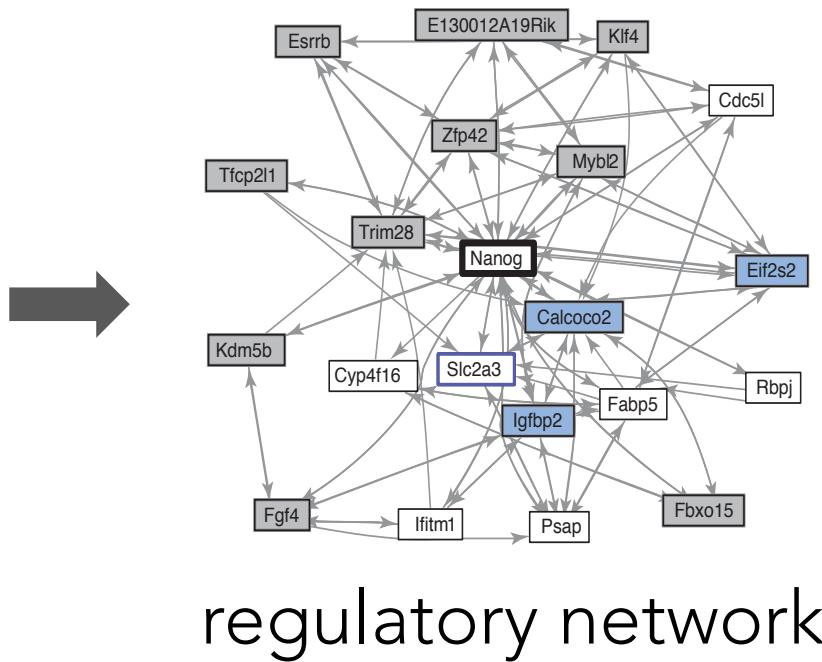
DREMI computes MI off of a Conditional Entropy



$$H(Y|X) - H(Y|X|X)$$



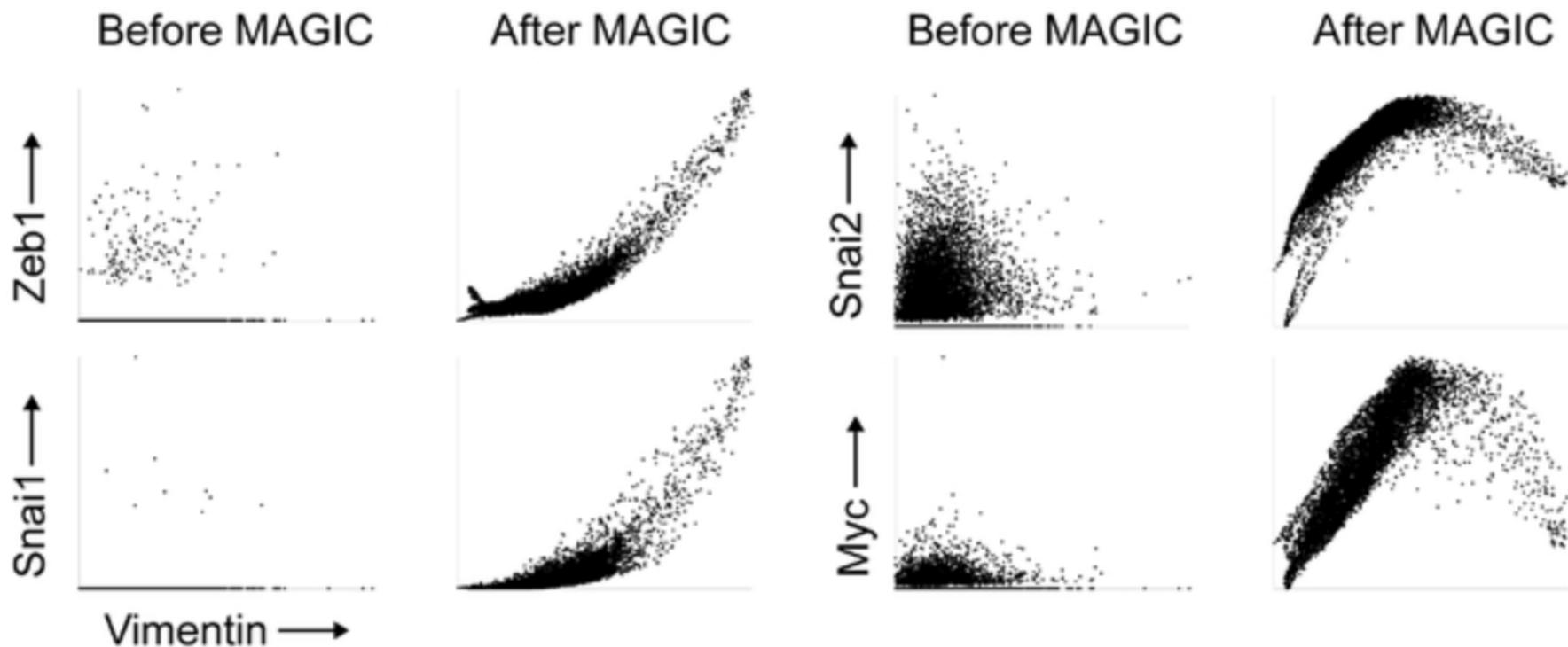
scRNA-seq data



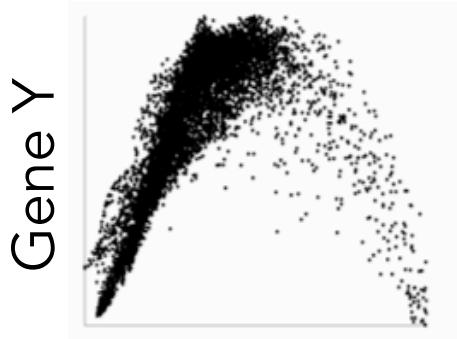
regulatory network

Using “snapshot” scRNA-seq data to predict transcription factor-target interactions involved in the epithelial—mesenchymal transition.

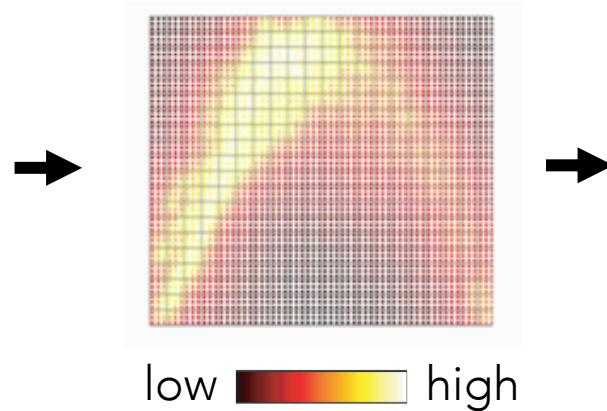
MAGIC Helps DREMI



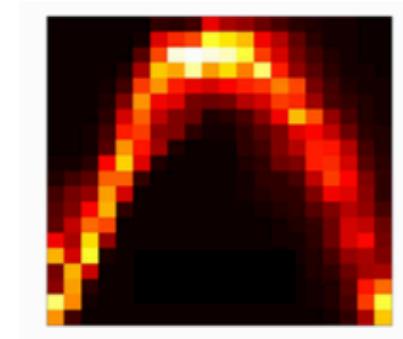
Data after MAGIC



Density estimation

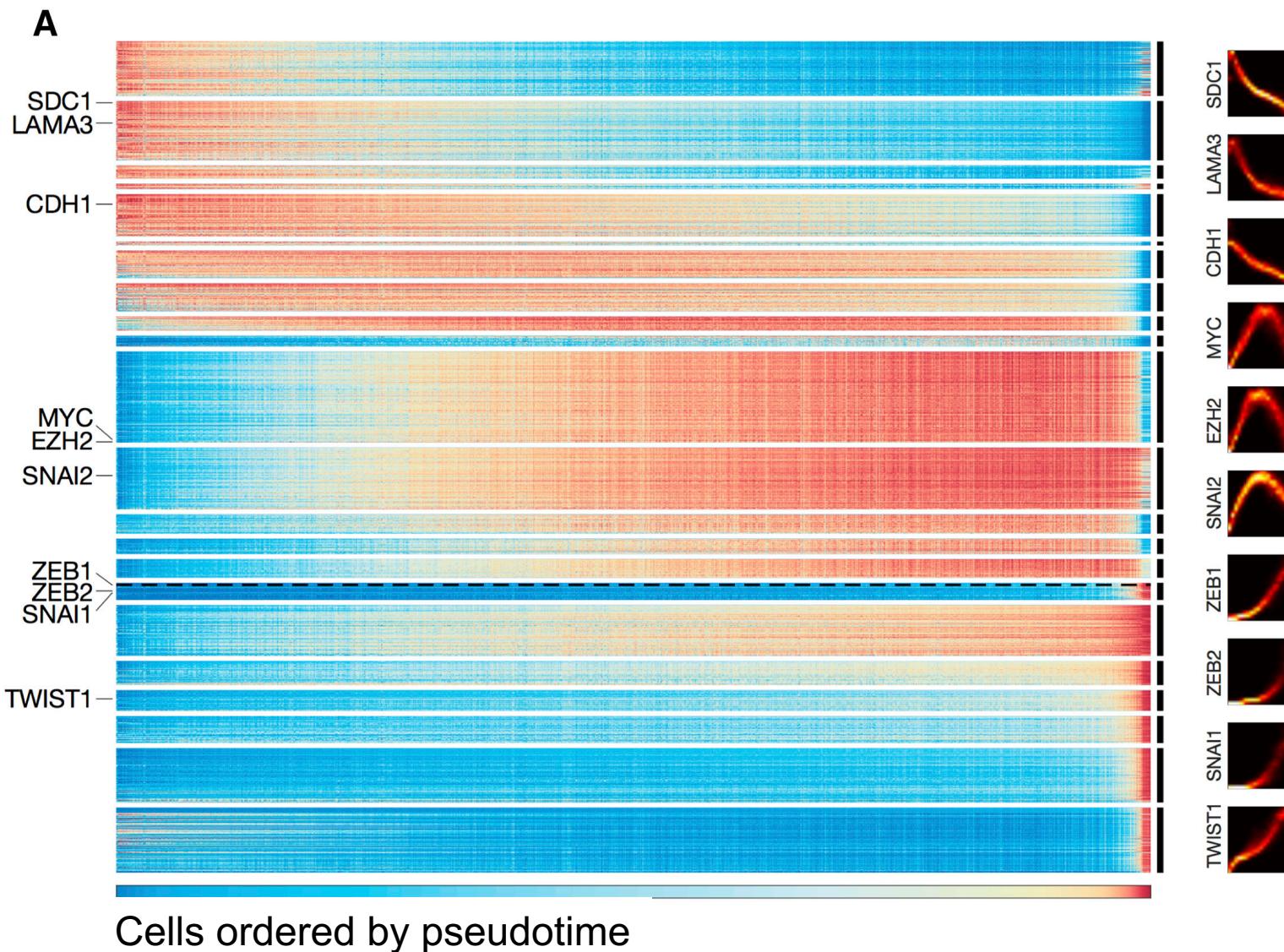


DREMI



quantifying gene-gene relationships

Ranking genes by association with pseudotime identifies drivers of differentiation

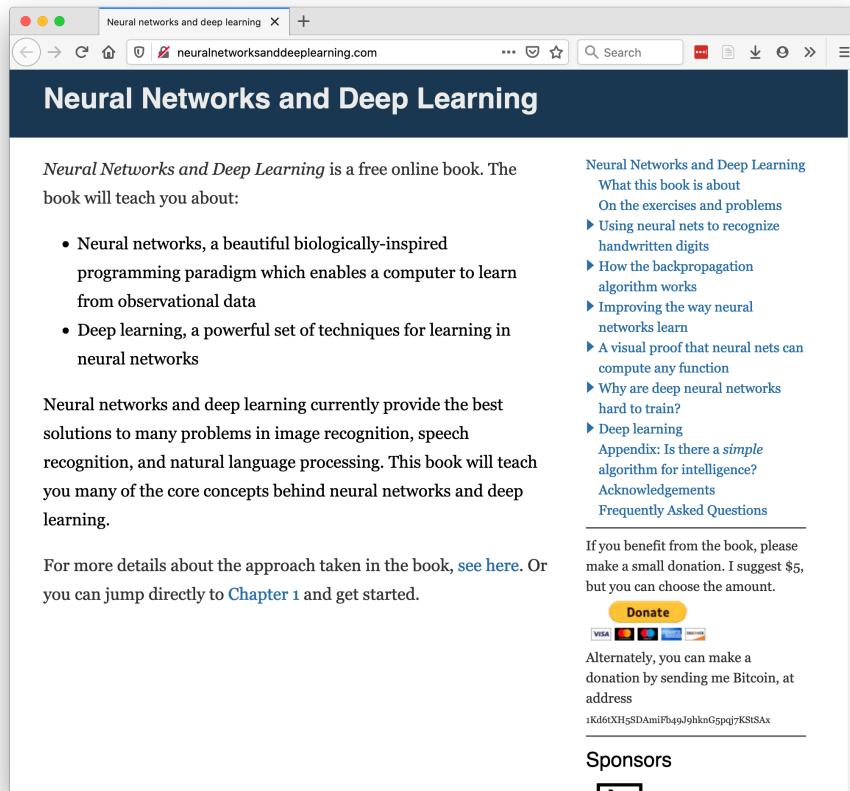


Conclusions

1. Mutual information (MI) can be used to identify non-linear relationships between variables
2. Density normalization allows for quantifying MI in regions of data sparsity
3. MI can quantify gene-gene relationships to infer regulatory relationships
4. MI of genes with pseudotime identifies drivers of differentiation

For tomorrow: Ch. 1 & 2

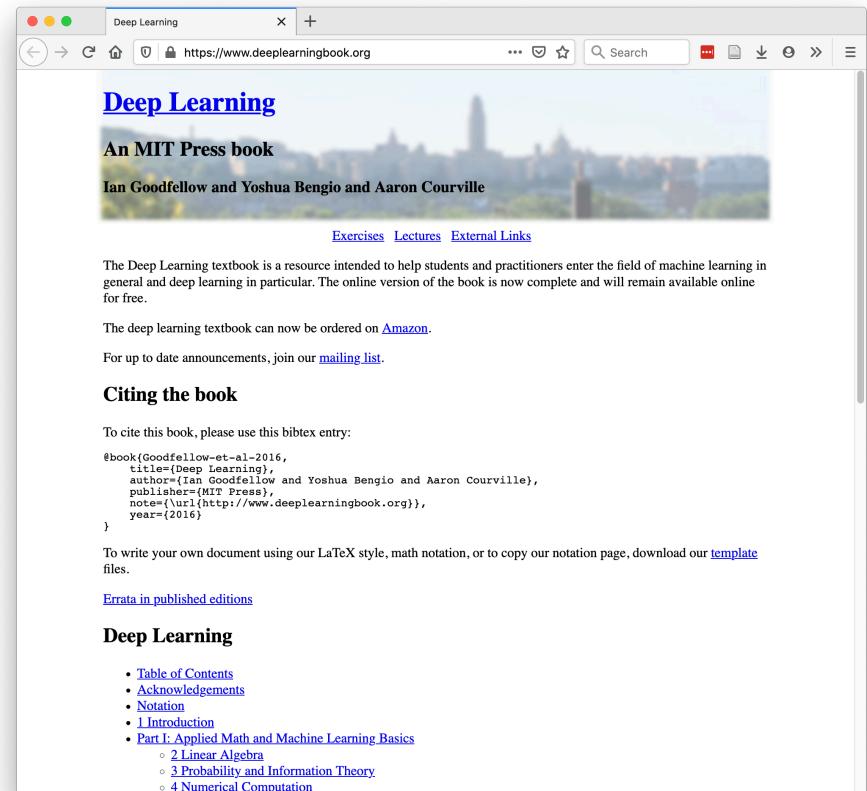
(Beginner)



The screenshot shows a web browser window for "Neural Networks and Deep Learning". The title bar says "Neural networks and deep learning X". The main content area has a dark blue header with the text "Neural Networks and Deep Learning". Below the header, there is a paragraph about the book being a free online resource that will teach you about neural networks and deep learning. It lists two main bullet points: "Neural networks, a beautiful biologically-inspired programming paradigm which enables a computer to learn from observational data" and "Deep learning, a powerful set of techniques for learning in neural networks". A larger paragraph below explains that neural networks and deep learning provide solutions to various problems like image recognition, speech recognition, and natural language processing. At the bottom, there is a "Donate" button with payment method icons (VISA, MasterCard, etc.) and a Bitcoin address: 1Kd6tXH5SDAmiFb49J9hknG5pqj7KSASx.

<http://neuralnetworksanddeeplearning.com/>

(Advanced)



The screenshot shows a web browser window for "Deep Learning". The title bar says "Deep Learning X". The main content area has a dark blue header with the text "Deep Learning". Below the header, it says "An MIT Press book" and credits "Ian Goodfellow and Yoshua Bengio and Aaron Courville". There are links for "Exercises", "Lectures", and "External Links". A paragraph states that the Deep Learning textbook is a resource for machine learning general and deep learning, available online for free. It mentions ordering on Amazon and joining a mailing list. A section titled "Citing the book" provides a BibTeX entry:

```
#book{Goodfellow-et-al-2016,
  title="Deep Learning",
  author="Ian Goodfellow and Yoshua Bengio and Aaron Courville",
  publisher="MIT Press",
  note={(url:https://www.deeplearningbook.org)},
  year={2016}
}
```

To cite the book, use this BibTeX entry. You can write your own document using LaTeX style, or download template files. There is a link for "Errata in published editions".

The "Deep Learning" section contains a table of contents with links to chapters 2, 3, and 4.

<https://www.deeplearningbook.org/>