The Krishnaswamy Laboratory

Yale Genetics and Yale SEAS present

Machine Learning

for

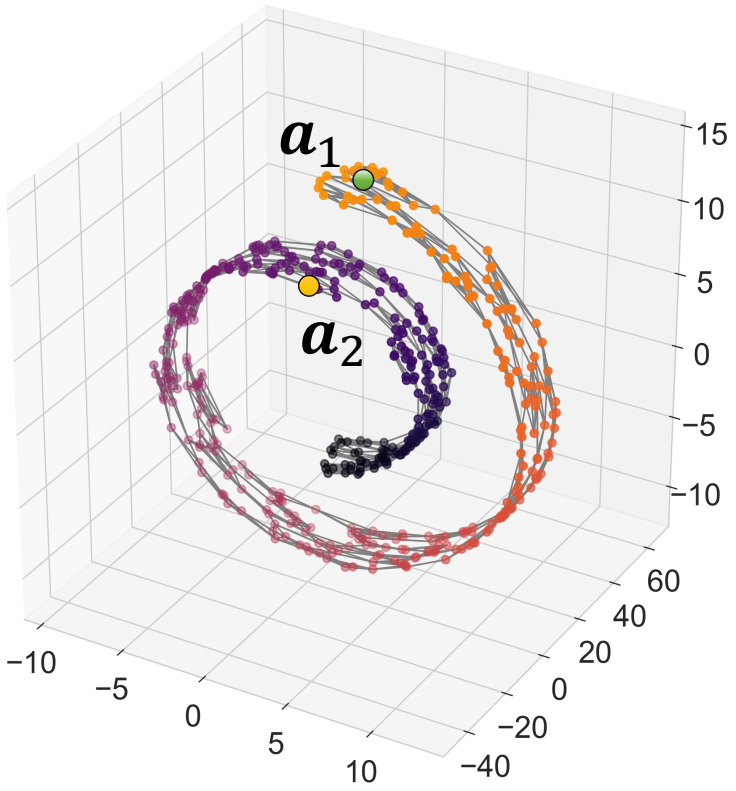
Single Cell Analysis

Online - May 20-29, 2020

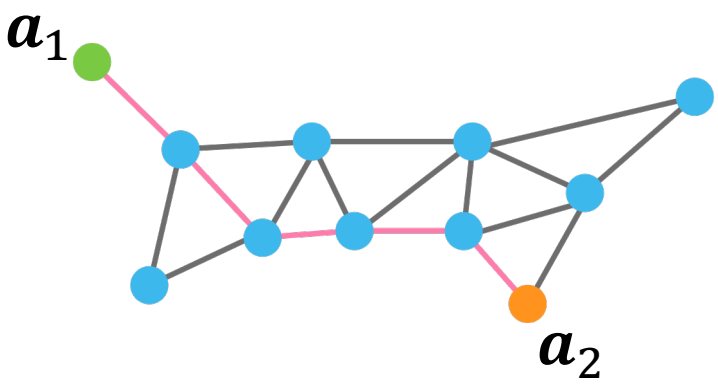
What is your favorite sport to follow?

Recap from Day 2

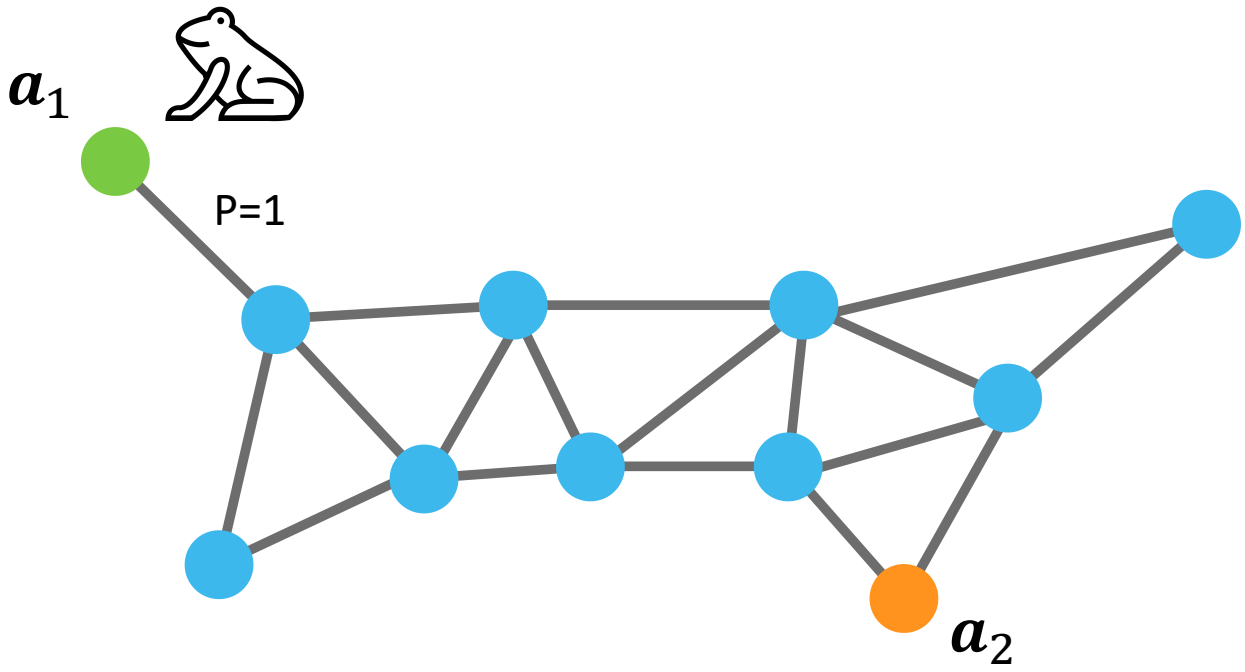
Paths or “walks” on graphs reveal dominant data manifold directions



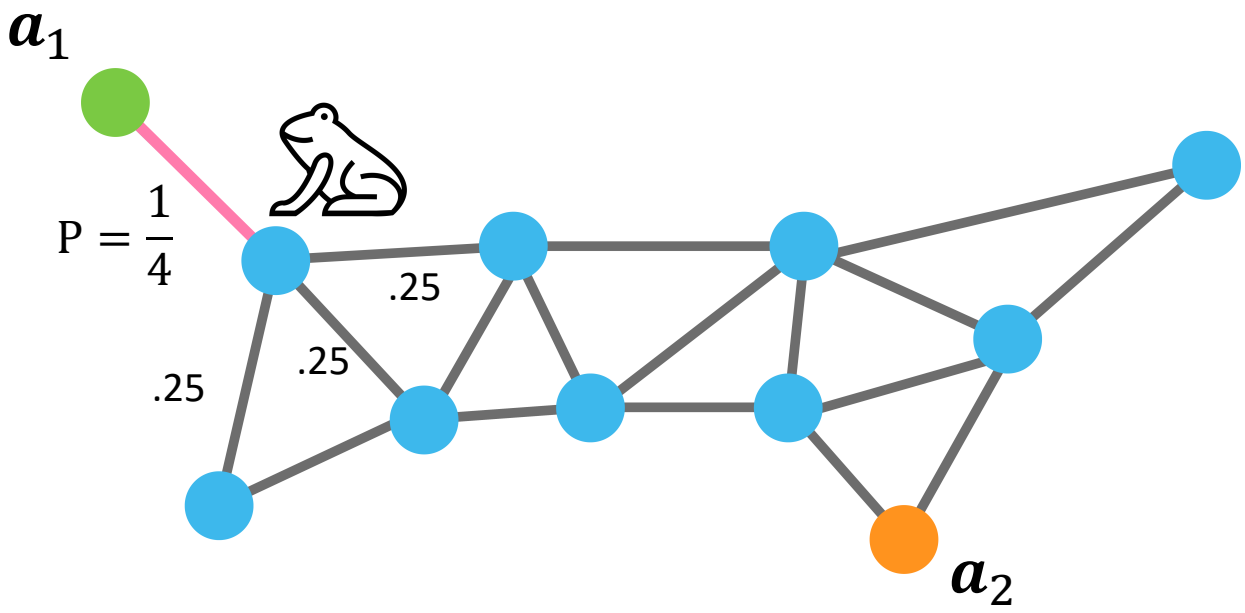
Shortest path between points



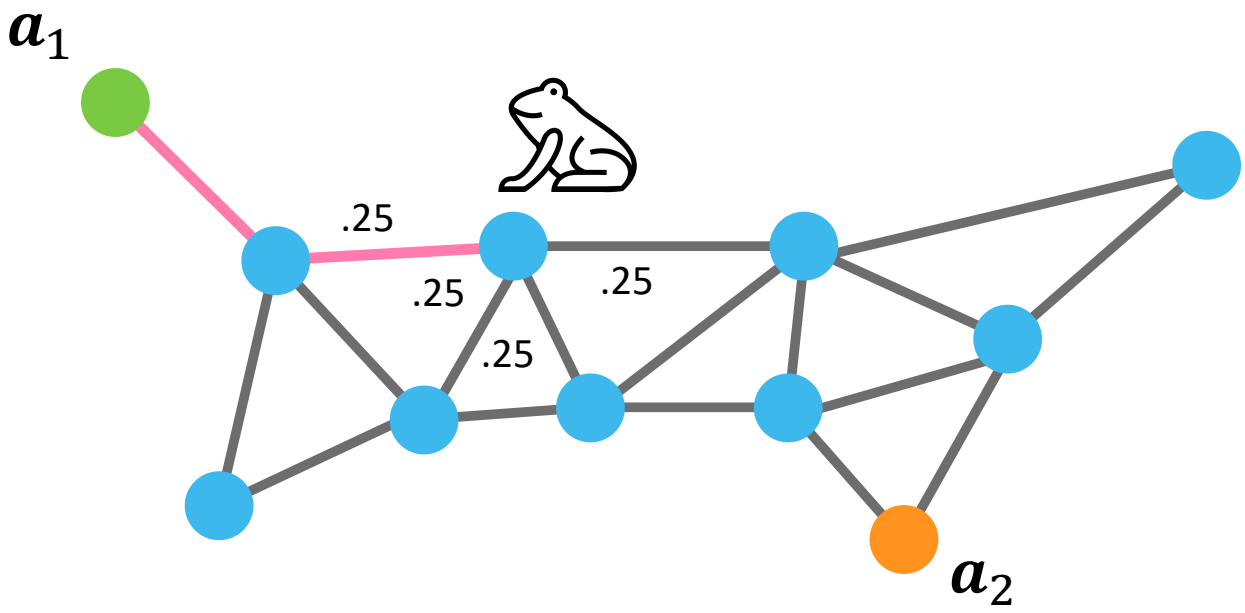
Paths or “walks” on graphs reveal dominant data manifold directions



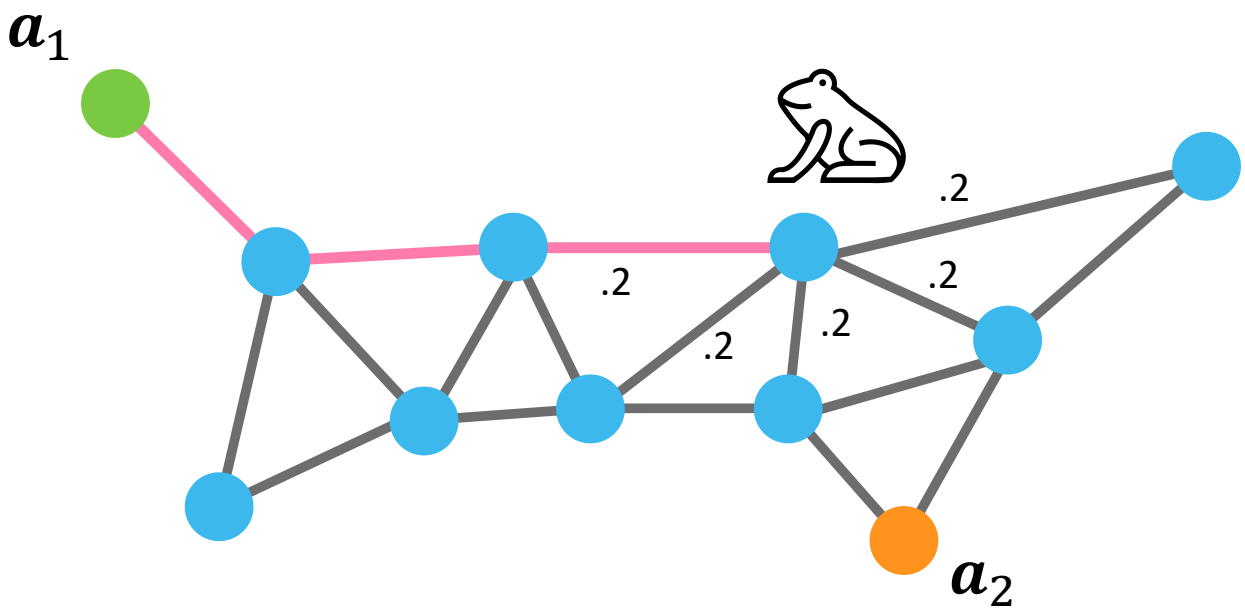
Paths or “walks” on graphs reveal dominant data manifold directions



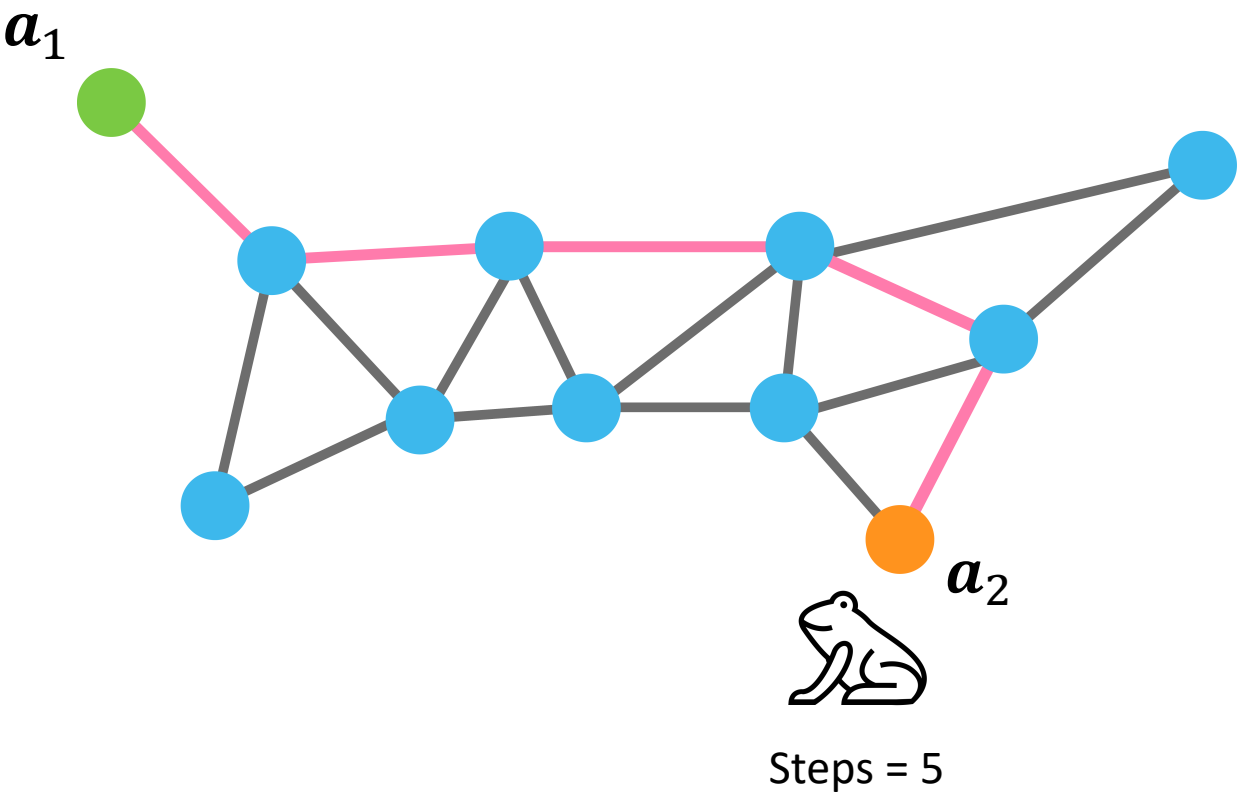
Paths or “walks” on graphs reveal dominant data manifold directions

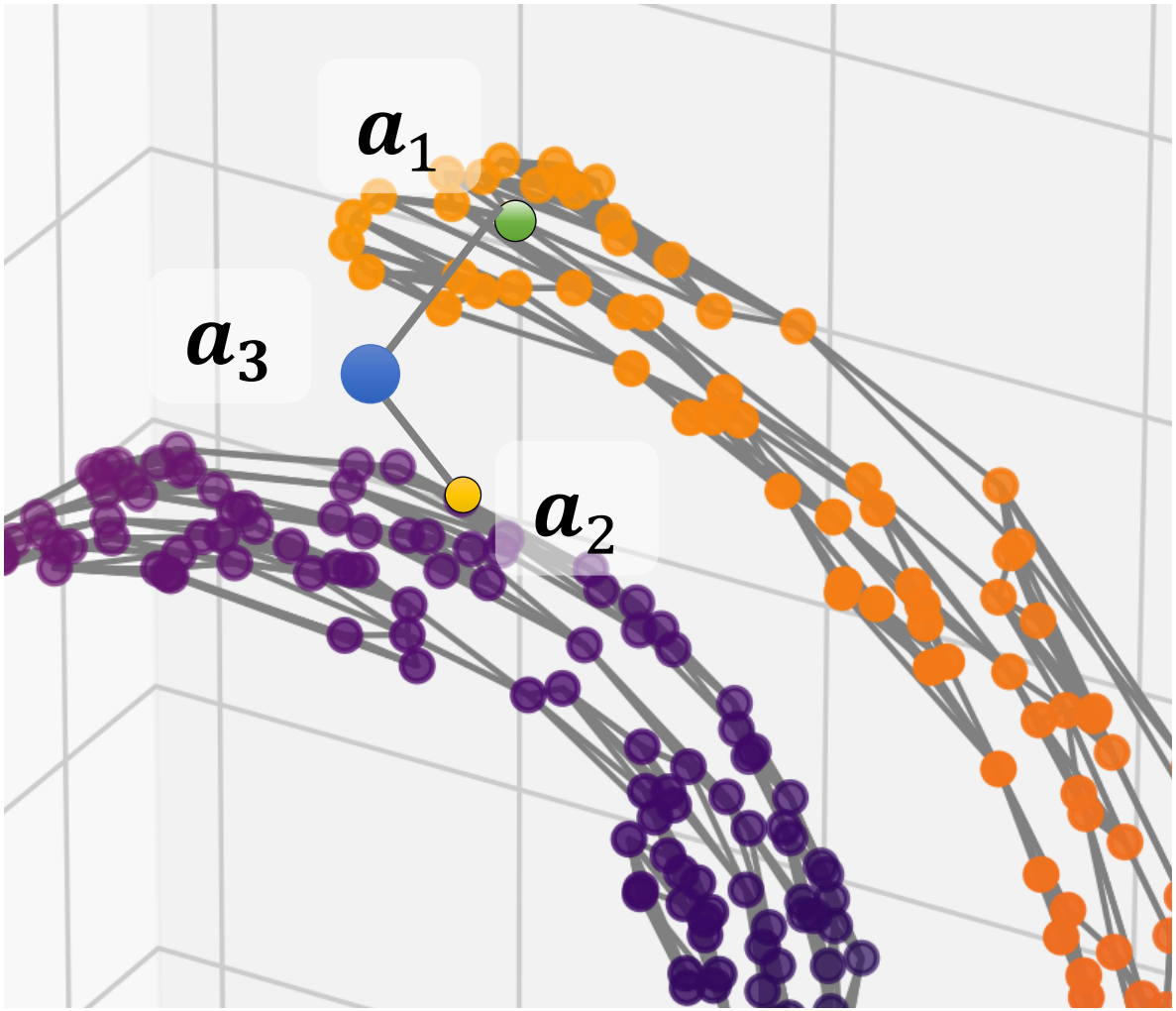


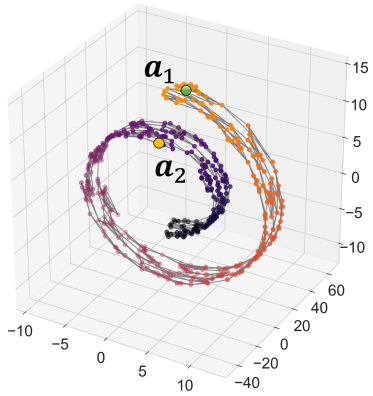
Paths or “walks” on graphs reveal dominant data manifold directions

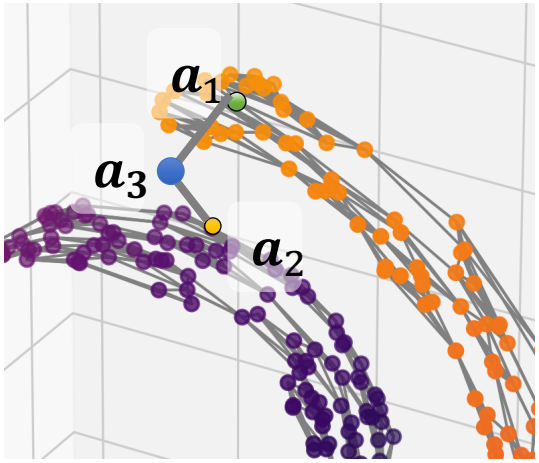


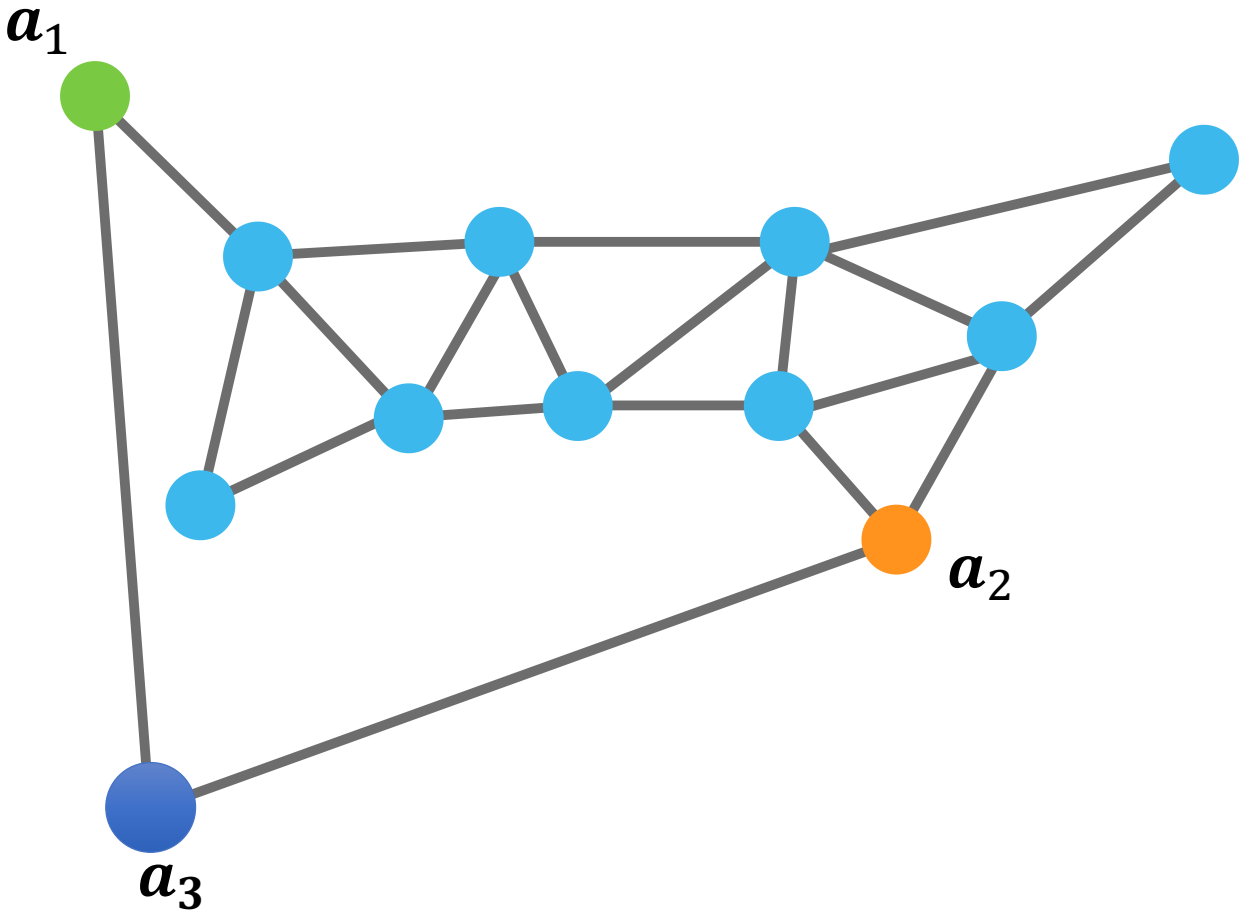
Paths or “walks” on graphs reveal dominant data manifold directions

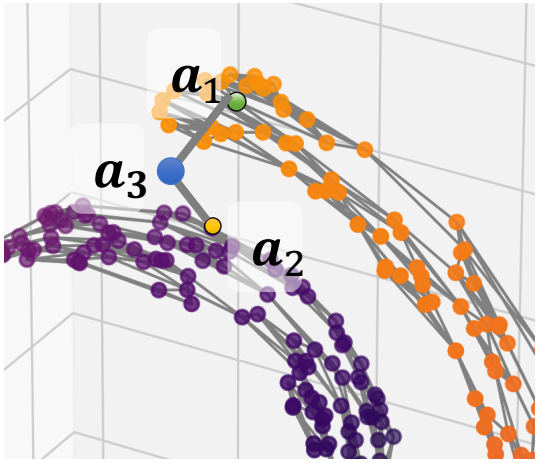


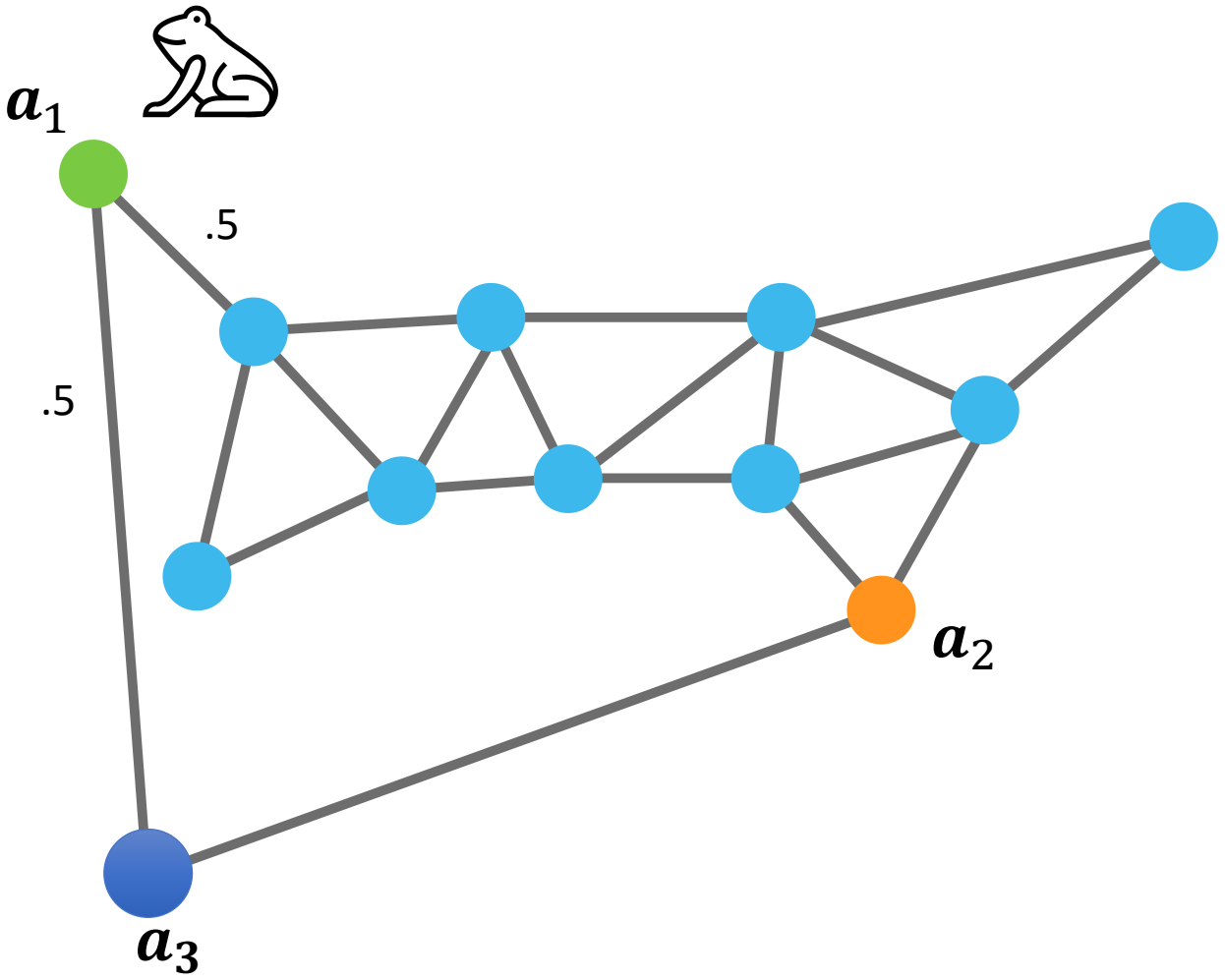

a_1
a_3
a_2


a_1
a_2
15
10
5
0
−5
−10
60
40
20
0
−20
−40
−10
−5
0
5
10

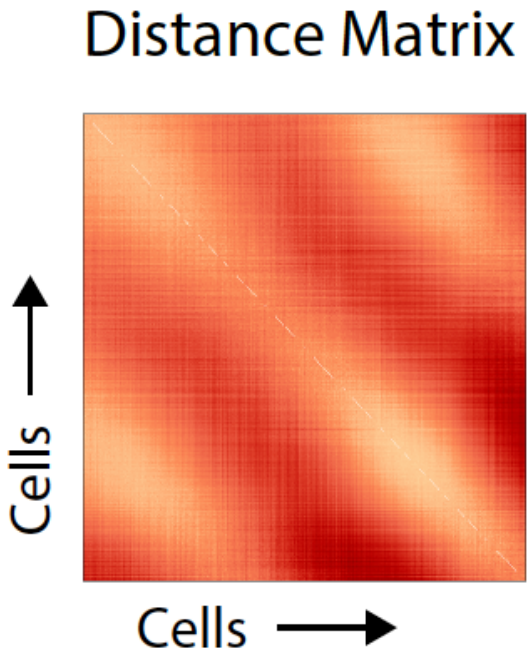

a_1
a_3
a_2


a_1
a_2
a_3


a_1
a_3
a_2

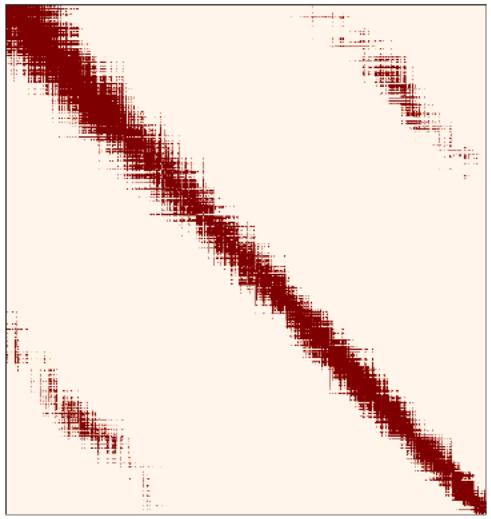

a_1
.5
.5
a_2
a_3

Distance Matrix

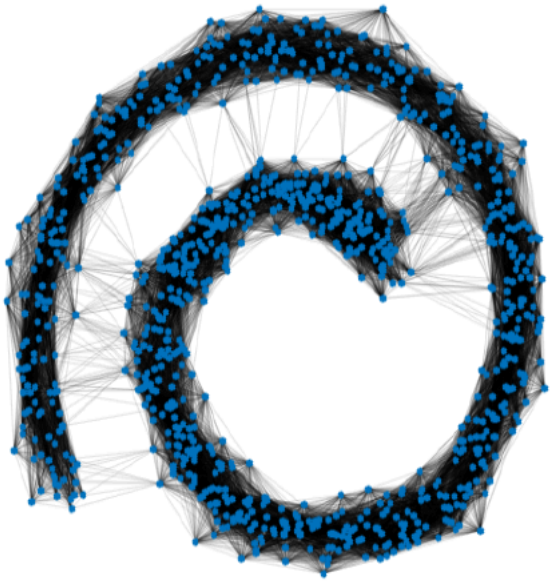


Entries are distances $D(x_i, x_j) = \sqrt{||x_i - x_j||}$

Affinity Matrix



(Gaussian Kernel)



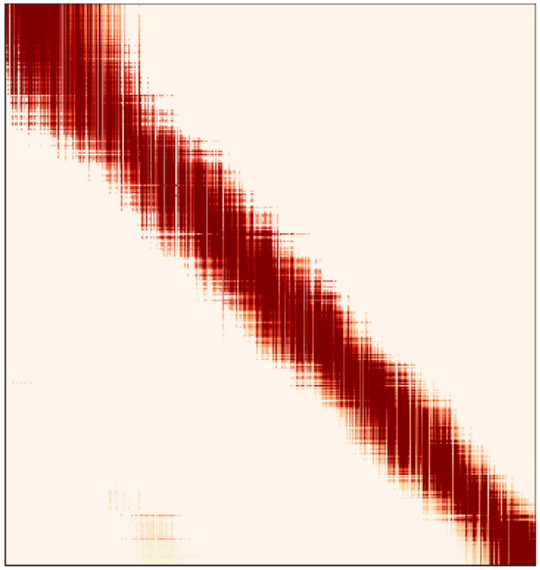
Graph Representation

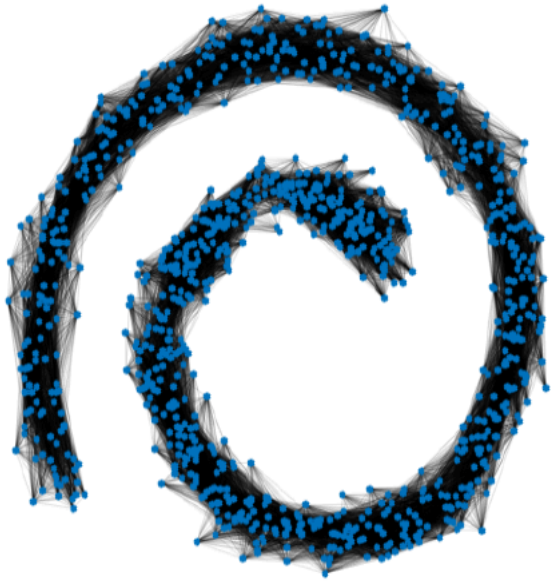
Entries are affinities

$$A(x_i, x_j) = \exp(-\frac{D(x_i,x_j)^2}{\sigma})$$

Powered Markov Matrix—Denoises the Affinities (relationship between points)

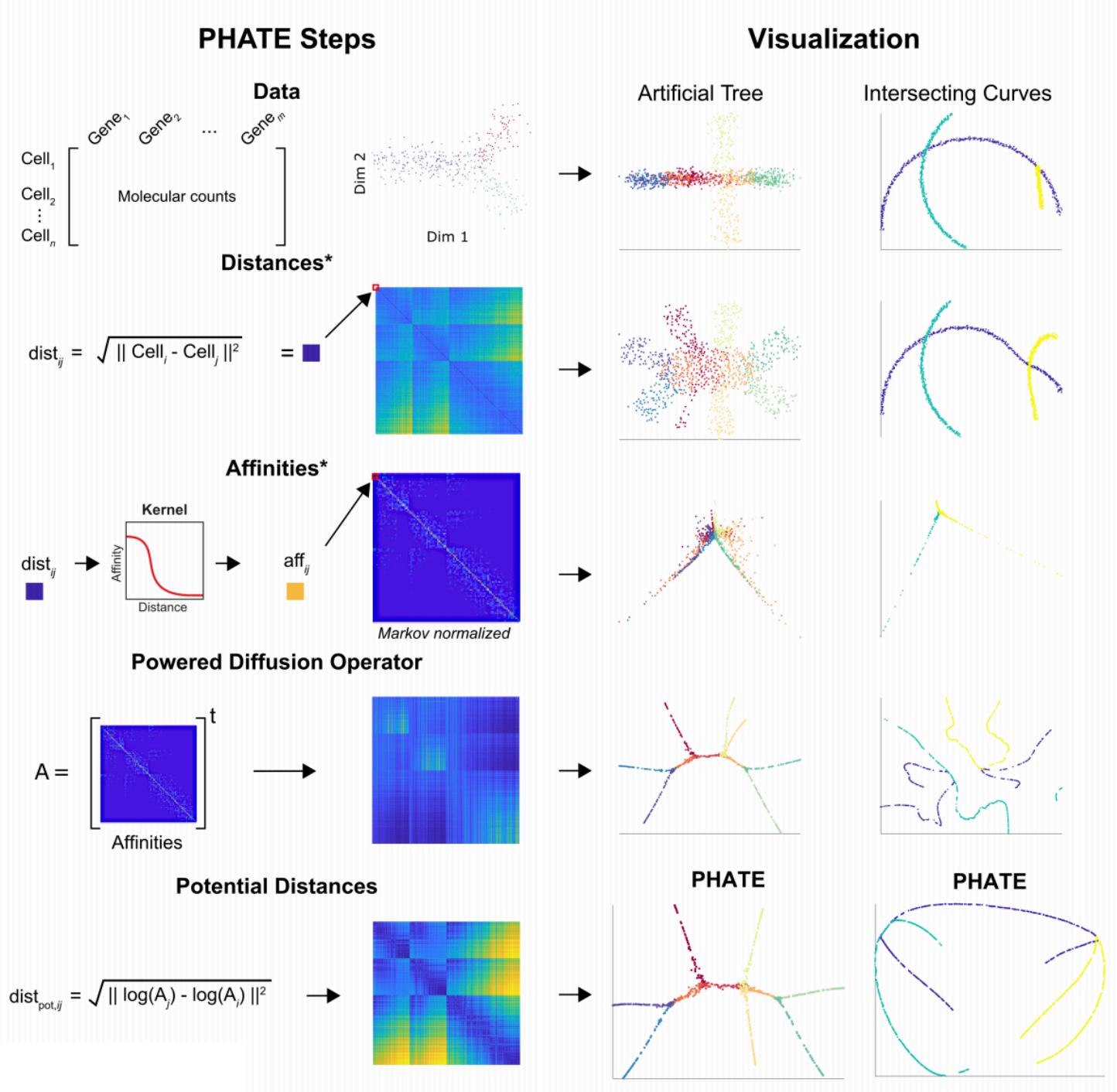
Powered Markov Matrix



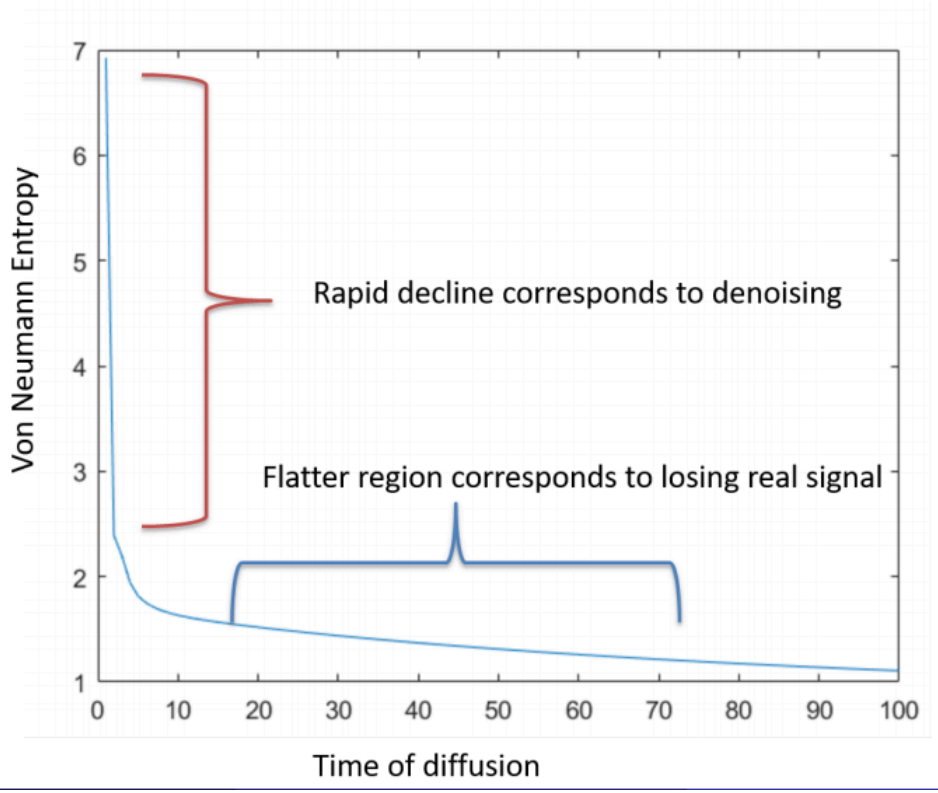


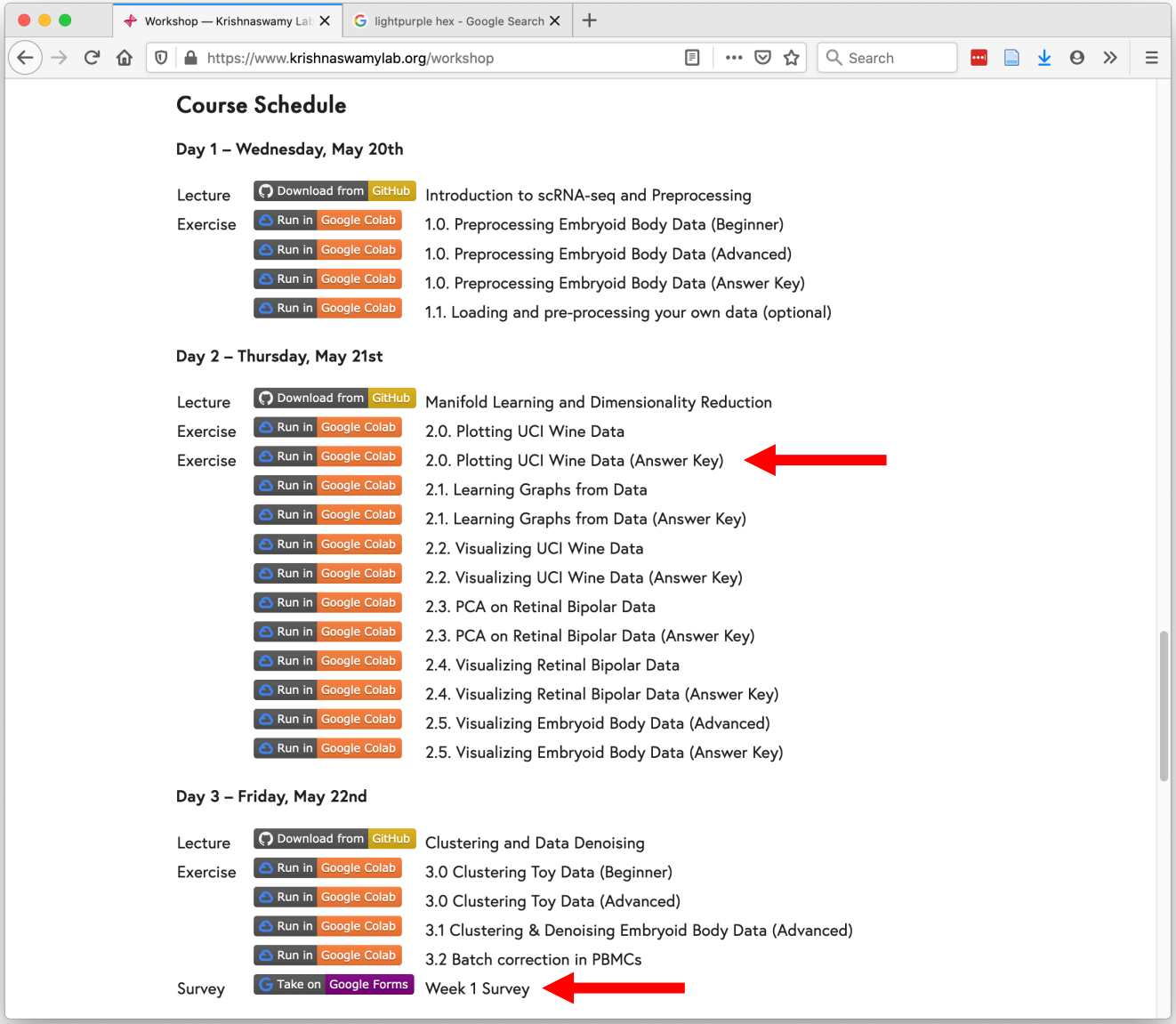
Entries are row normalized affinities

$$M(x_i, x_j) = \frac{A(x_i, x_j)}{\sum_j A(x_i, x_j)}$$

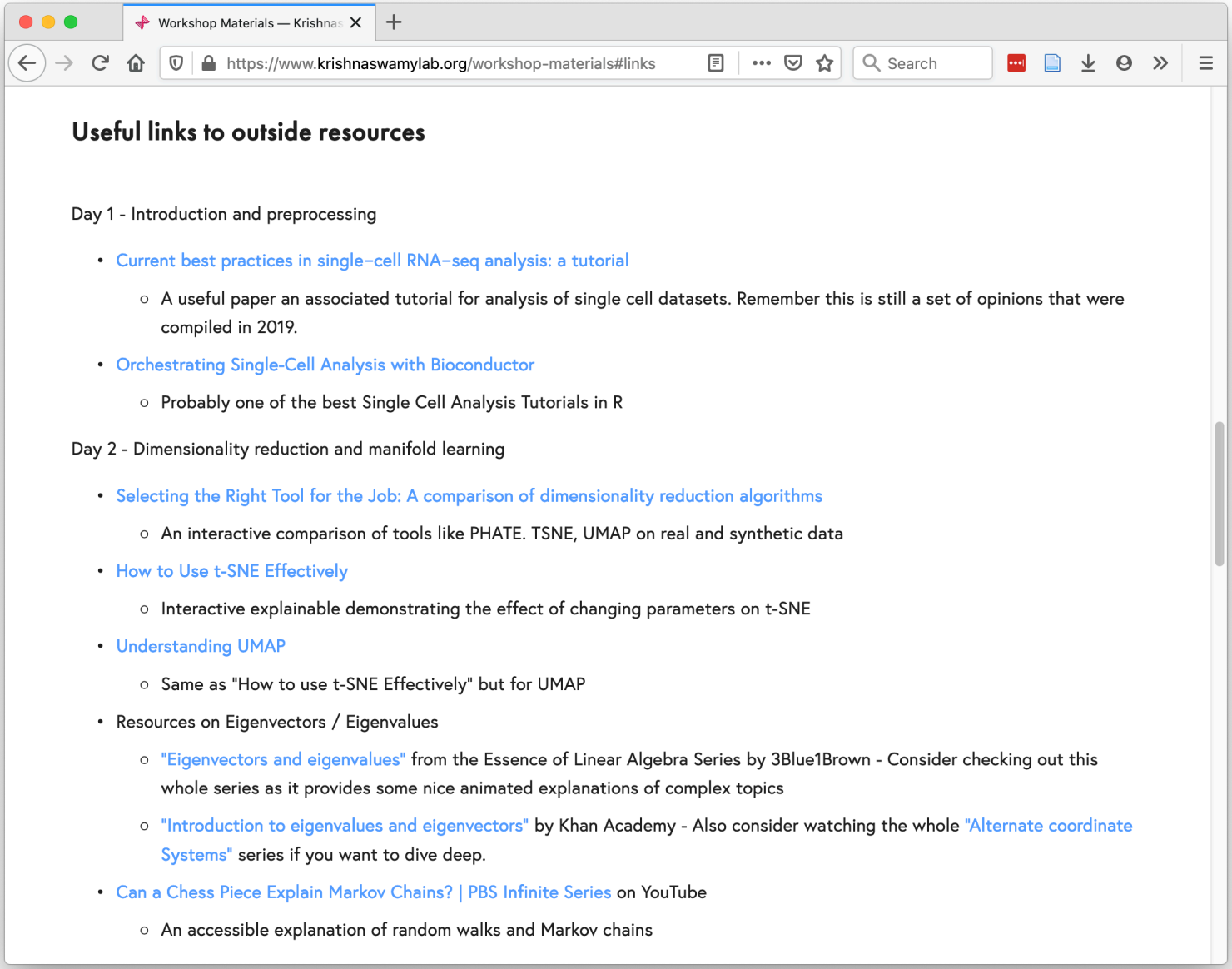

PHATE Steps
Visualization
Data
Gene1
Gene2
...
Genem
Cell1
Cell2
Cellsn
Molecular counts
Dim 2
Dim 1
Artificial Tree
Intersecting Curves
(PCA)
Distances*
distij = √|| Celli - Cellj ||2
=
(MDS)
Affinities*
distij
Kernel
Affinity
Distance
affij
Markov normalized
(tSNE)
Powered Diffusion Operator
A =
t
Affinities
(DMs)
Potential Distances
distpot,ij = √|| log(Aj) - log(Ai) ||2
PHATE
PHATE

Time of Diffusion t





https://www.krishnaswamylab.org/workshop



Useful links to outside resources

Day 1 - Introduction and preprocessing

- Current best practices in single–cell RNA–seq analysis: a tutorial
  - A useful paper an associated tutorial for analysis of single cell datasets. Remember this is still a set of opinions that were compiled in 2019.
- Orchestrating Single-Cell Analysis with Bioconductor
  - Probably one of the best Single Cell Analysis Tutorials in R

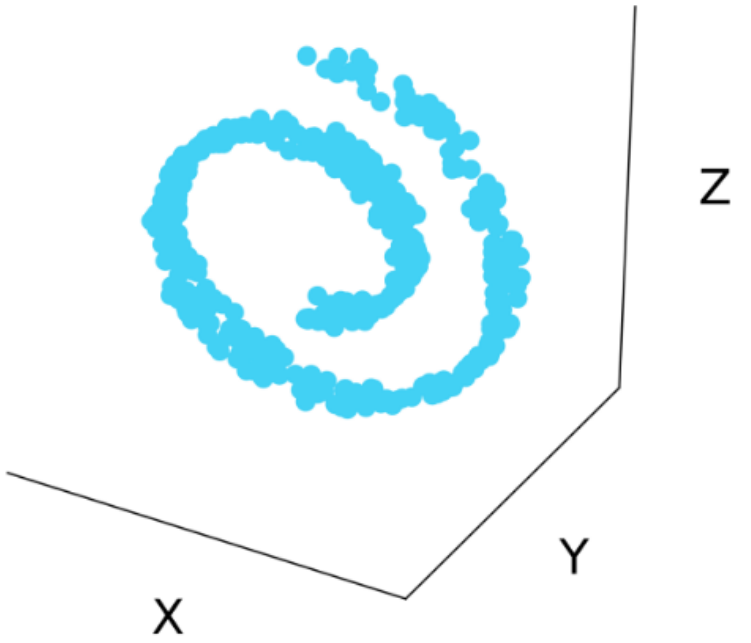
Day 2 - Dimensionality reduction and manifold learning

- Selecting the Right Tool for the Job: A comparison of dimensionality reduction algorithms
  - An interactive comparison of tools like PHATE. TSNE, UMAP on real and synthetic data
- How to Use t-SNE Effectively
  - Interactive explainable demonstrating the effect of changing parameters on t-SNE
- Understanding UMAP
  - Same as "How to use t-SNE Effectively" but for UMAP
- Resources on Eigenvectors / Eigenvalues
  - "Eigenvectors and eigenvalues" from the Essence of Linear Algebra Series by 3Blue1Brown - Consider checking out this whole series as it provides some nice animated explanations of complex topics
  - "Introduction to eigenvalues and eigenvectors" by Khan Academy - Also consider watching the whole "Alternate coordinate Systems" series if you want to dive deep.
- Can a Chess Piece Explain Markov Chains? | PBS Infinite Series on YouTube
  - An accessible explanation of random walks and Markov chains

https://www.krishnaswamylab.org/workshop-materials#links

Day 3: Denoising, Batch Correction, and Clustering

What is the intrinsic or latent dimensionality of this data?



0

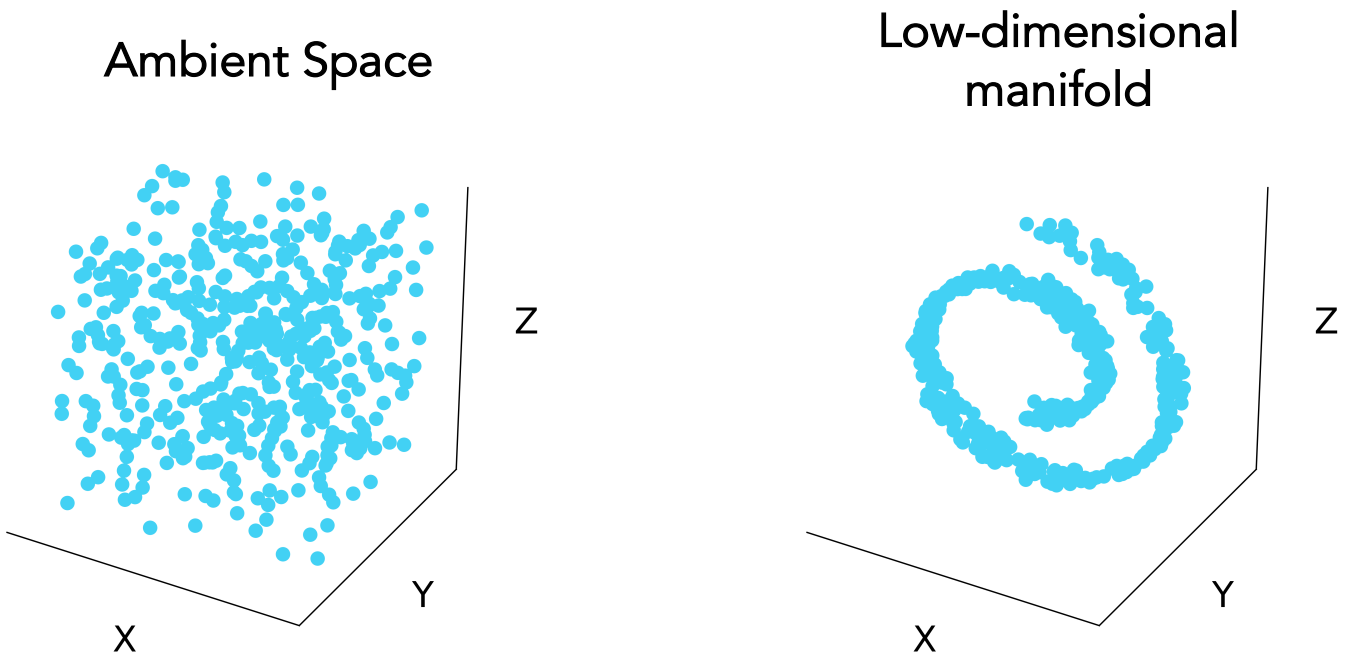
1

2

3

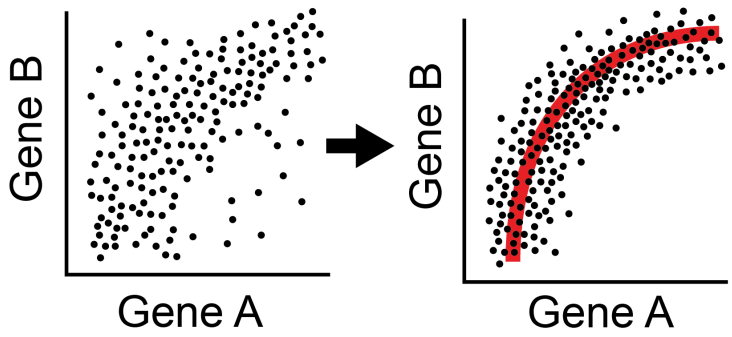
4

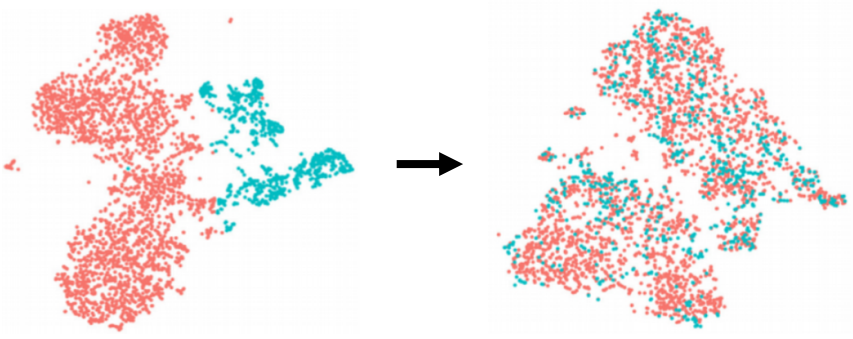
Latent structure in high dimensional data



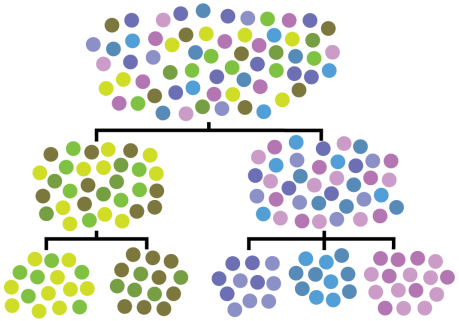
Other uses of affinity matrices, eigenvectors

Data denoising and batch normalization





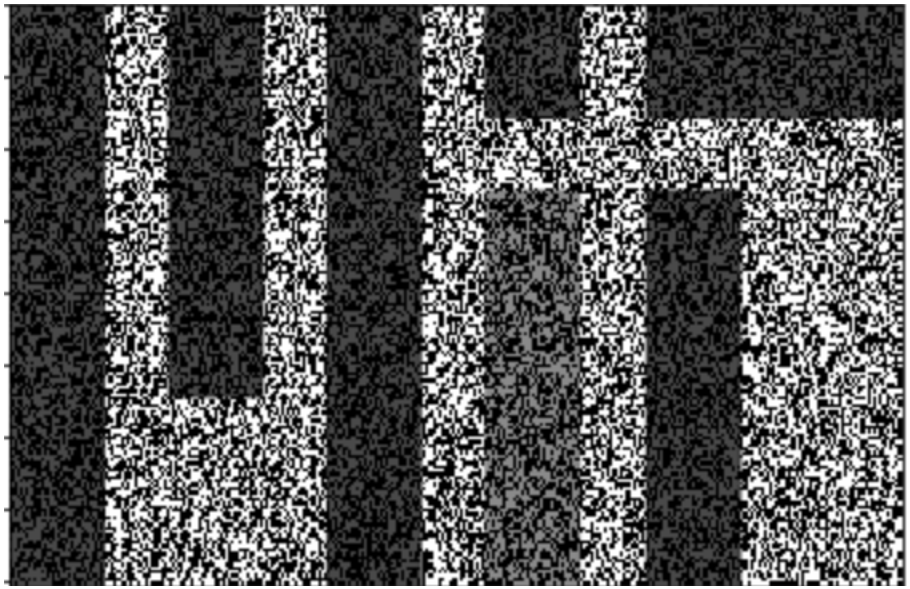
Clustering



Using the manifold model to denoise data

Denoising by eliminating dimensions

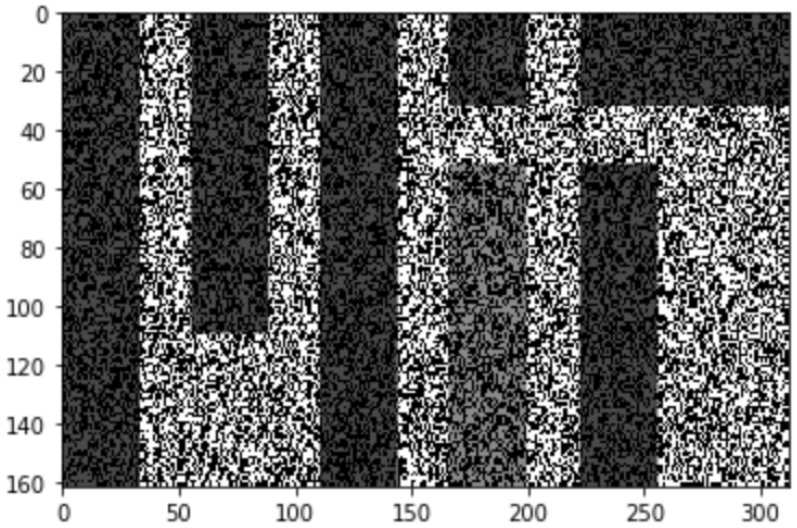
- The number of dimensions in a data that are independent are called RANK
- You can denoise data by lowering rank



Noisy image

Recreate the data without noise

- For this to work you have to have your dimensions be broken up into ***data dimensions and noise dimensions***
- Noisy image on the right is broken up into columns and rows instead
- What do we do?



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

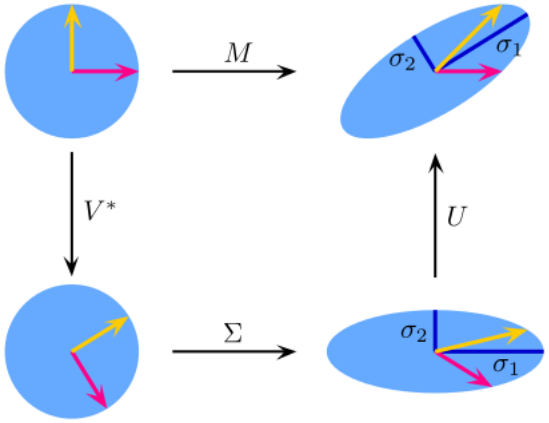
Text **YALEML** to **22333** once to join

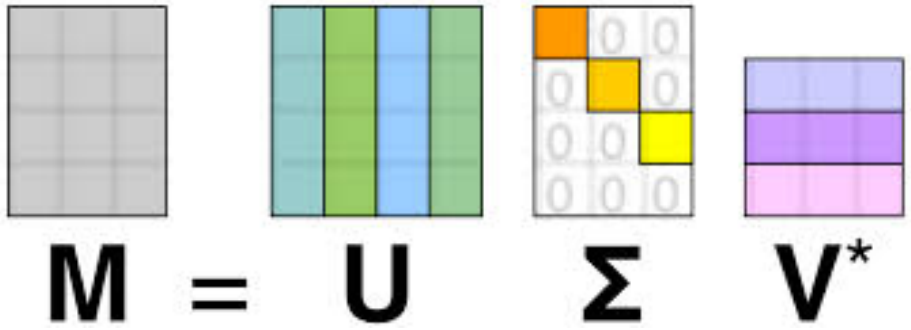
How do we get dimensions that correspond to data "signal" and "noise" ?

PCA splits signal and noise

- The axes with a lot of variation are likely to correspond to data
- Axes with little variation correspond to noise directions
- The amount of variation captured in each eigenvector is given by its eigenvalue
- PC1 will have the highest eigenvalue, captures the most variation ..
  - PC2 next most
    - And so on…

SVD: Process similar to Eigendecomposition on non-square matrices



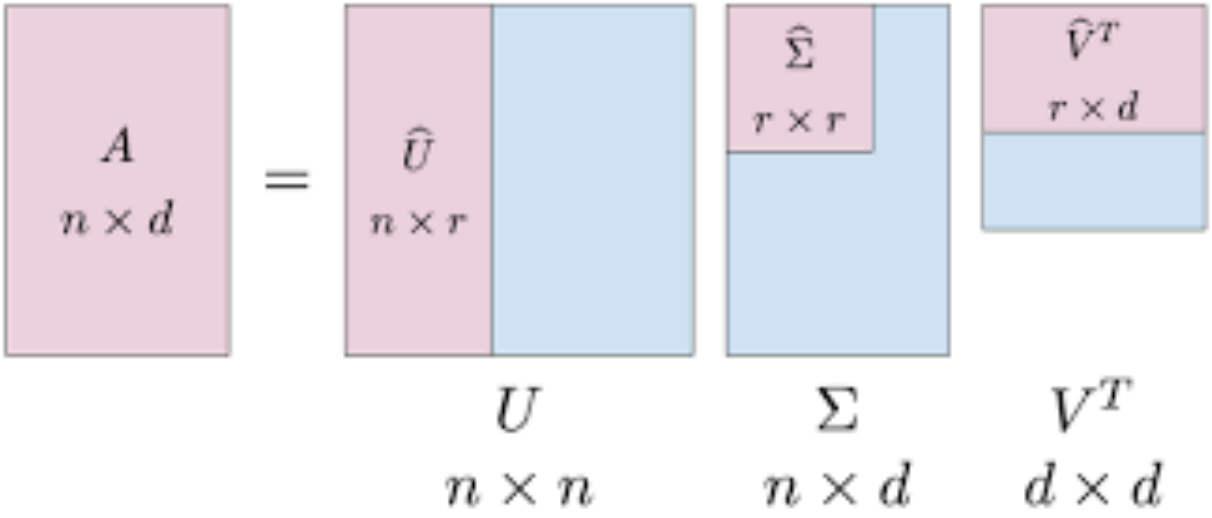


Singular values are eigenvalues of MM* or M*M
Singular vectors are eigenvectors of MM* or M*M

For a mean centered feature matrix the singular vectors U are also PCs

Low Rank approximation

- Eliminate the singular vectors with low singular values and recreate the matrix



Low Rank Approximation on MIT matrix

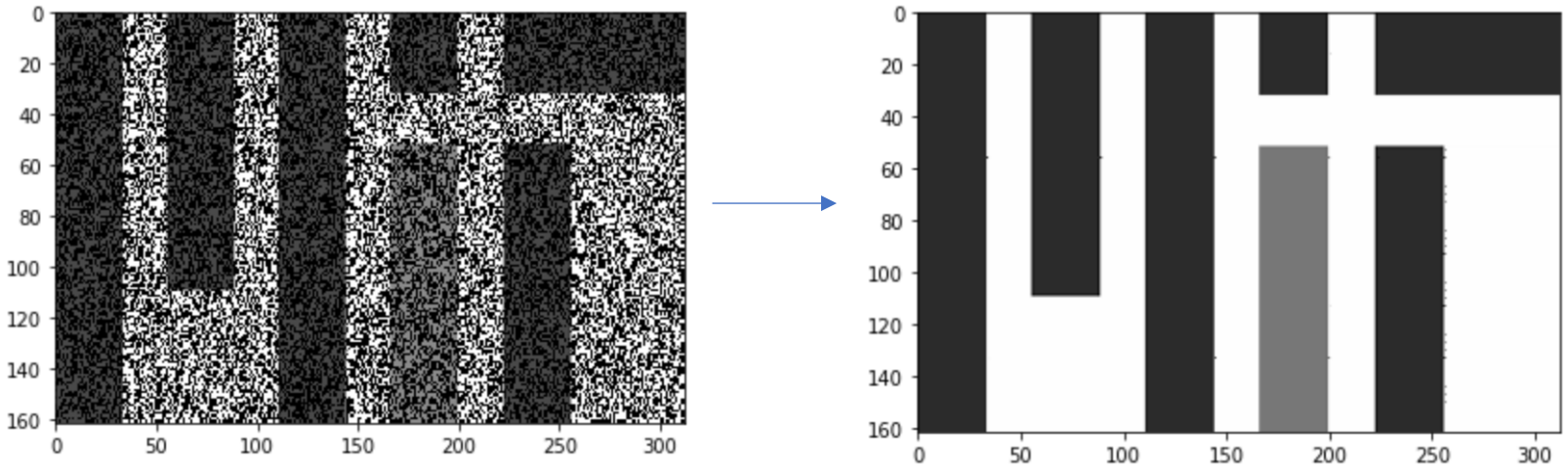
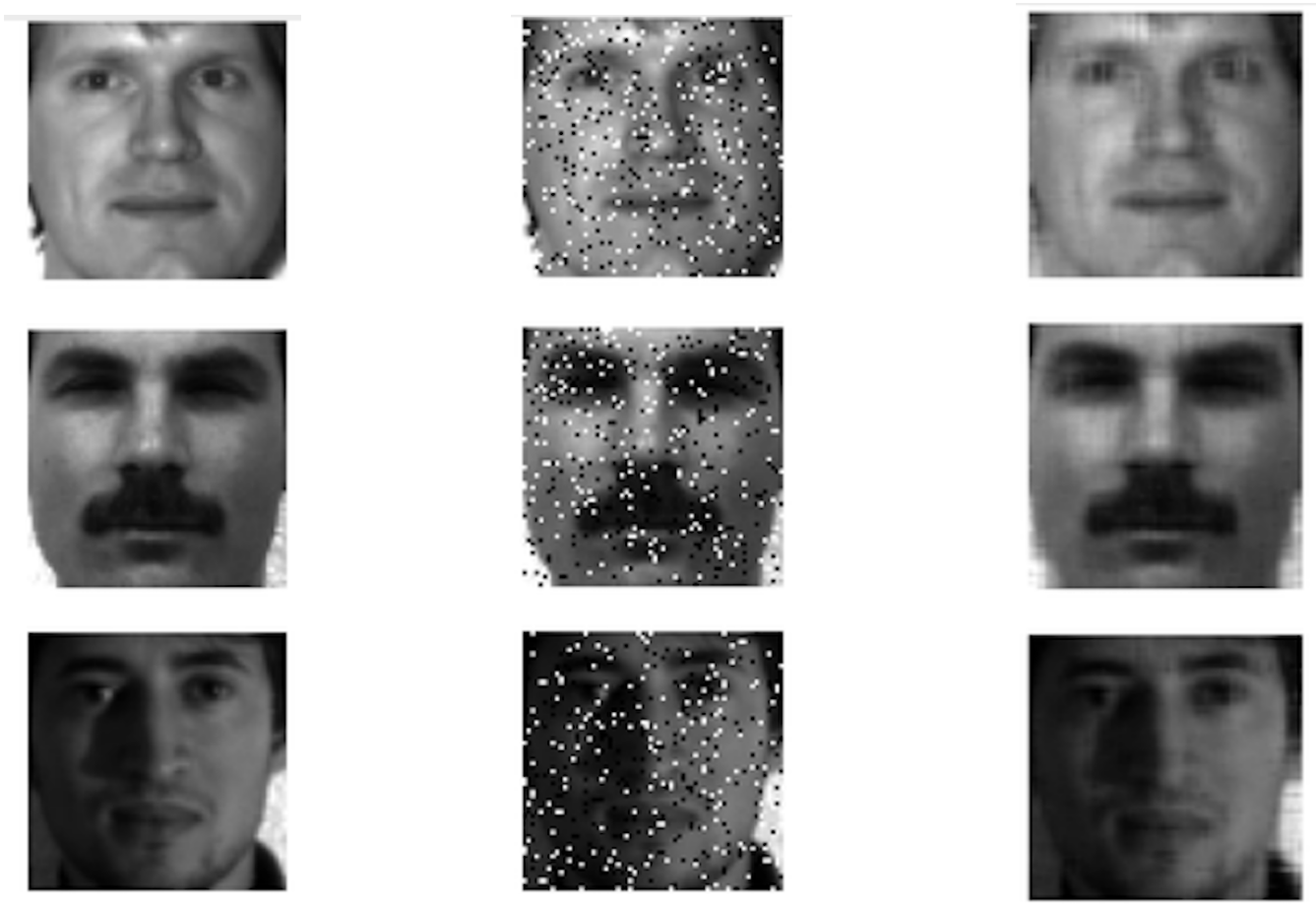


Image from https://medium.com/@amelie_yeh/

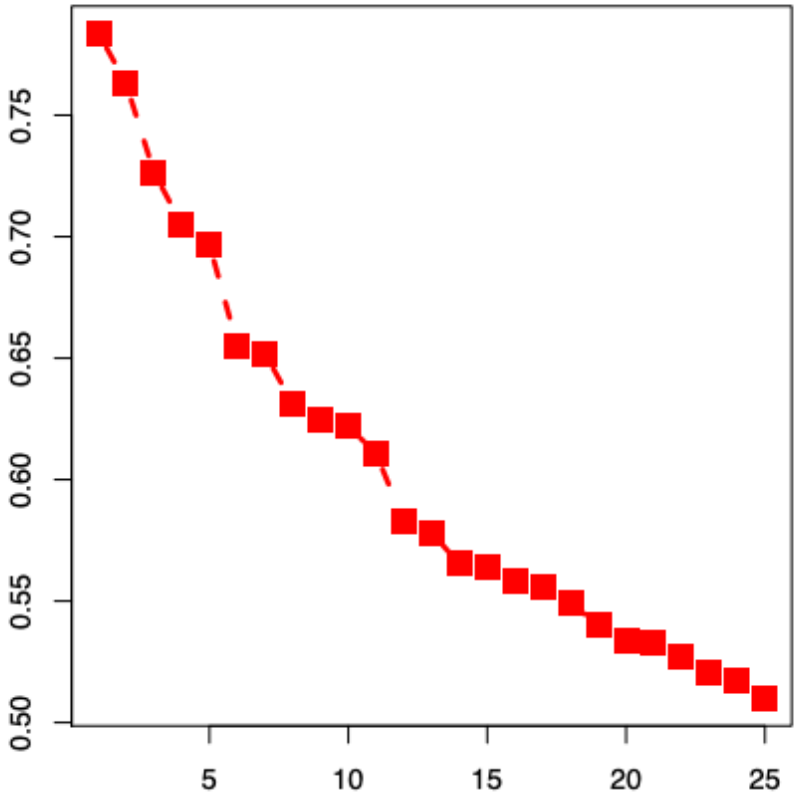
Data is smooth, Noise is jumpy



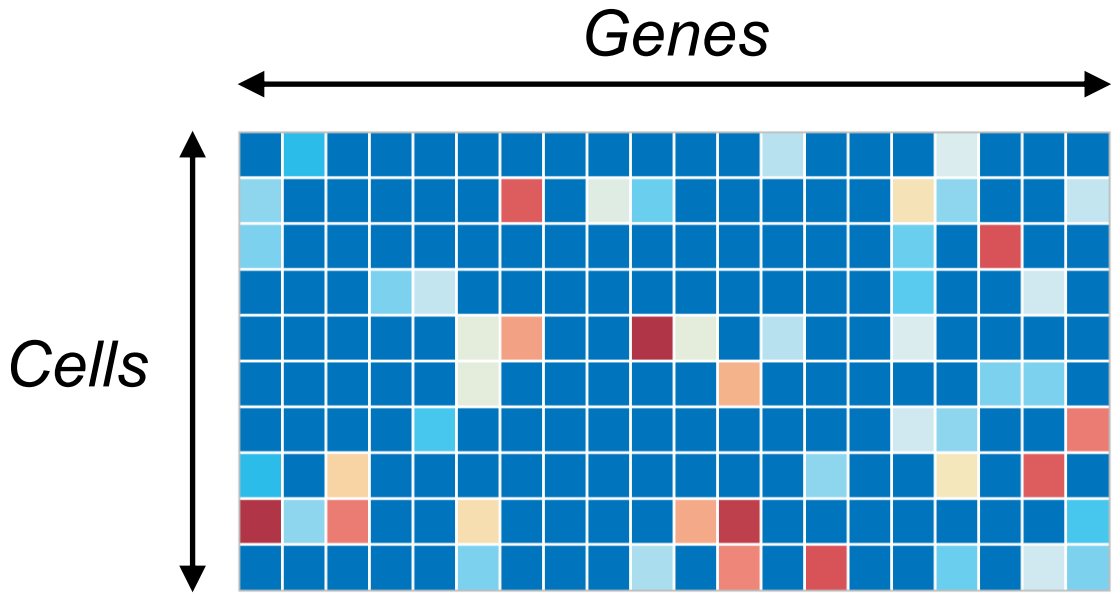
Can take off noise by taking off low-eigenvalued eigenvectors

How many eigenvectors should we use to reconstruct data? How many eigenvectors are noise? How do we know?

Eigengap



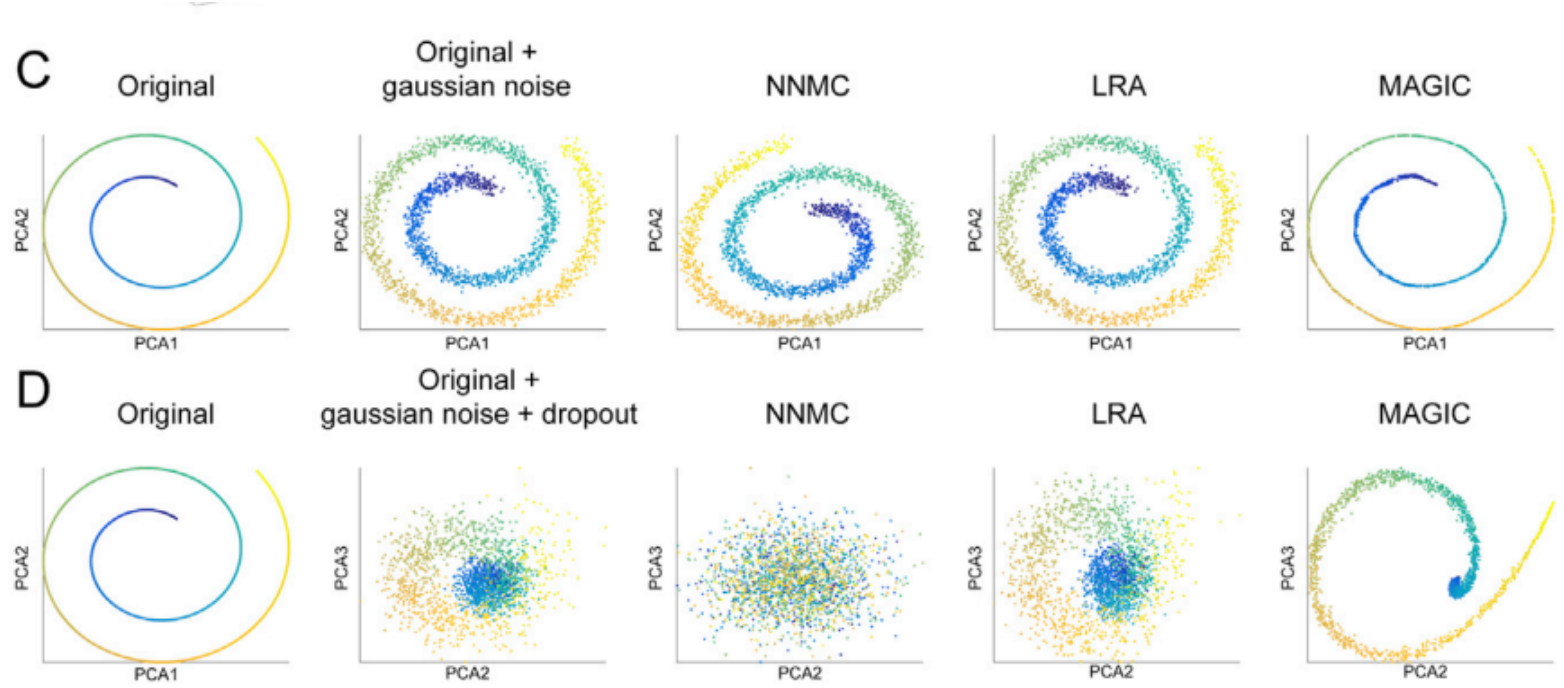
Jumps in eigenvalues
Might give you a clue



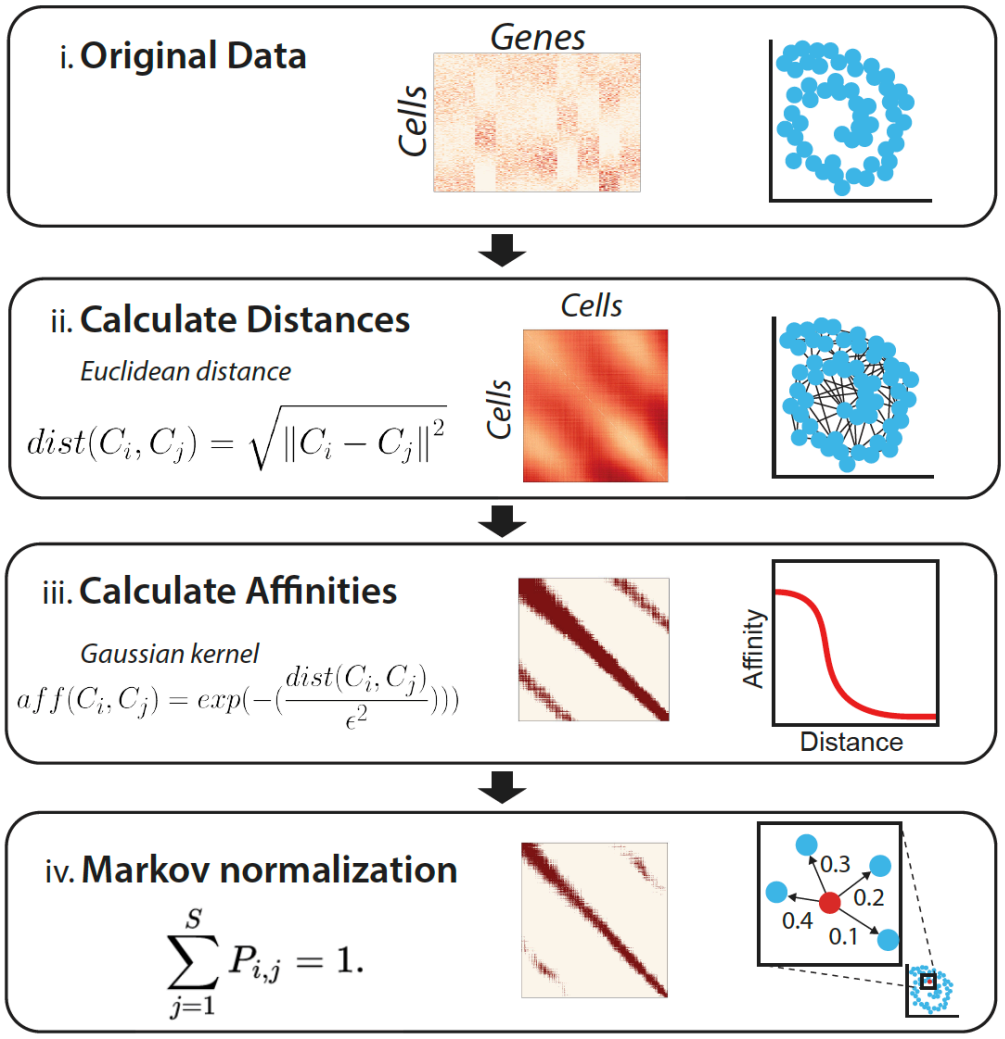
Data is noisy and sparse (scRNA-seq)

Can we use ideas from image denoising to denoise scRNA-seq data?

Low rank approximation for non-linear data

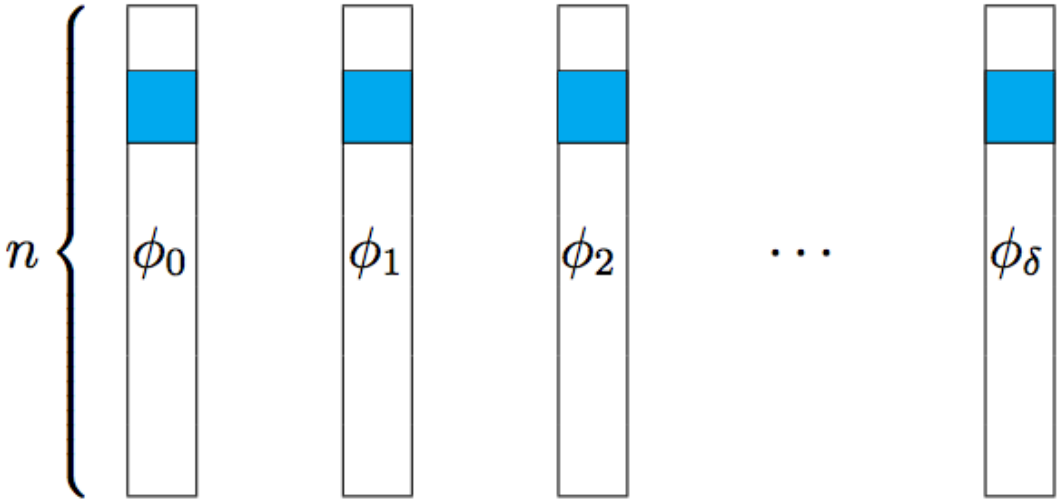


Problem with PCA/Single Value Decomposition (SVD): it takes off linear dimensions in data, noise can be along the non-linear data manifold


i. Original Data
Genes
Cells
ii. Calculate Distances
Euclidean distance
$dist(C_i, C_j) = \sqrt{\|C_i - C_j\|^2}$
Cells
Cells
iii. Calculate Affinities
Gaussian kernel
$aff(C_i, C_j) = exp(-(\frac{dist(C_i, C_j)}{\epsilon^2}))$
Affinity
Distance
iv. Markov normalization
$\sum_{j=1}^{S} P_{i,j} = 1.$
0.3
0.2
0.4
0.1

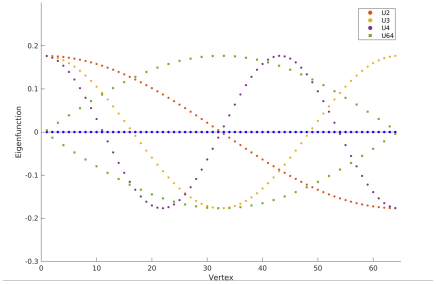
Eigenvectors of Affinity Matrix

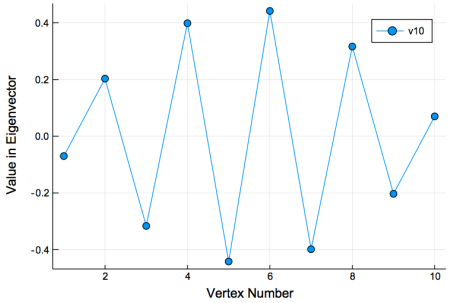
$$1 = \boxed{\lambda_0} \geq \boxed{\lambda_1} \geq \boxed{\lambda_2} \geq \cdots \geq \boxed{\lambda_\delta} > 0$$

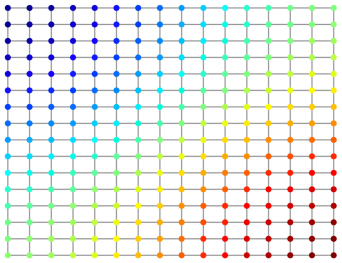


$$x \mapsto \Phi(x) \triangleq [\lambda_0\phi_0(x)\,,\ \lambda_1\phi_1(x)\,,\ \lambda_2\phi_2(x)\,,\ \ldots\,,\ \lambda_\delta\phi_\delta(x)]^T$$

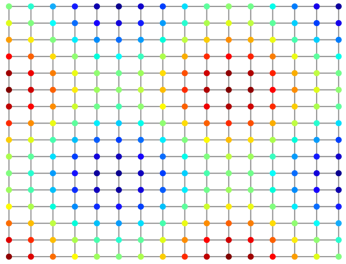
Eigenvectors are frequency harmonics



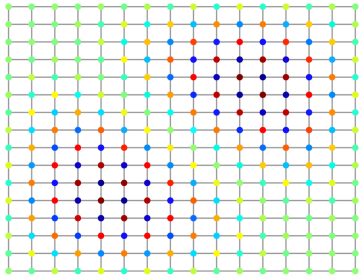




2nd Eigenvector

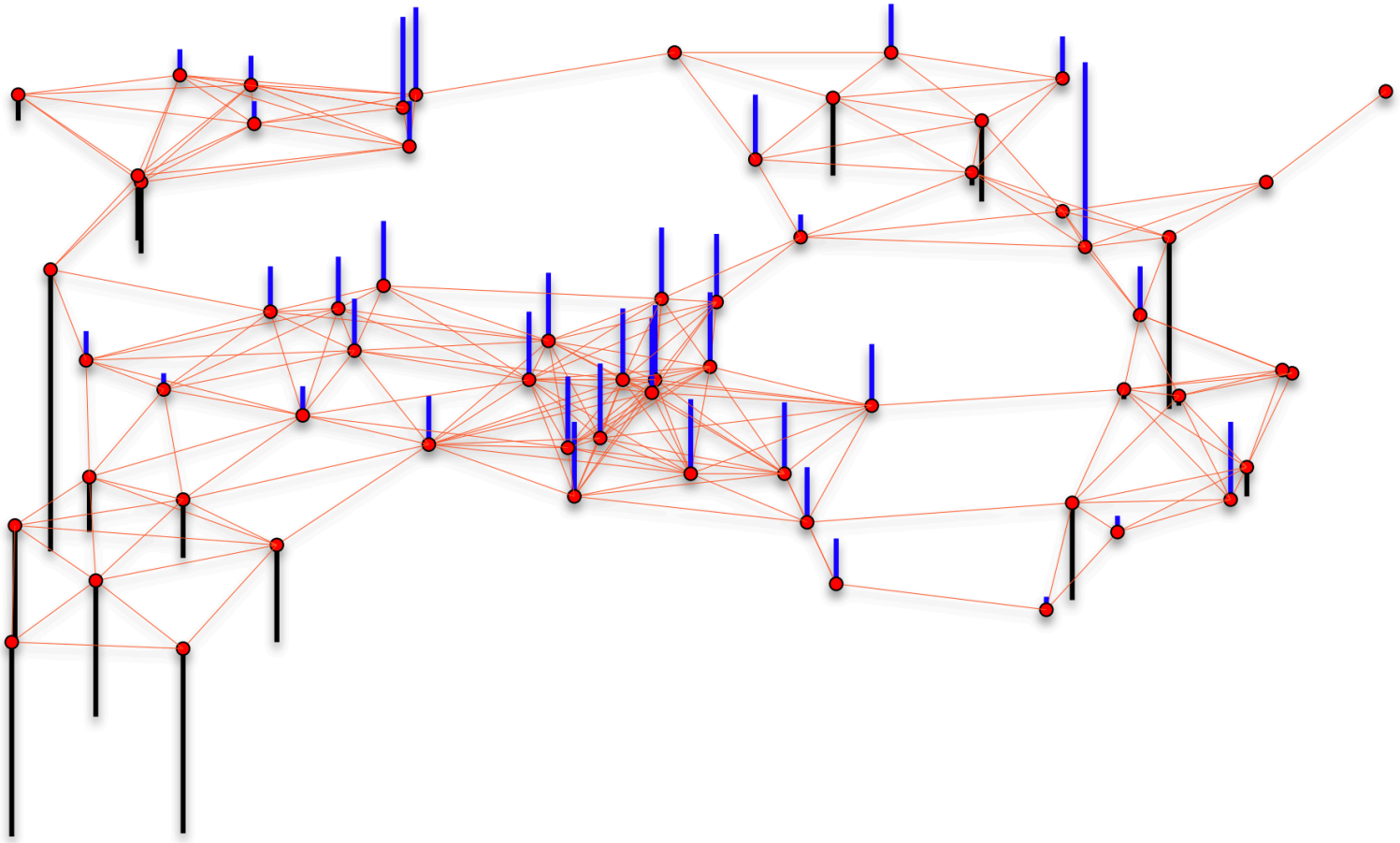


10th Eigenvector

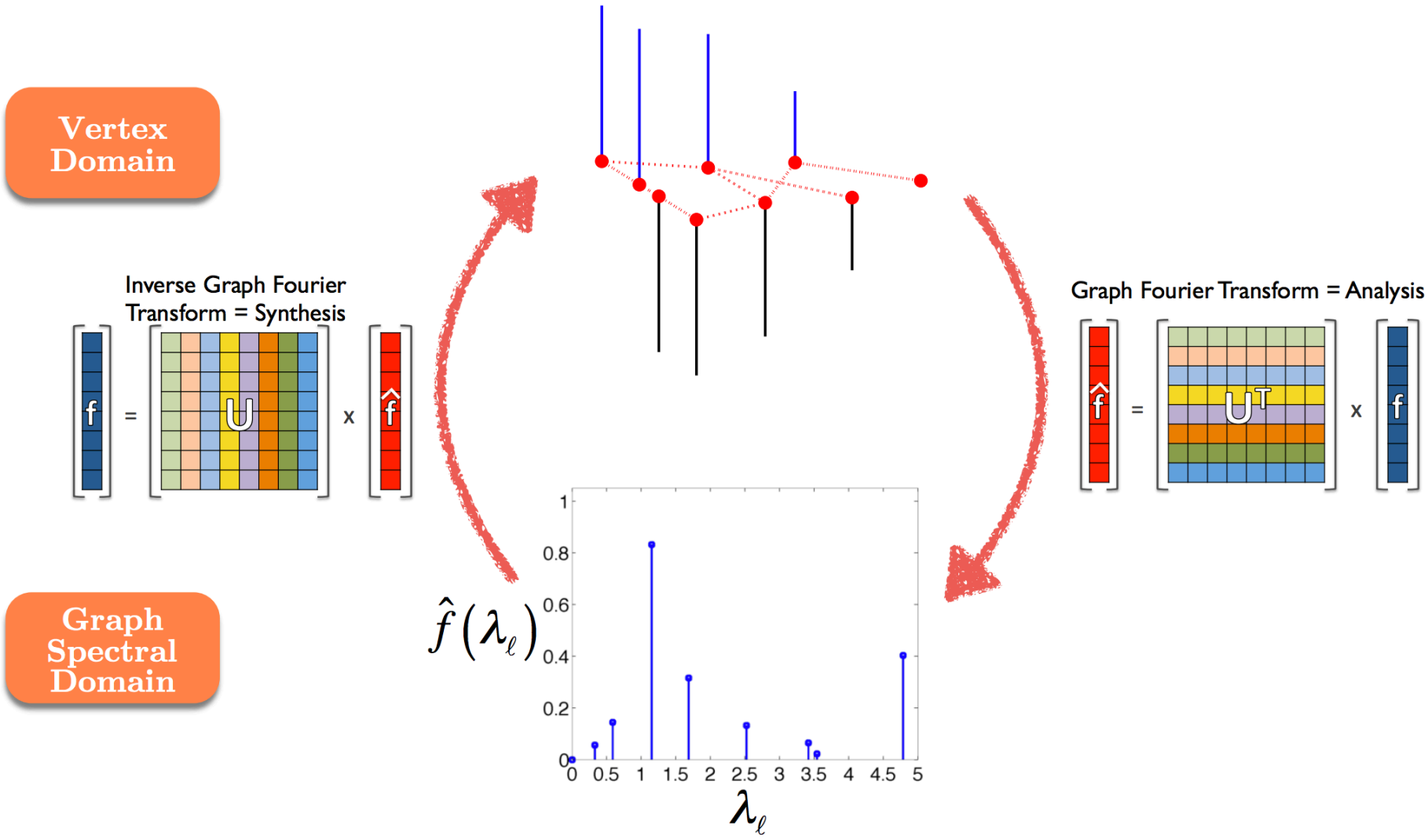


2nd to last eigenvector

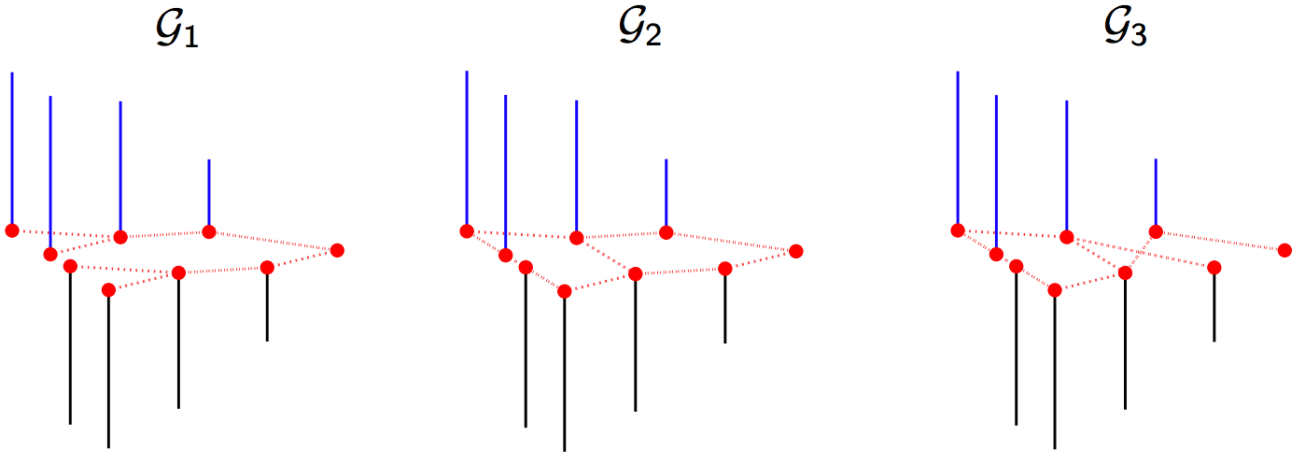
Cells are nodes, mRNA gene-counts are signals on a graph



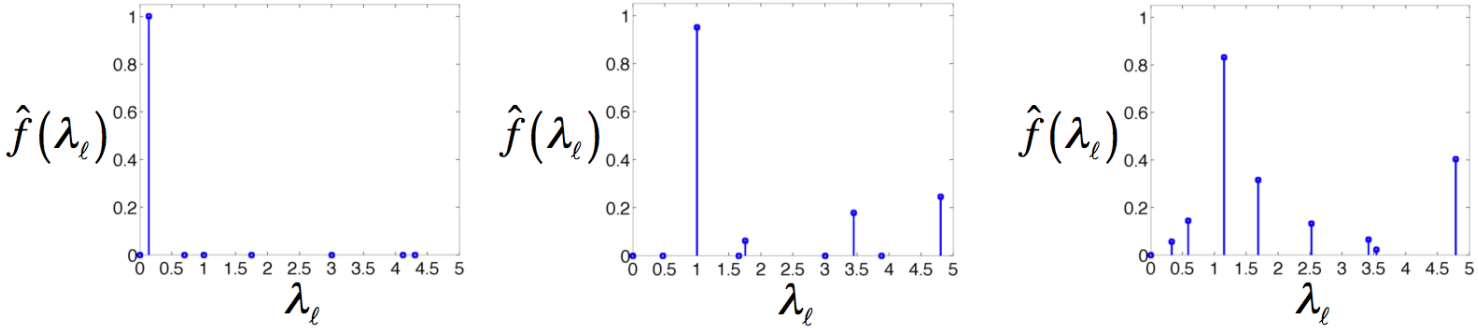
Graph Fourier Transform



Vertex Domain



Graph Spectral Domain

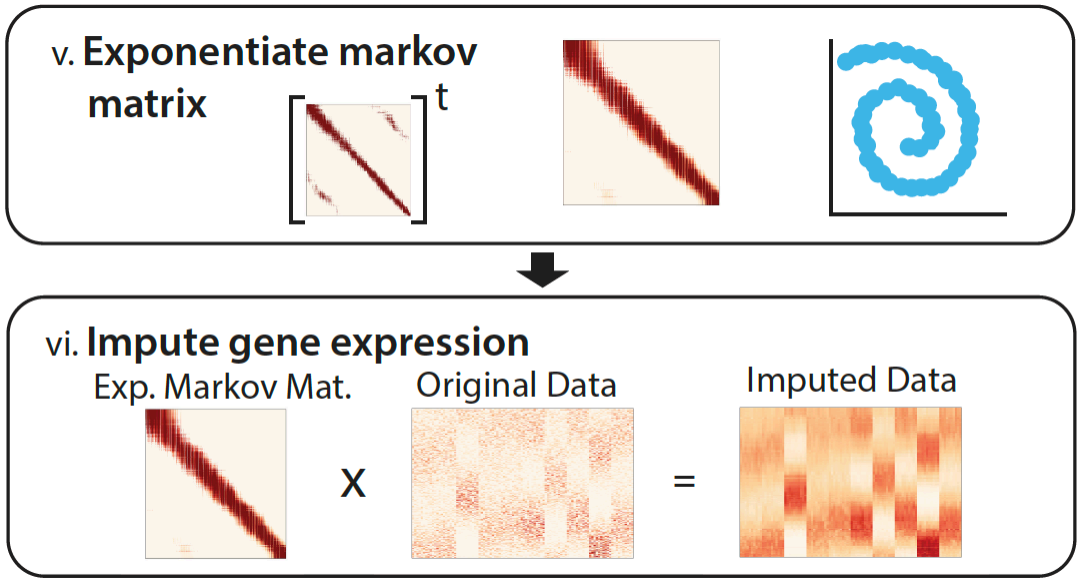


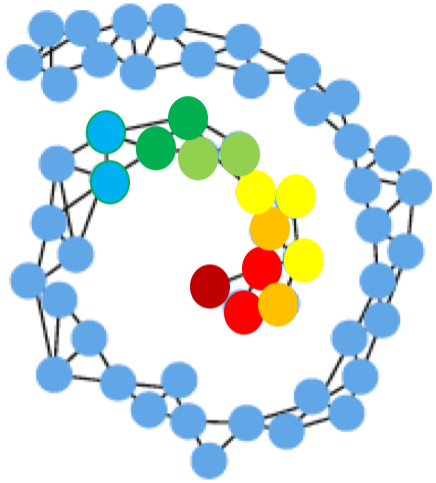
How many eigenvectors should we take off?

MAGIC

- Softly filters eigenvectors, down-weights them in a regular scheme rather than totally taking them off
- This is called low-pass filtering on the spectrum of the graph

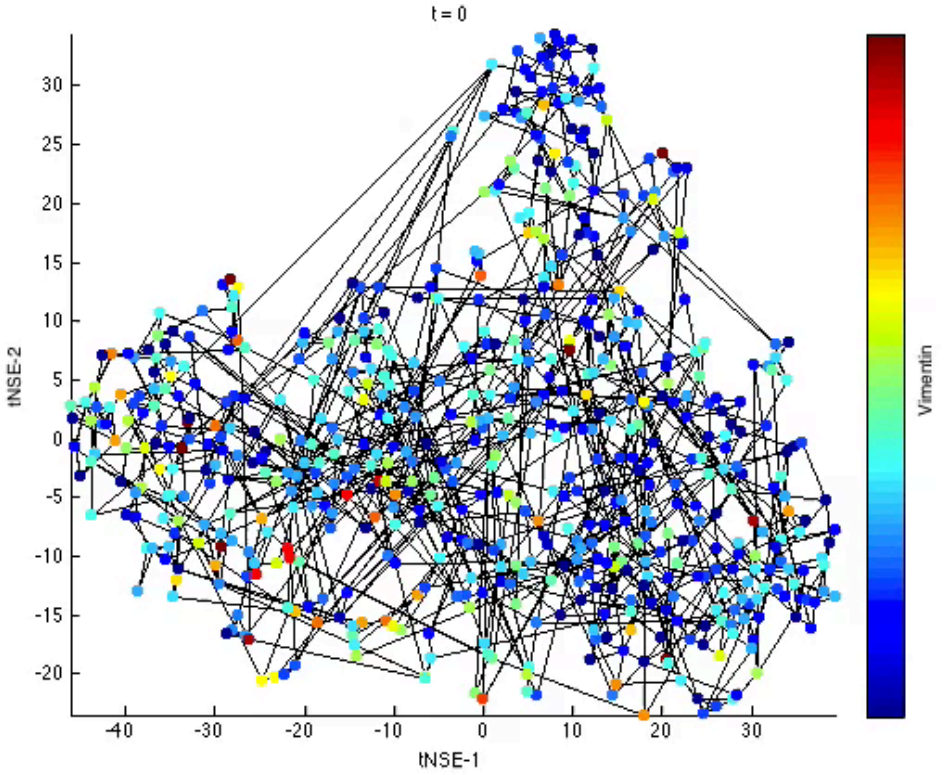
Imputation Step = Smoothing on Graph





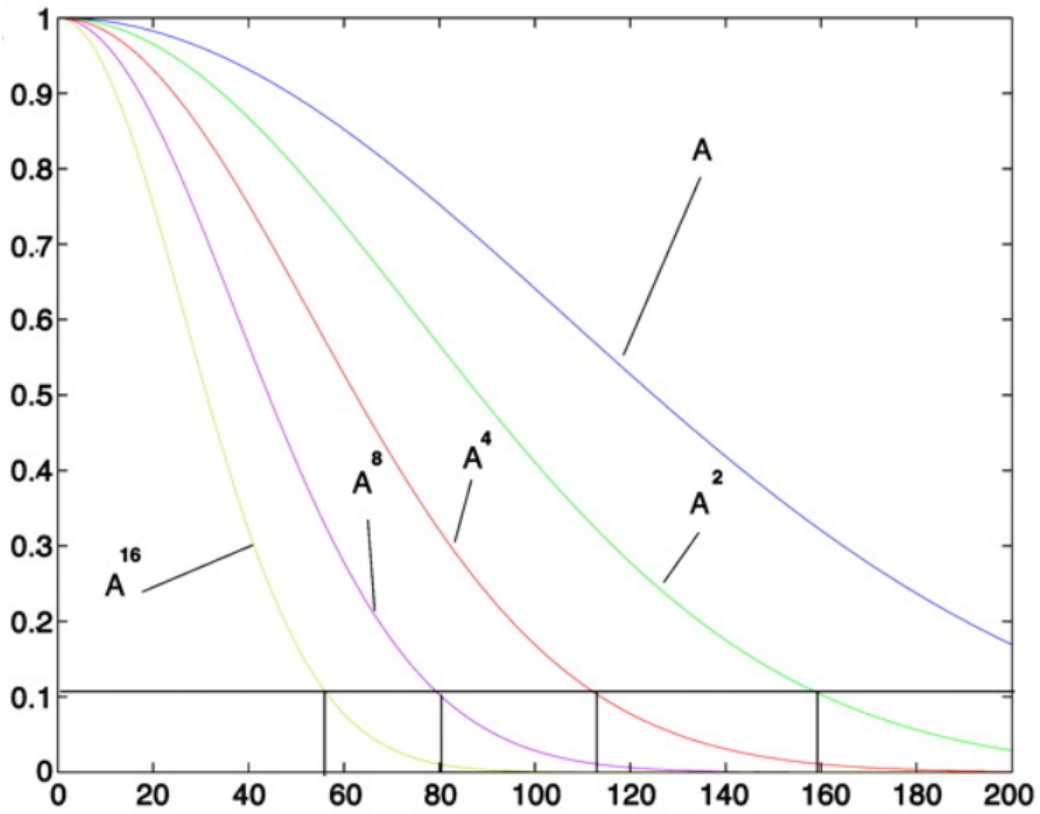
Vertex Domain

- Smooths signal on graph
- Takes weighted average of neighbor

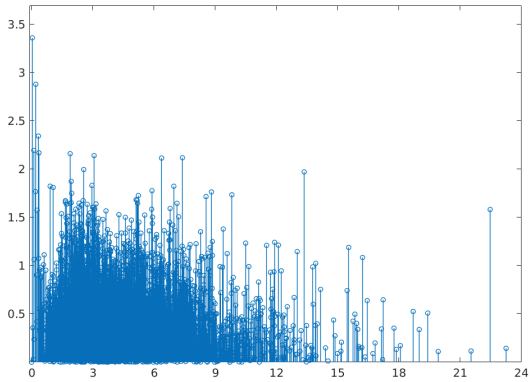


Low pass filter of Eigenvectors

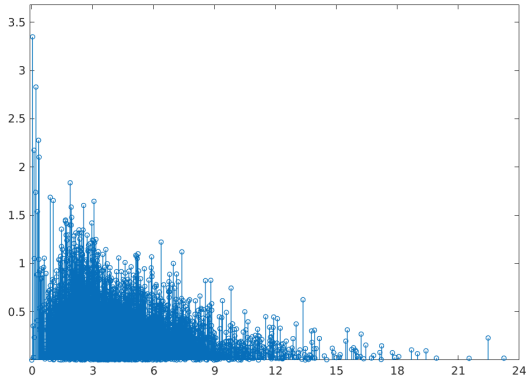
$$\boldsymbol{\Phi}_t(\boldsymbol{x}_i) : \boldsymbol{x}_i \longmapsto \left[\lambda_1^t \phi_1(i), \lambda_2^t \phi_2(i), \lambda_3^t \phi_3(i), ..., \lambda_{M-1}^t \phi_{M-1}(i)\right] \in \mathbb{R}^{M-1}$$



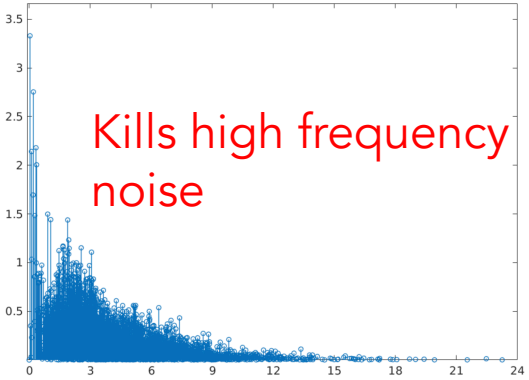
Frequency domain



No Smoothing

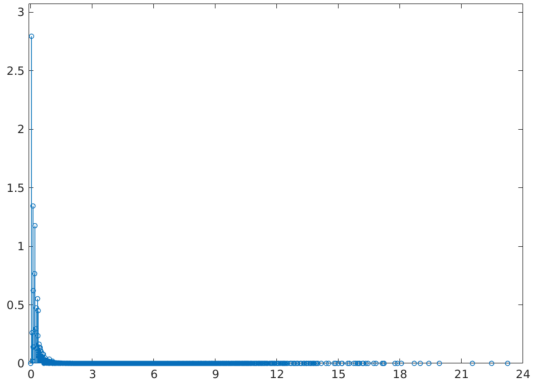


T=2

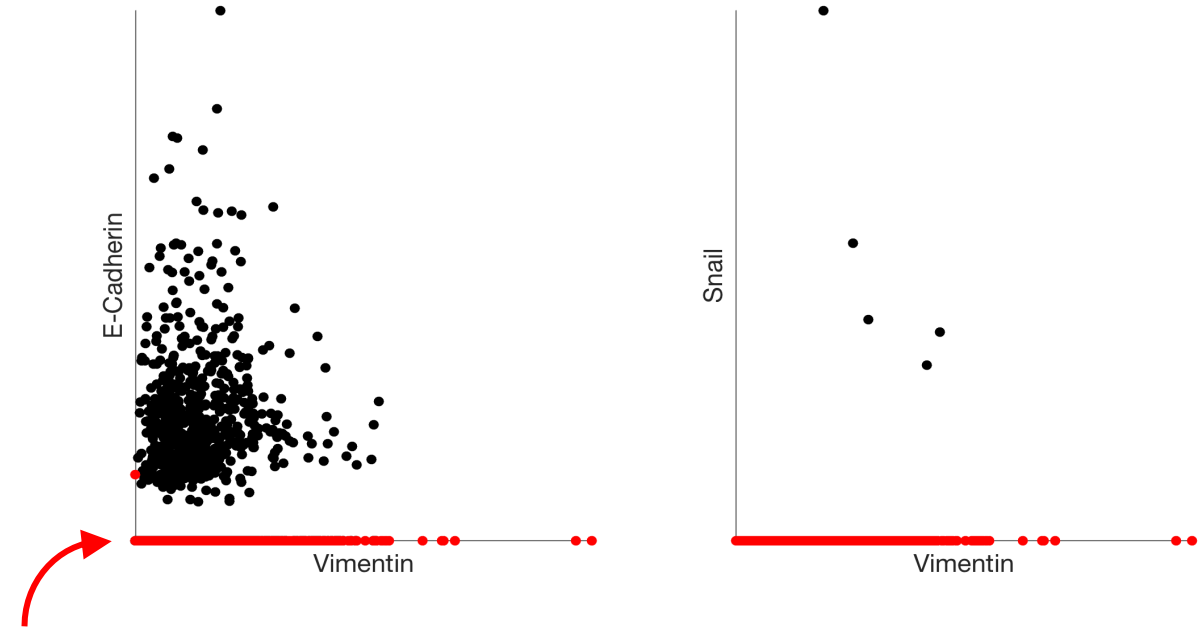


Kills high frequency noise

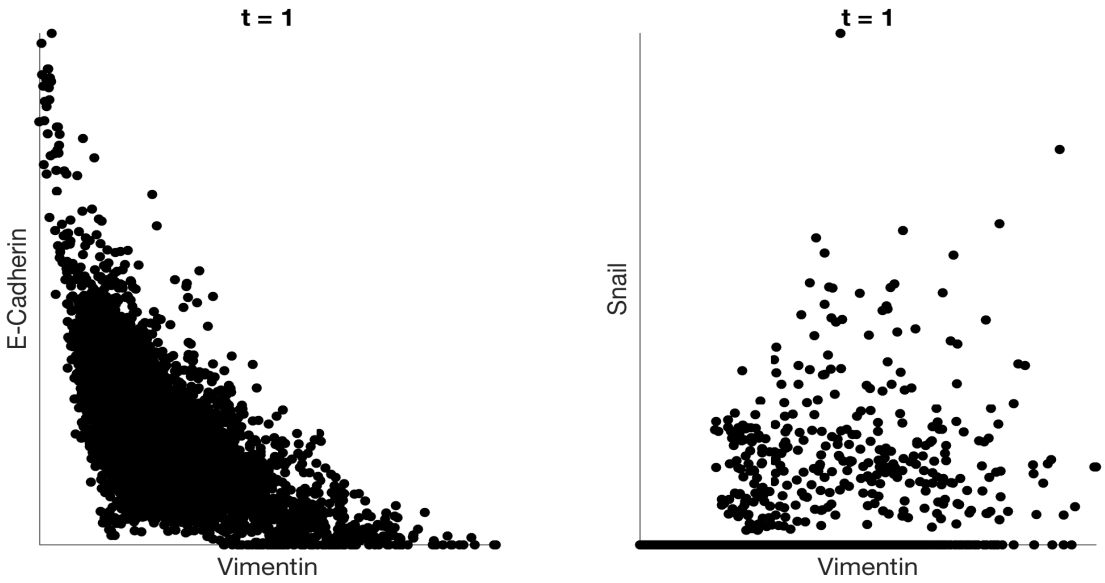
T=5



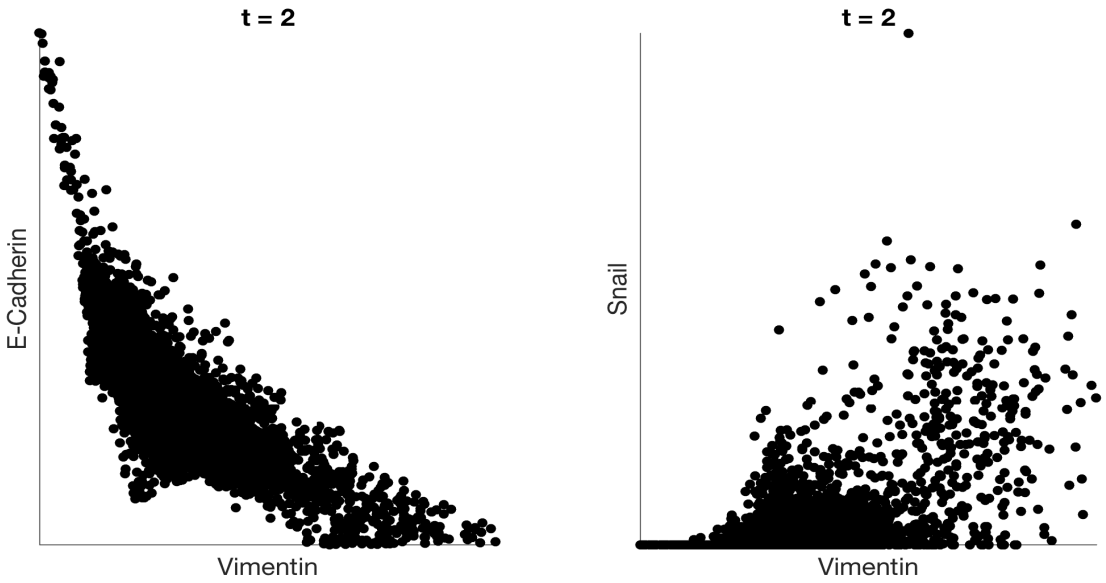
T=100



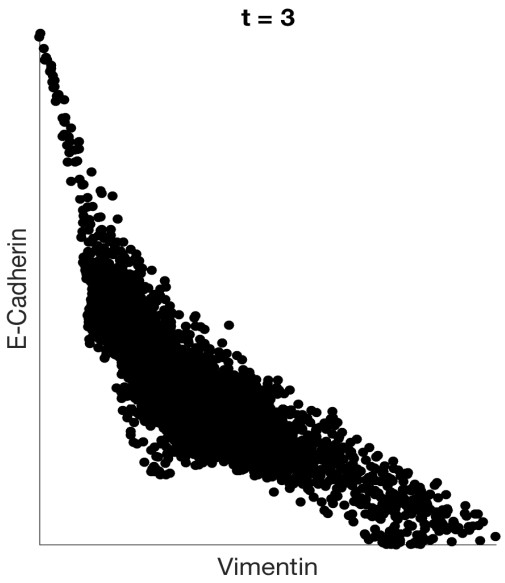
original data with dropout

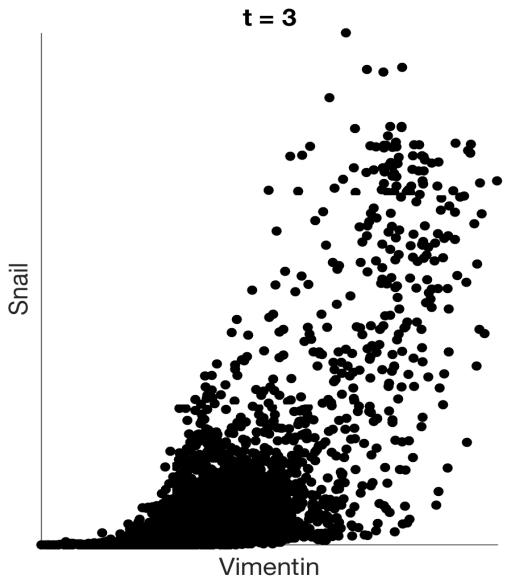


imputation with MAGIC

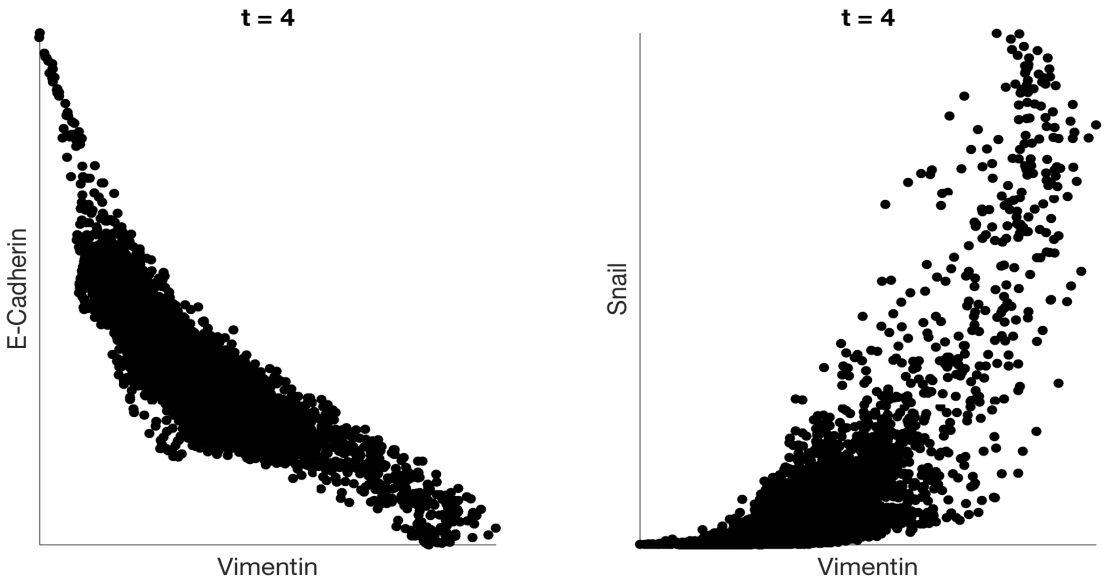


imputation with MAGIC

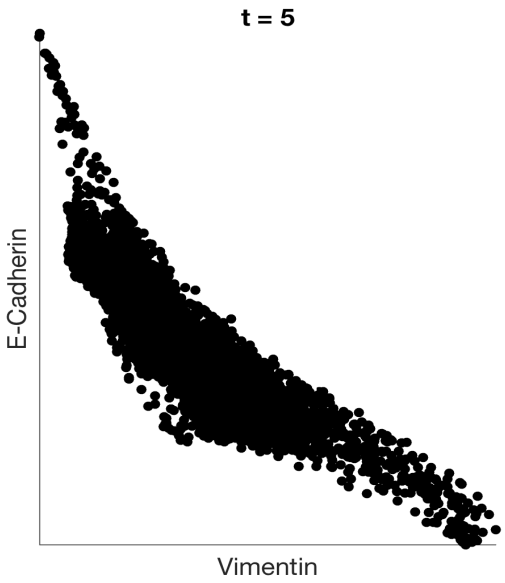


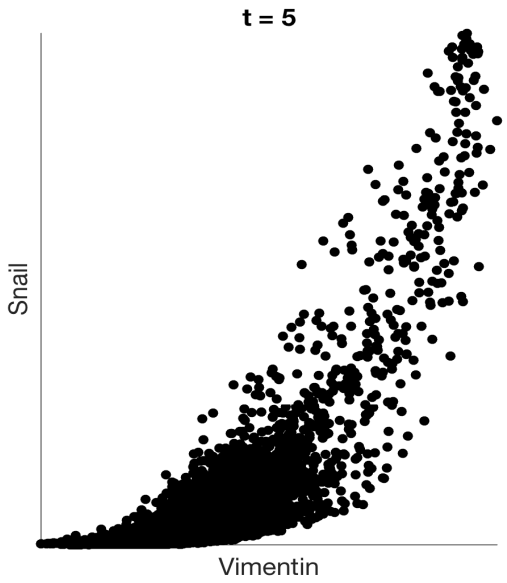


imputation with MAGIC

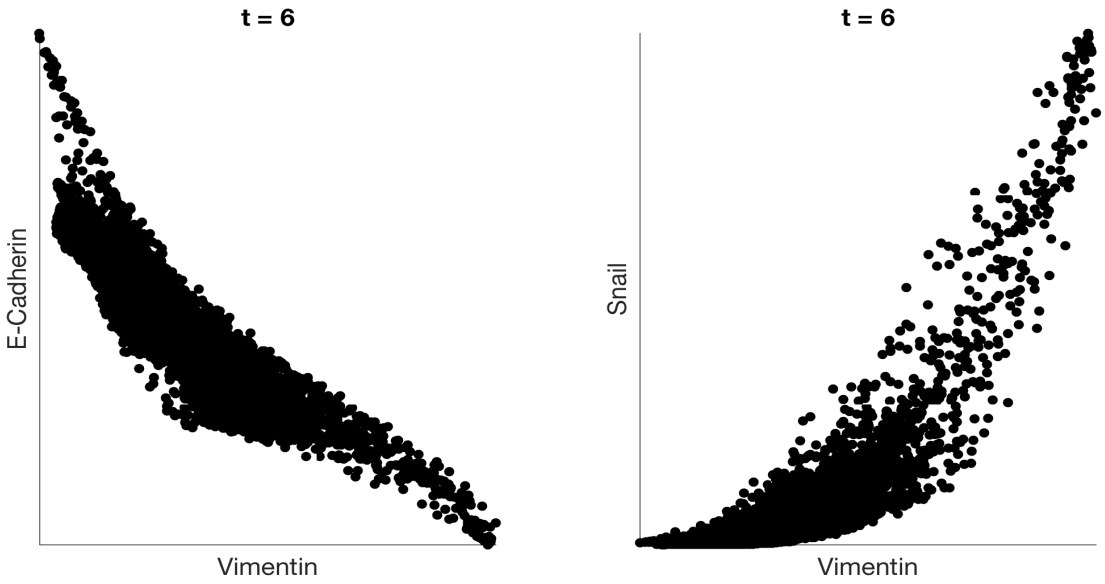


imputation with MAGIC



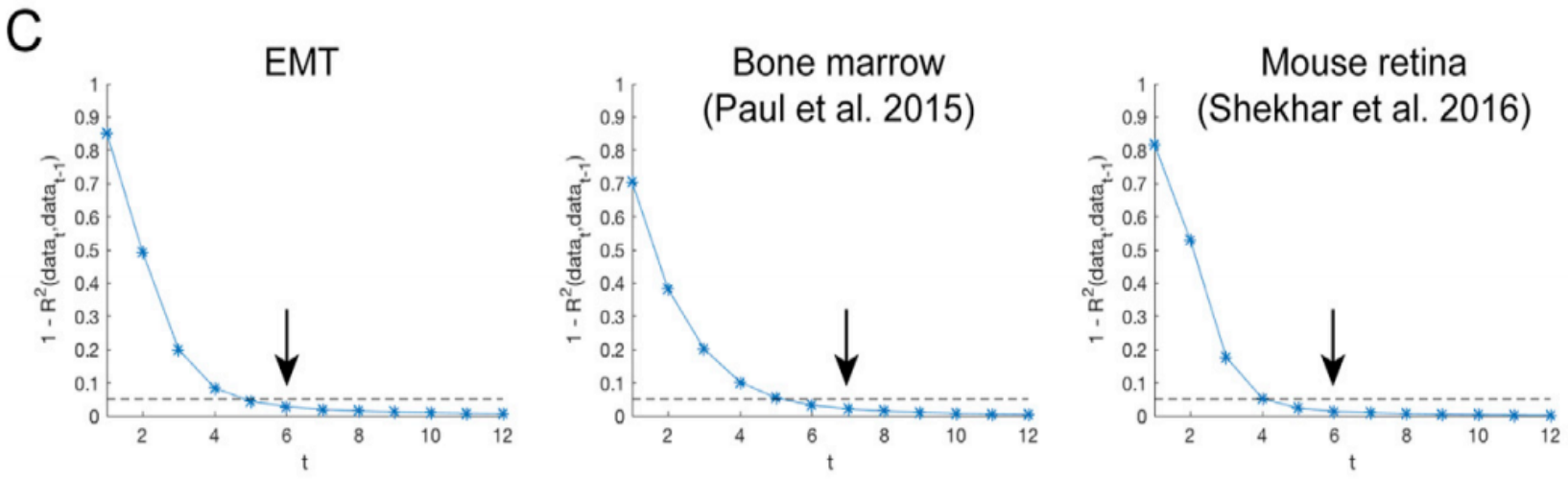


imputation with MAGIC



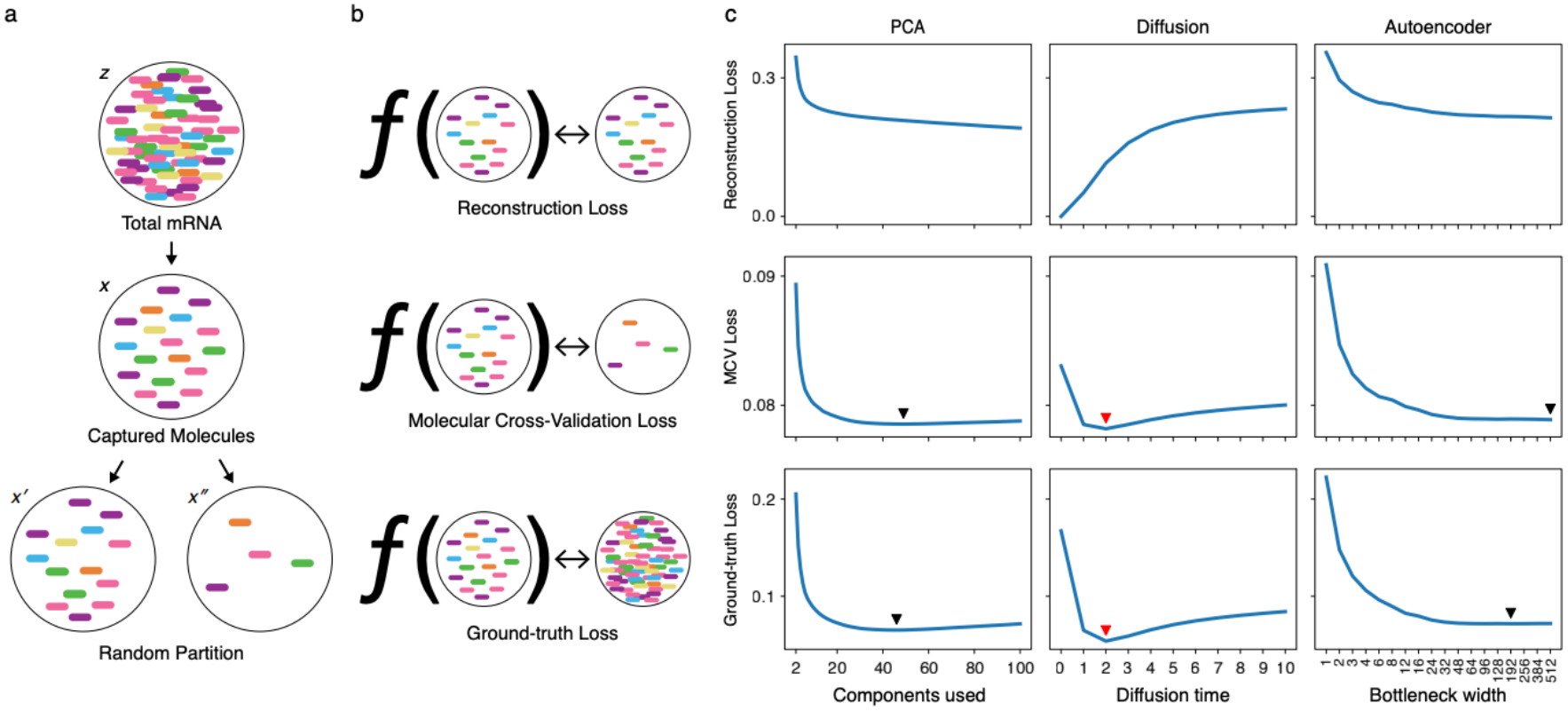
imputation with MAGIC

What t should we choose?



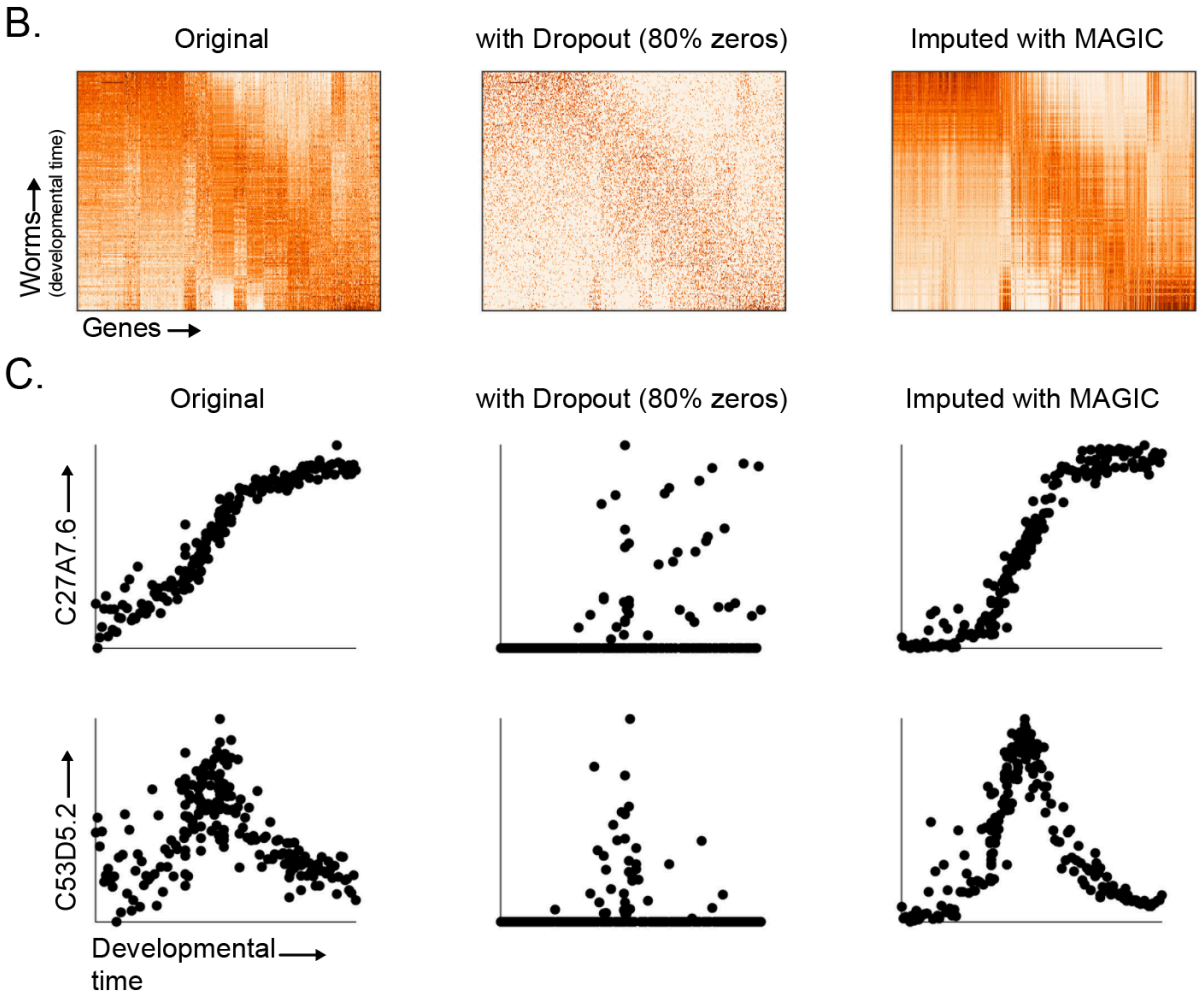
Separate “learning phase” from “stable phase”, quit when data stabilizes

Molecular Validation



https://www.biorxiv.org/content/10.1101/786269v1.abstract

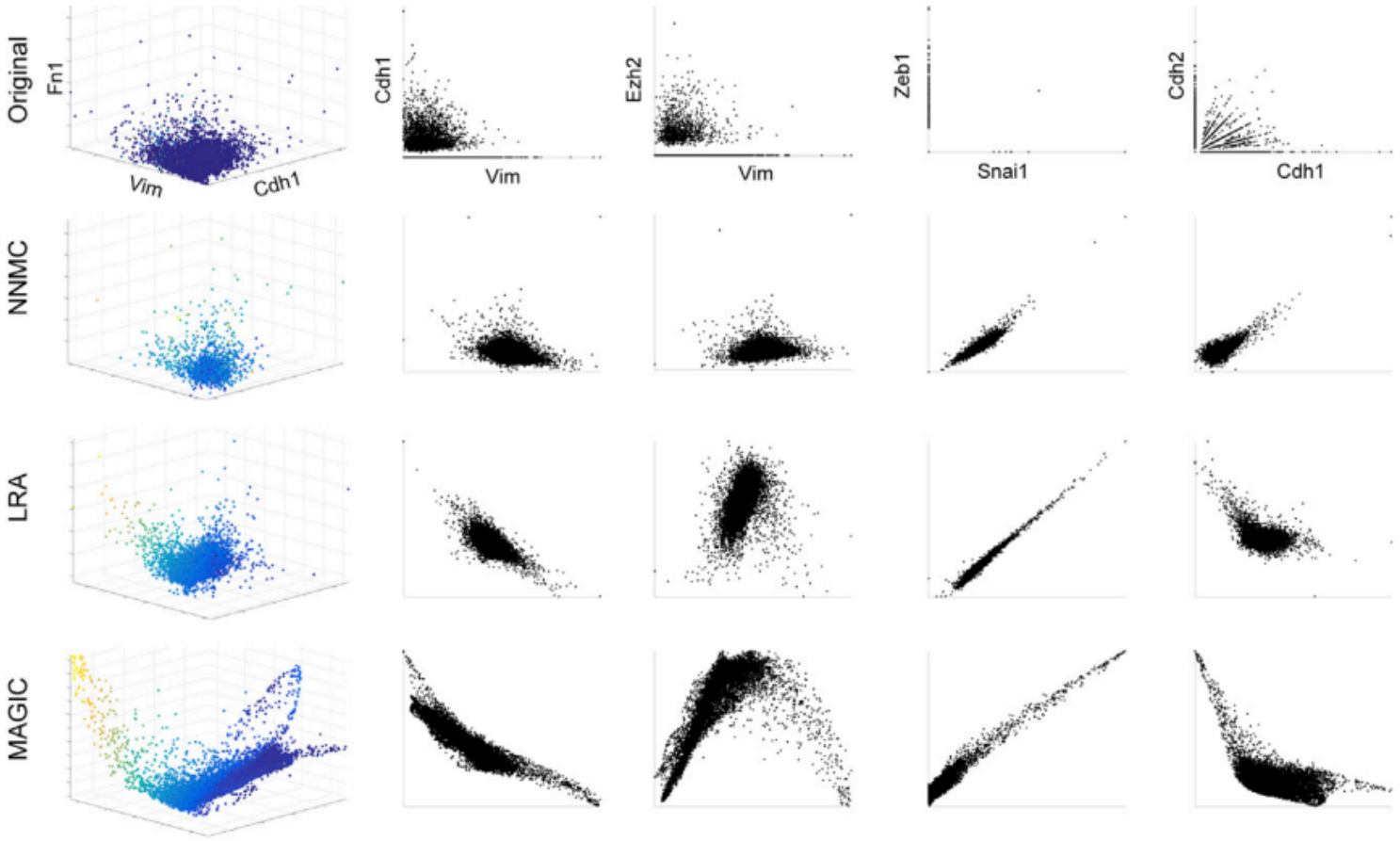
MAGIC recovers gene-gene relationships in an artificially dropped-out dataset



Denoising vs Missing value Imputation

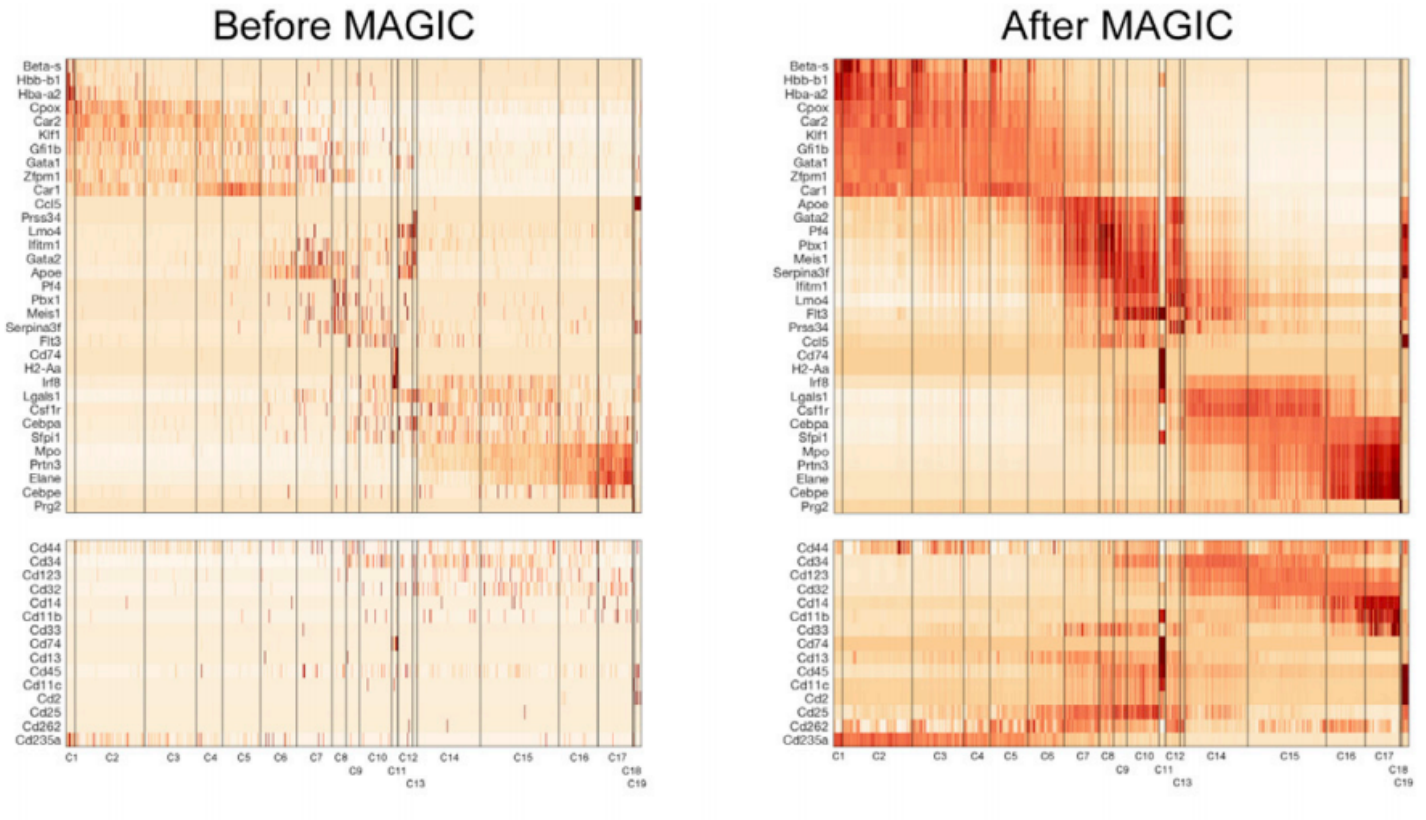
- Denoising assumes that data has some noise, all entries have noise added to them
- Missing value completion methods assume that the existing values are correct and that some values are missing (not measured)
- Matrix completion is a way of filling in missing values
- There is some literature in the single cell world trying to differentiate between true zeros and false zeros
  - People disagree on how this should be done
  - With a denoising viewpoint it is not necessary to distingsuish

Comparing MAGIC to LRA and matrix completion

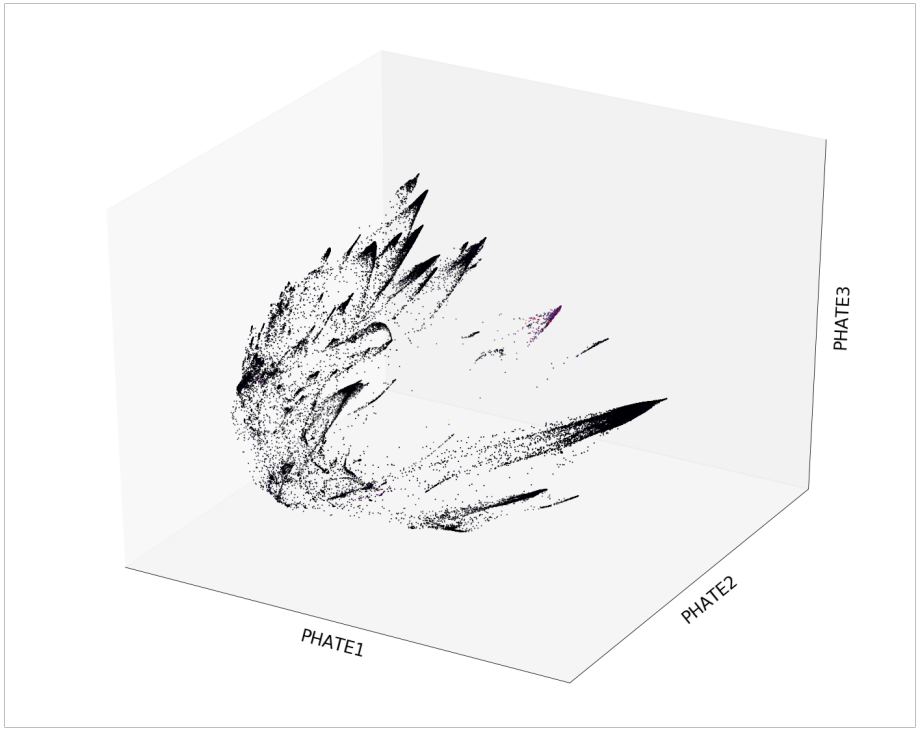


For what tasks is denoising necessary?

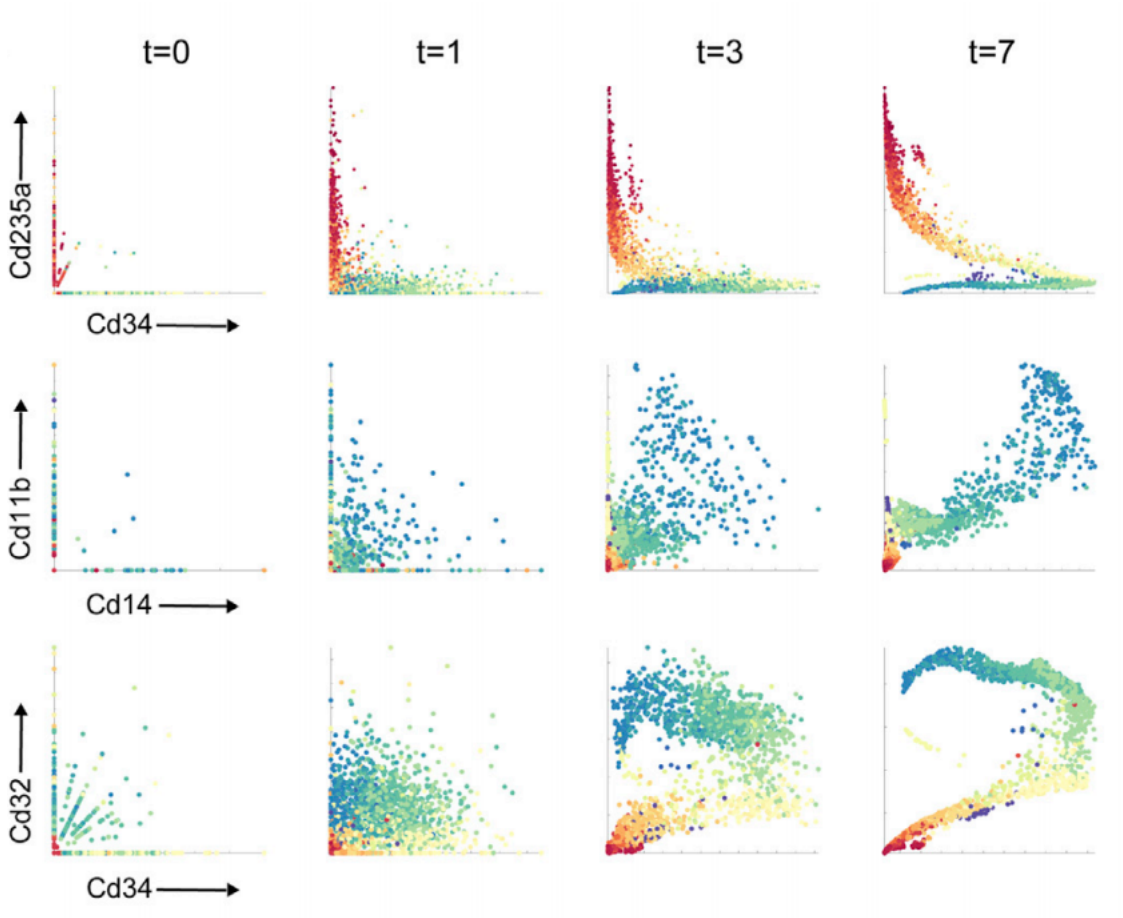
Restoring expressions of characteristic markers



Coloring by MAGIC Expression



Understanding Gene Gene Relationships



Summary of data denoising

- Diffusing or smoothing values over a graph can denoise data
- This kind of denoising is similar to averaging expression values across neighbors
- More diffusion = more denoising

What questions do you have?

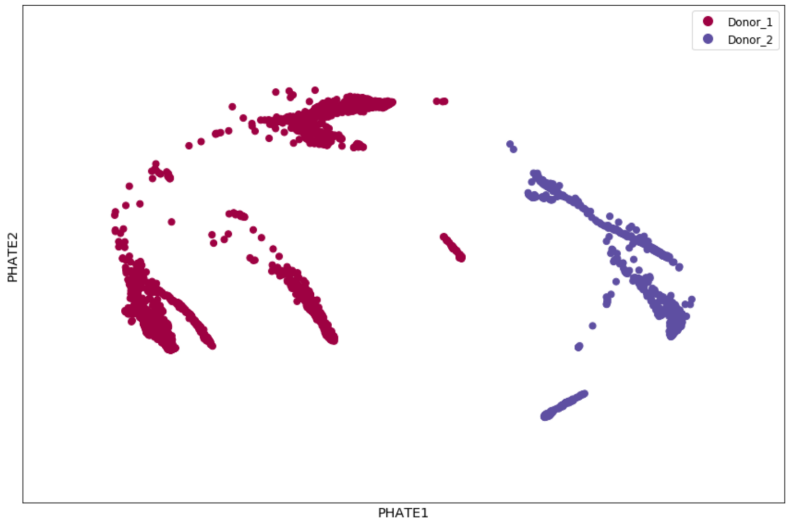
Please submit on Slack

Batch correction

Batch Effects in Single Cell

- **Systematic, non-biological differences between samples due to measurement conditions**
- Differences could be due to ambient conditions (temperature, humidity), machine calibration, differences in titration, bead/antibody batch, etc
- Sometimes refers to actually actual biological differences but those that are “uninteresting” (background demographics rather than immediate drug effect).

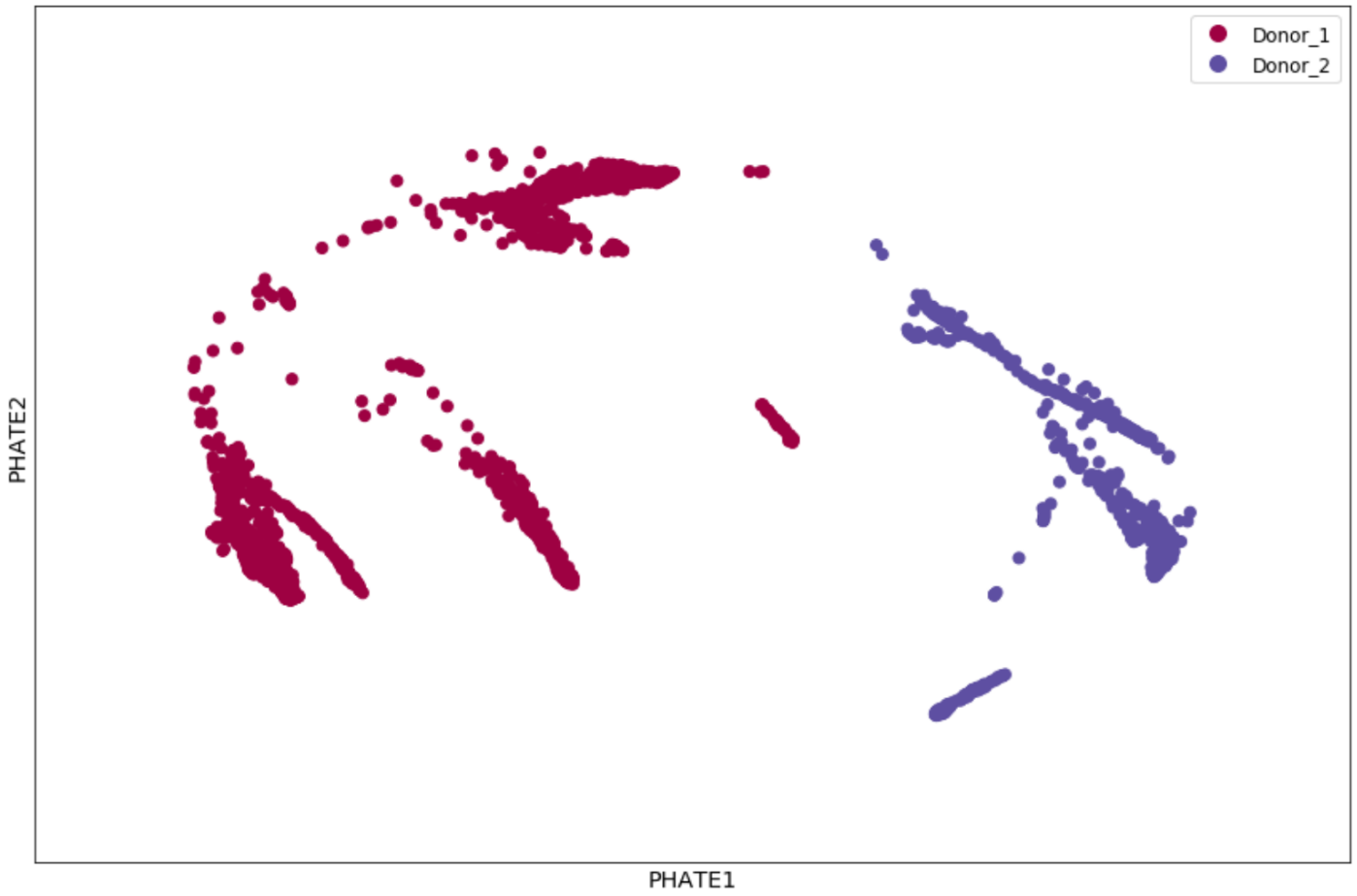
Problem with Batch Effect



Samples become hard to compare

All genes, cell types seem different!

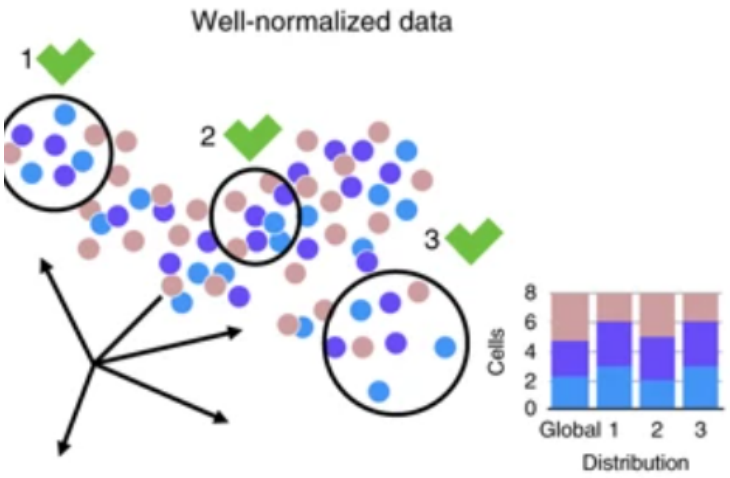
How do we detect batch effect?



Visualization!

Diversity Measures

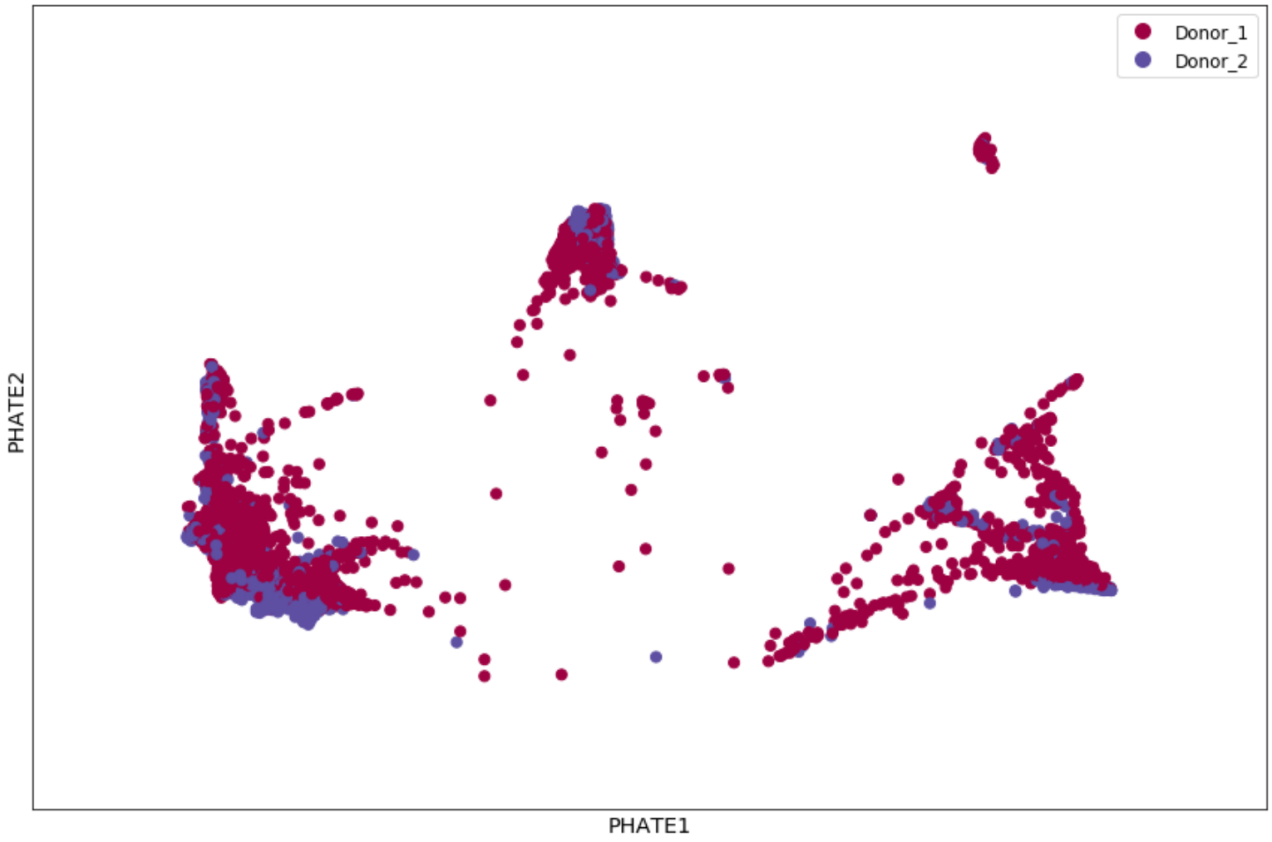
$$H' = -\sum_{i=1}^{R} p_i \ln p_i$$



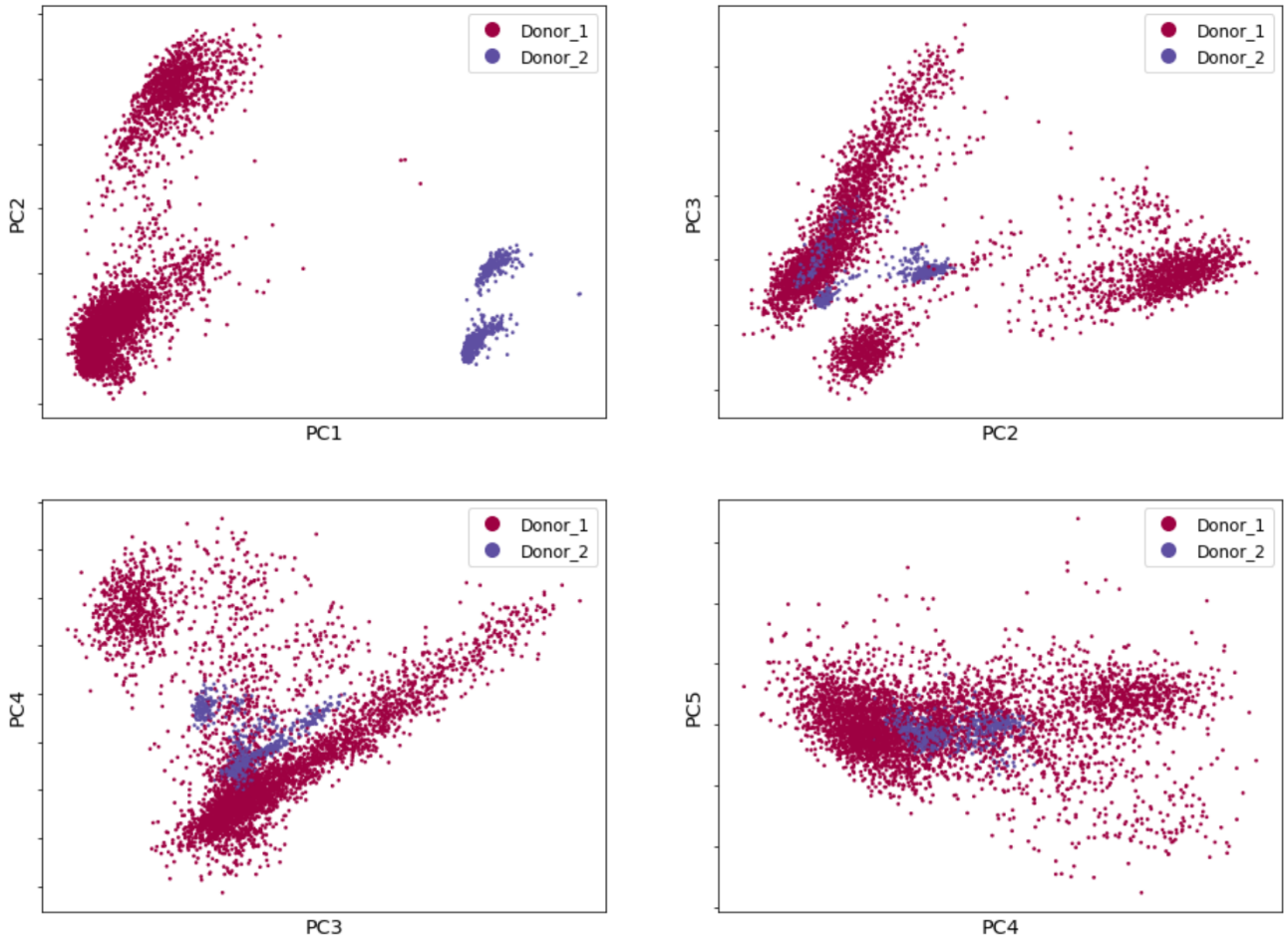
- **Diversity measures including Shannon entropy have been used to quantify the degree of "mixture" between samples**
- kBet [Buttner et al] compare global population proportions to local
- A priori, we don't know what the degree of mixture should be, but if batch correction works, it should increase

Batch Correction

Take off artifactual variation and keep biological variation!



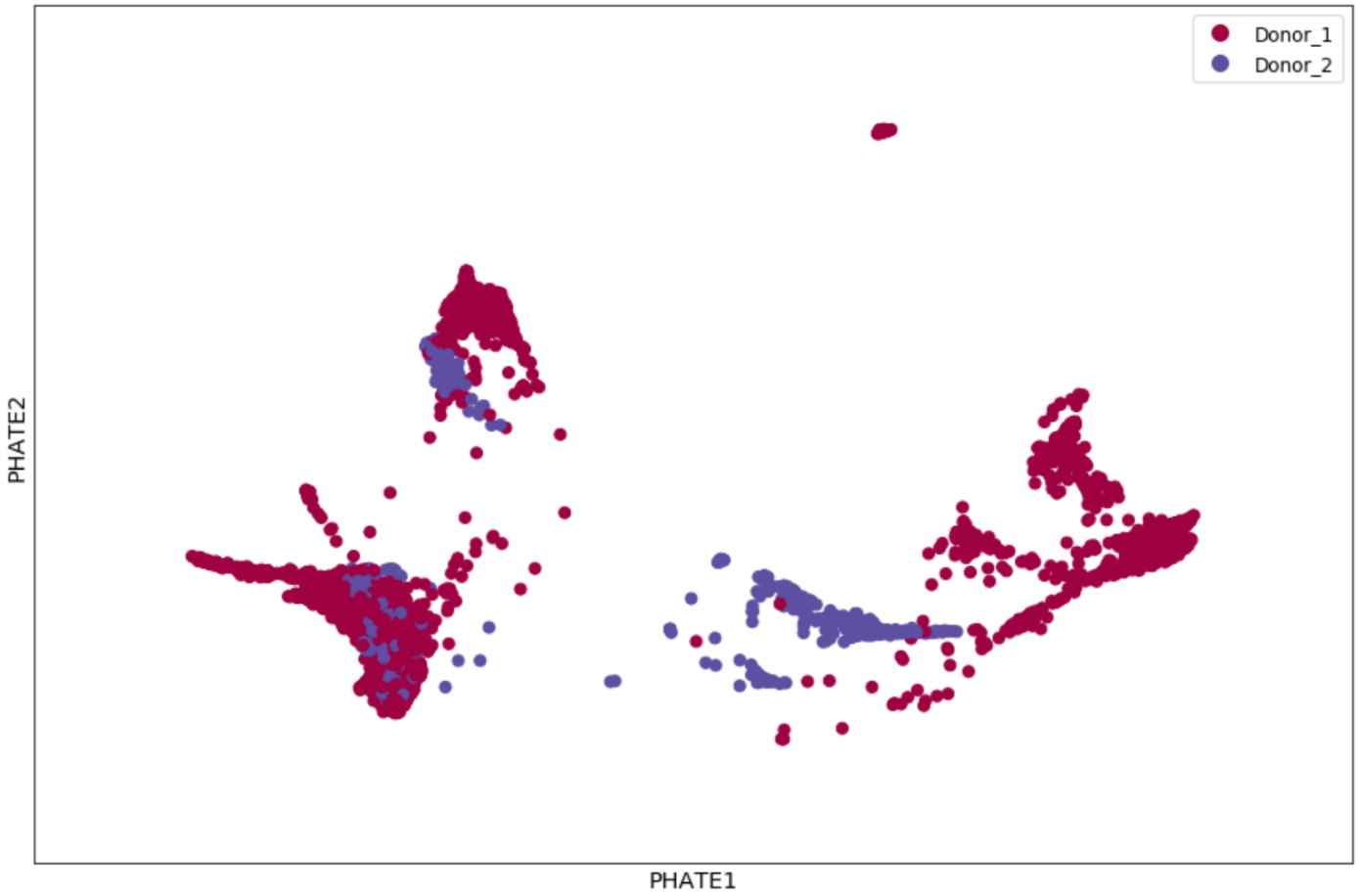
Can we use PCA to correct batch effect?



How could we use PCA or SVD to remove batch effect?

This noise is large scale

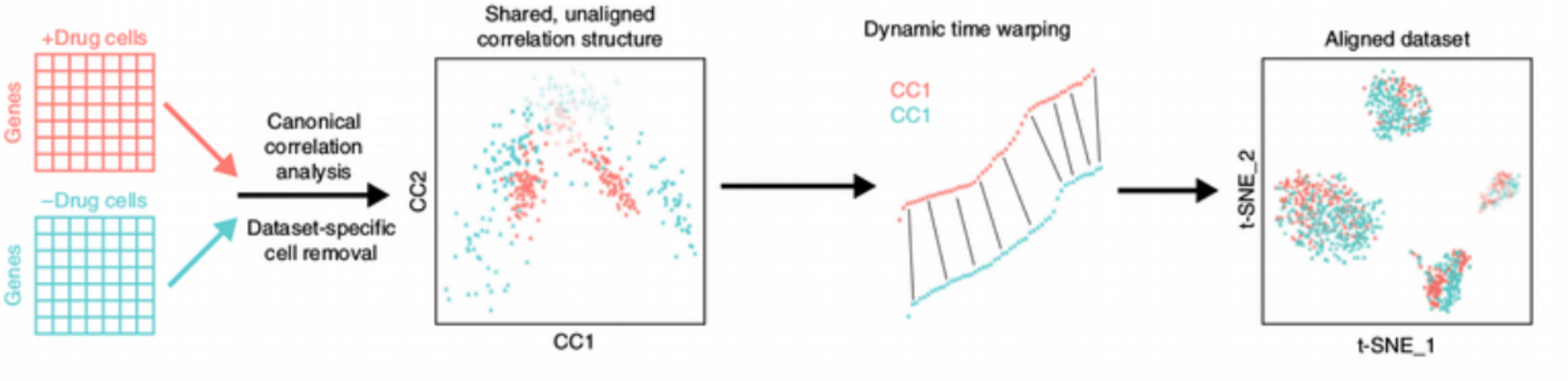
- Could be potentially addressed by removing first PC



Problem with PCA

- It can address a linear shift in the data
- But cannot handle more complex non-linear effects

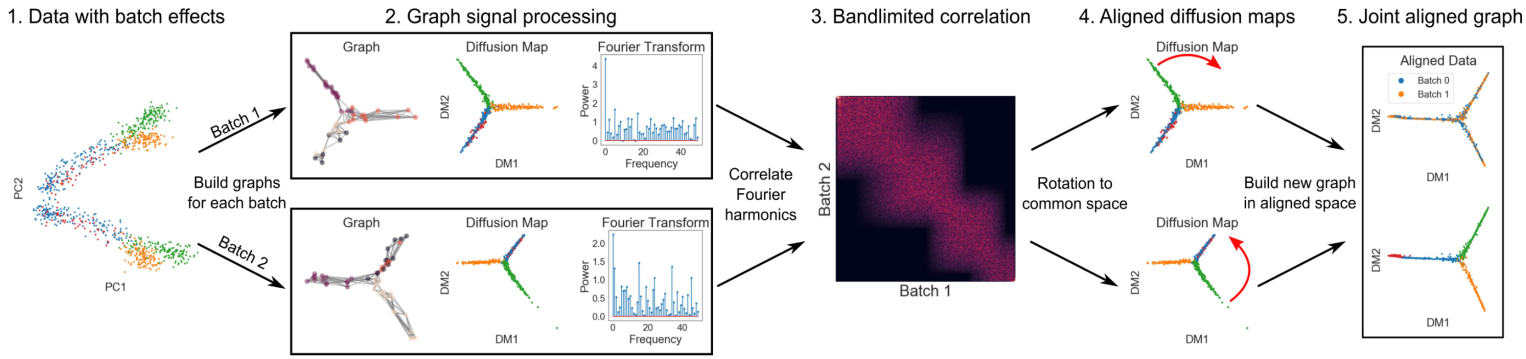
Cannonical Correlation Analysis



CCA : uses a variant of PCA that brings two sets of samples to a common space

- Can cluster or perform other analysis in common space but cannot go back to original space
- Finds linear axes of common variation
- Thrown off by non-matching populations

Harmonic Alignment

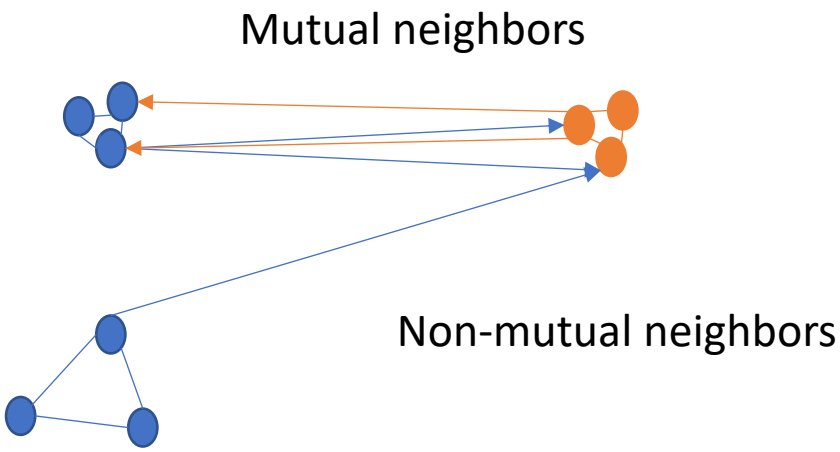


Aligns the diffusion components of two datasets using a rigid rotation that maximizes correlation of gene loadings onto eigenvectors

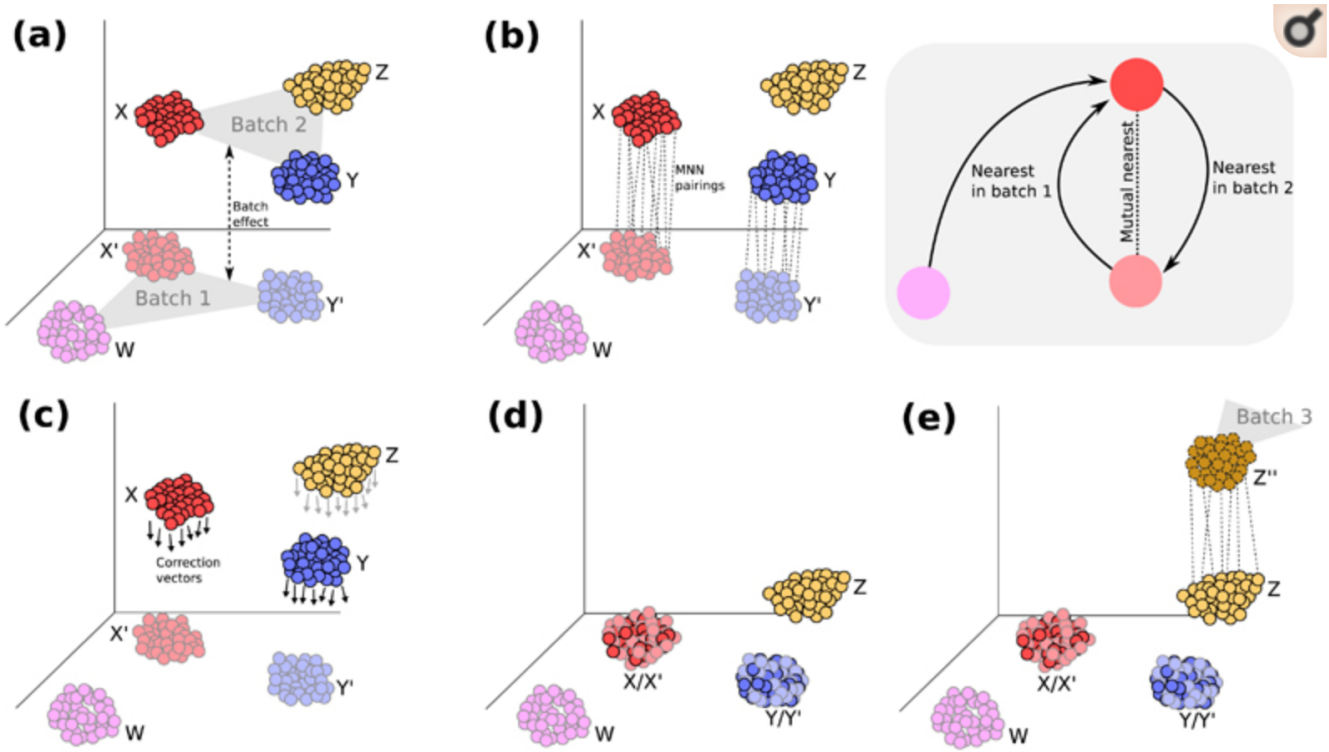
Simple rotation does not seem to handle all cases

Mutual Nearest Neighbors

- Creates a graph between two datasets
- Nearest neighbors in the other dataset could be “matching cells”
- But they have to be mutual!

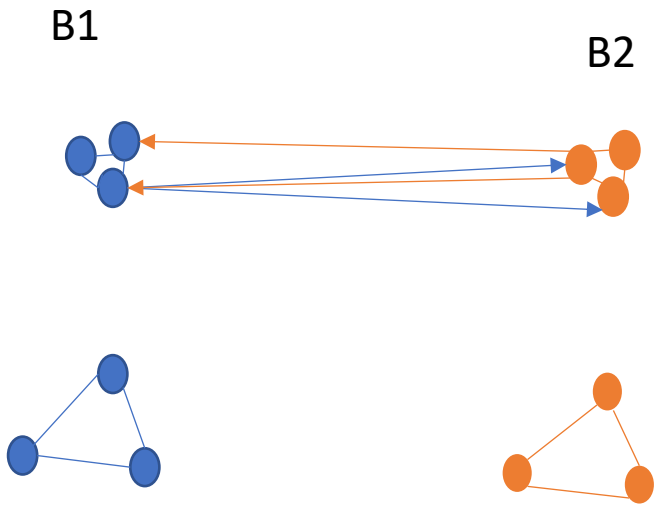


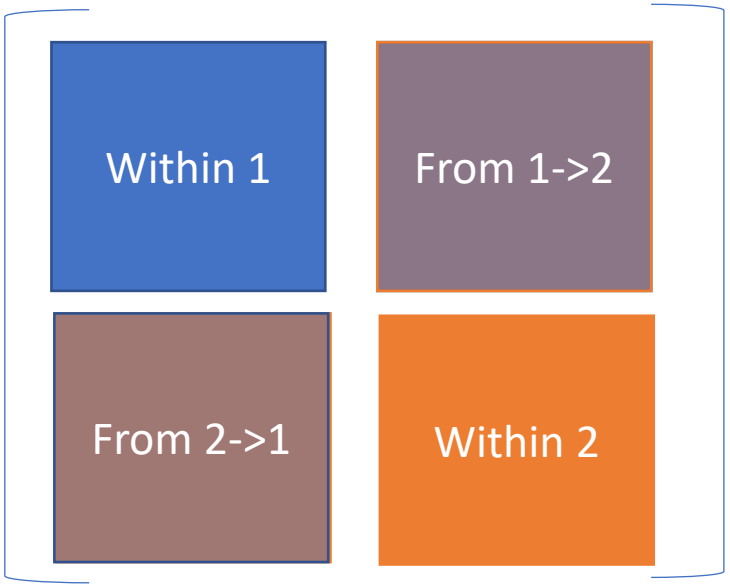
Mutual Nearest Neighbors



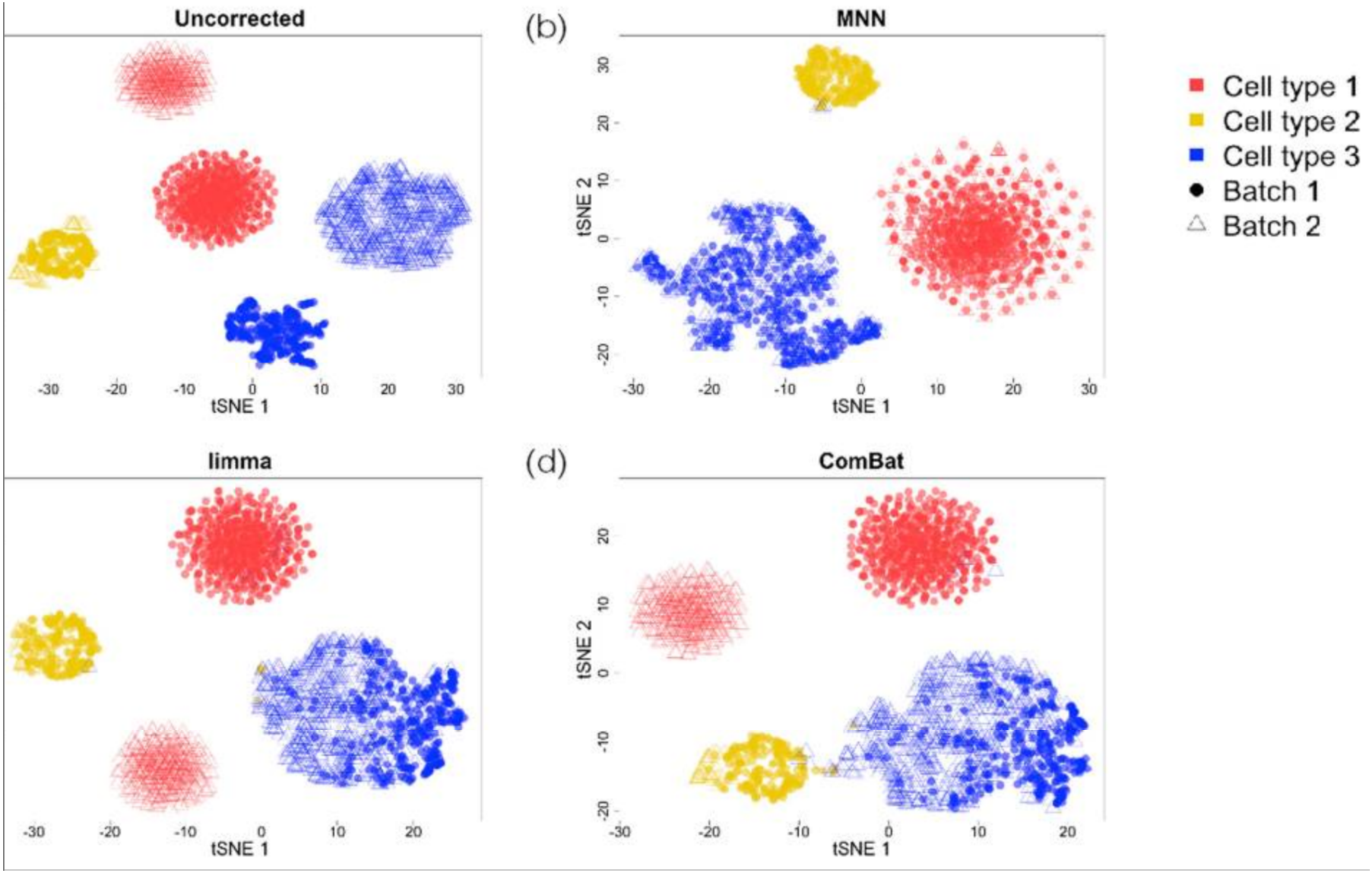
Use MNNs to find out the direction of the batch effect and correct it.

MNN with MAGIC

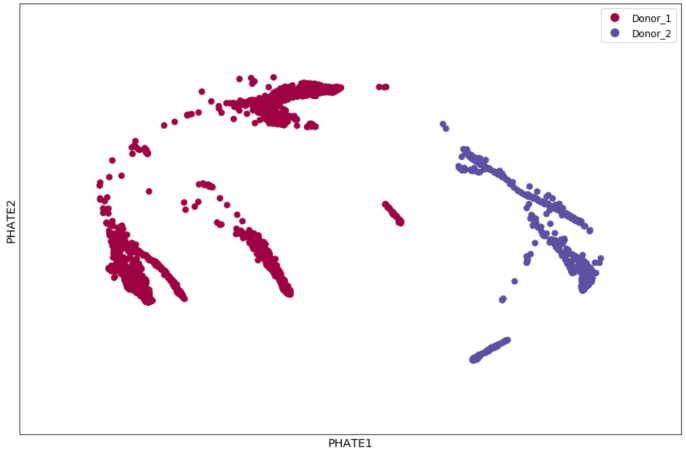


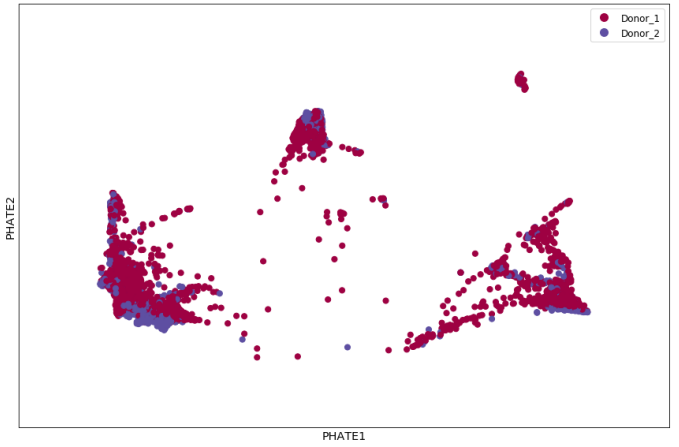


Uses softer affinity matrix and pulls the data in to correct the result


Uncorrected
(b)
MNN
limma
(d)
ComBat
tSNE 1
tSNE 2
Cell type 1
Cell type 2
Cell type 3
Batch 1
Batch 2

MNN with MAGIC Correction





Is mutual nearest neighbor (MNN) normalization linear or non-linear?

Linear

Non-linear

Summary of data denoising

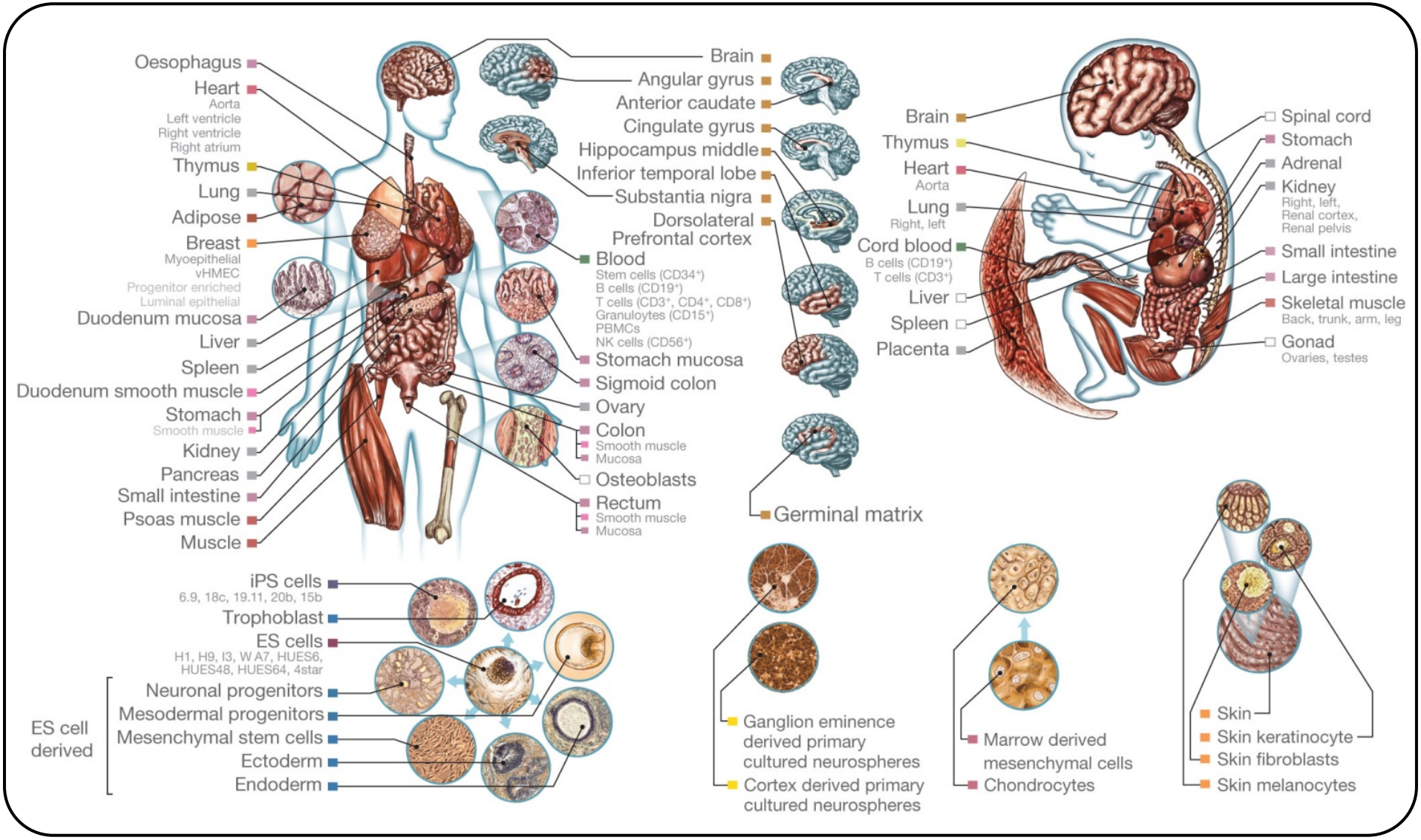
- Batch effects are sample-specific changes in measurements
- The goal of batch-normalization is to align cells of the same “type” across samples
- Mutual nearest neighbors (MNN) normalized batches by matching cells that are both close to each other

What questions do you have?

Please submit on Slack

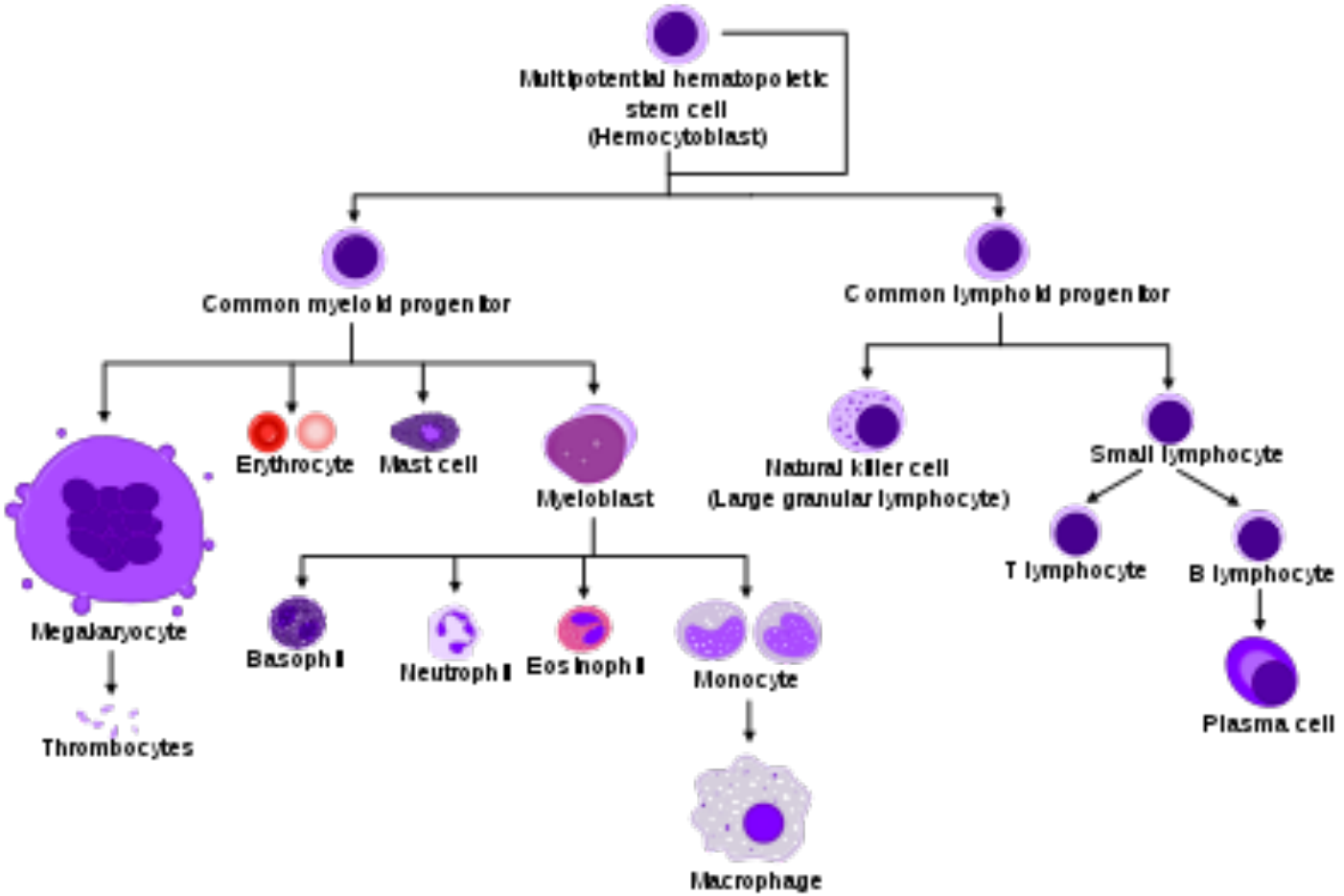
Clustering

High Degree of Complexity

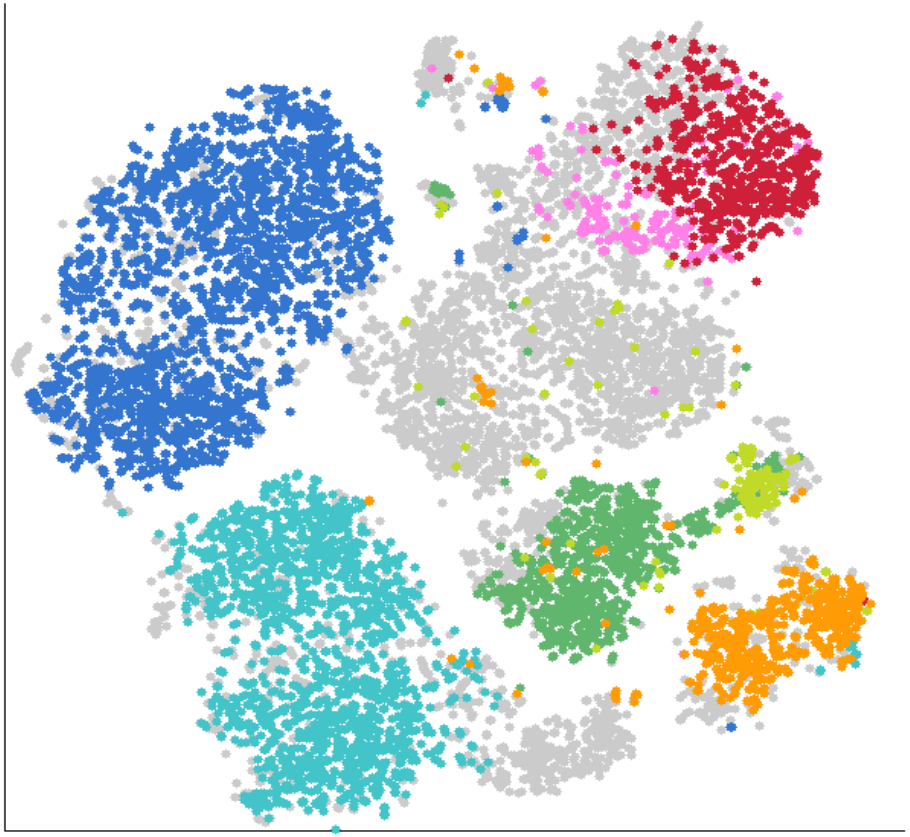


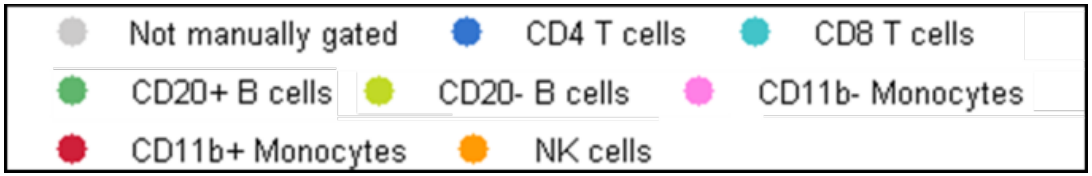
Roadmap Epigenomics Consortium Nature 2015

Phenotyping in Biology



tSNE Map of Immune Cells

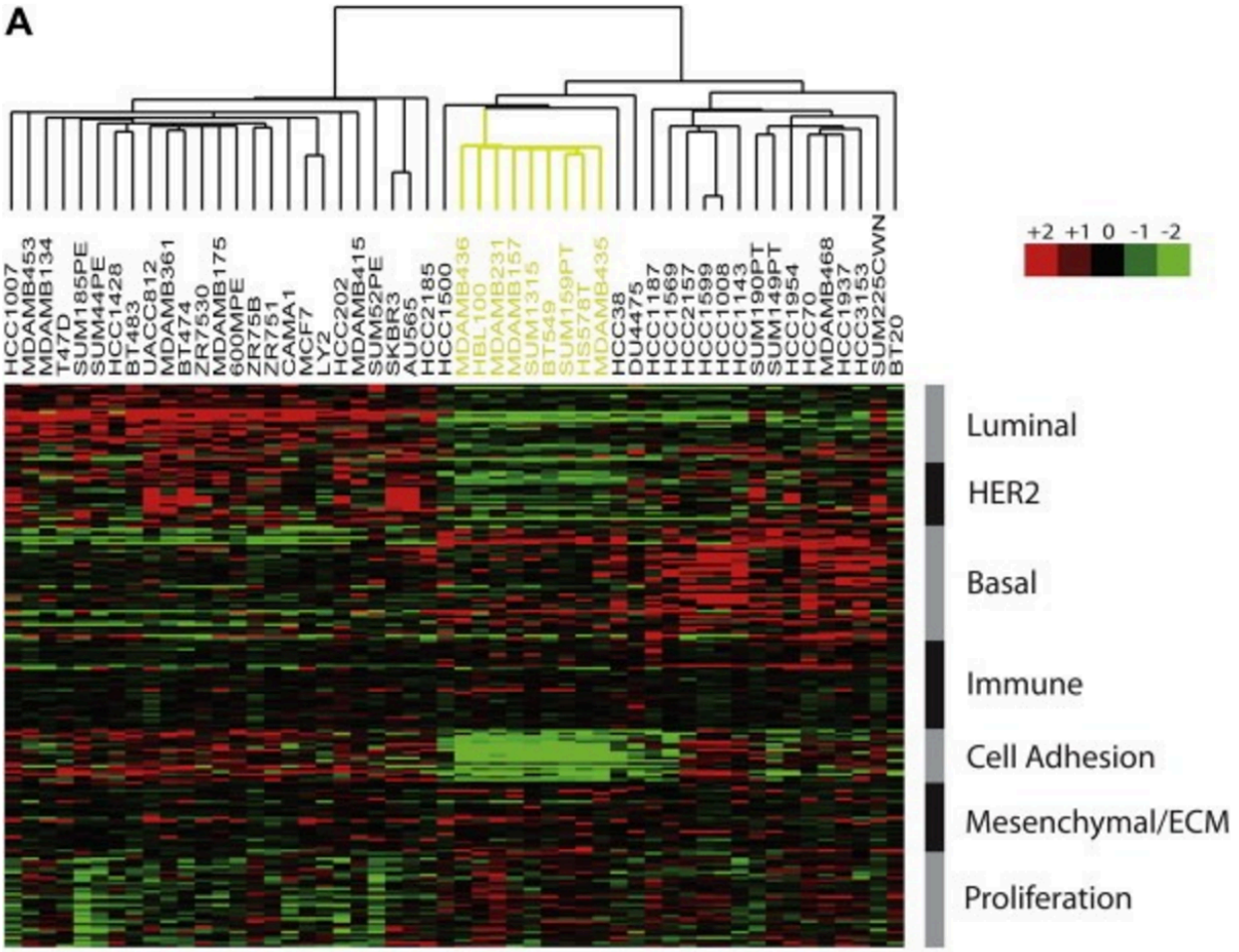




van der Maaten JMLR 2008, 2014

Amir *et al.* Nat. Biotech 2013

Breast Cancer Subtypes



Deconstructing the molecular portraits of breast cancer

Aleix Prat [1, 2, 3] and Charles M. Perou [1, 2, 3]

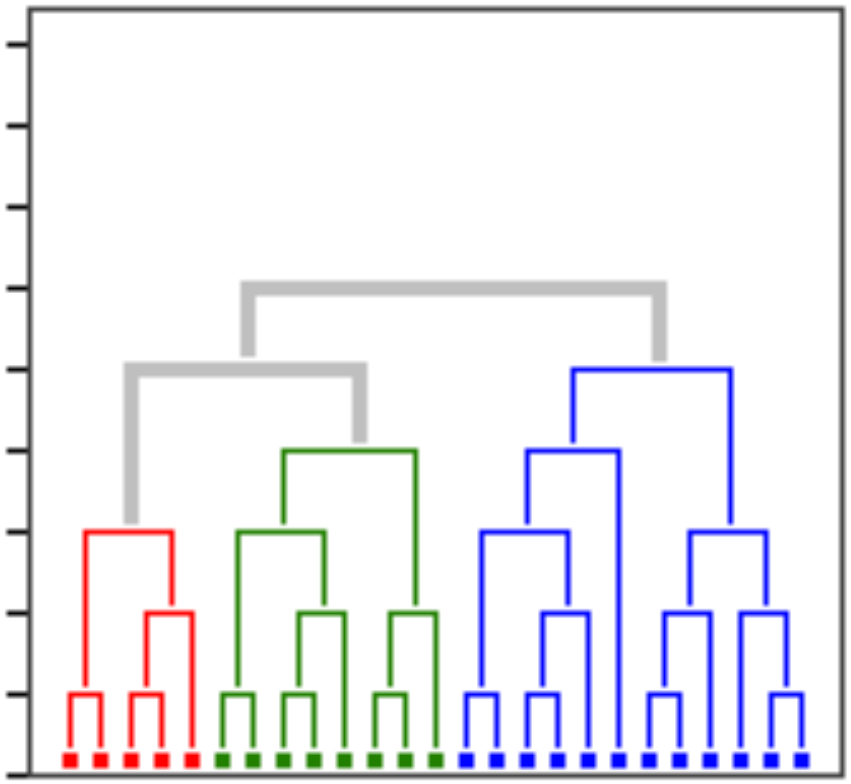
Author information ► Article notes ► Copyright and License information ►

Clustering Used in Many Ways in Biology

- Clustering gene expression profiles to find “modules” or groups of genes that work together
- Clustering patient data to see if patients have similar disease
- Clustering cells to find different cell types

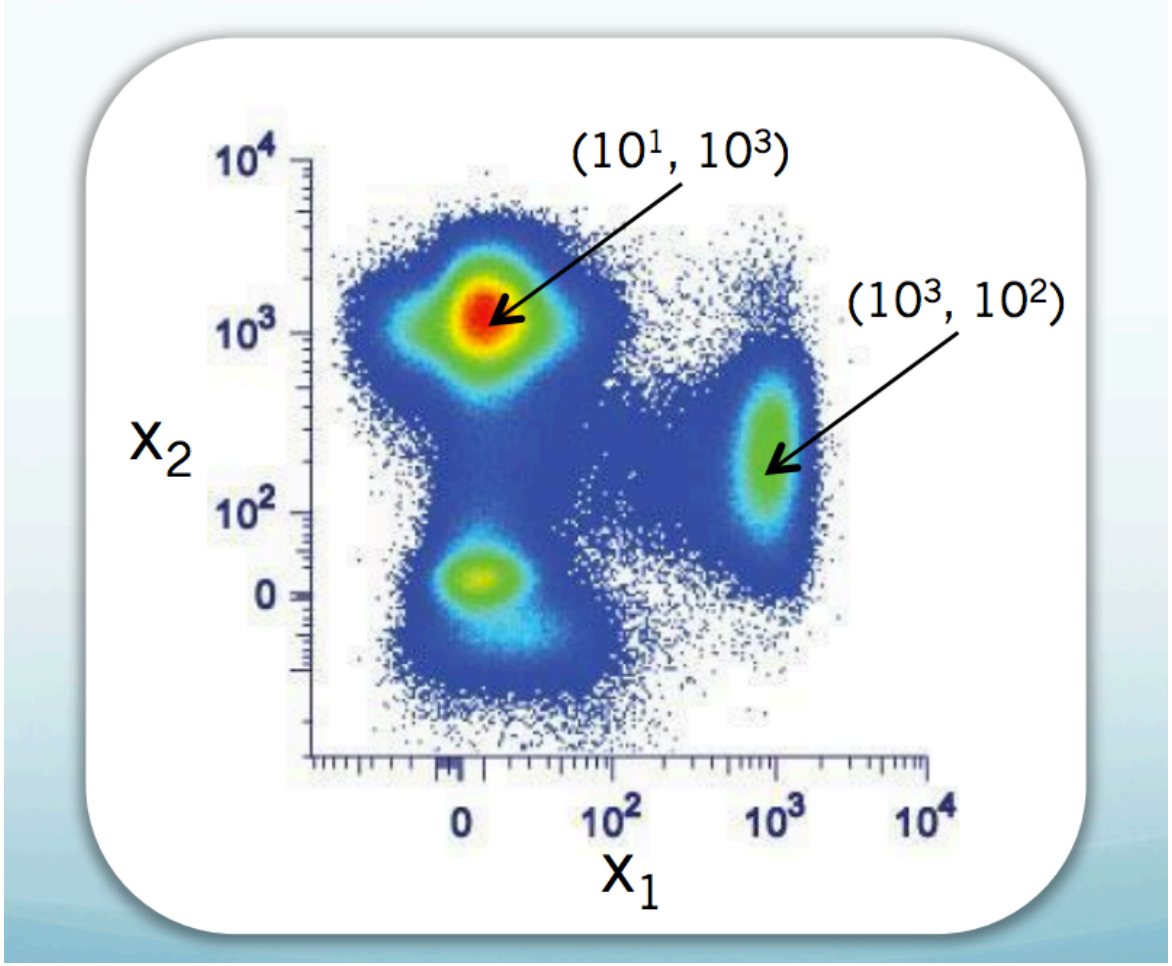
What is clustering?

Grouping of Similar Objects



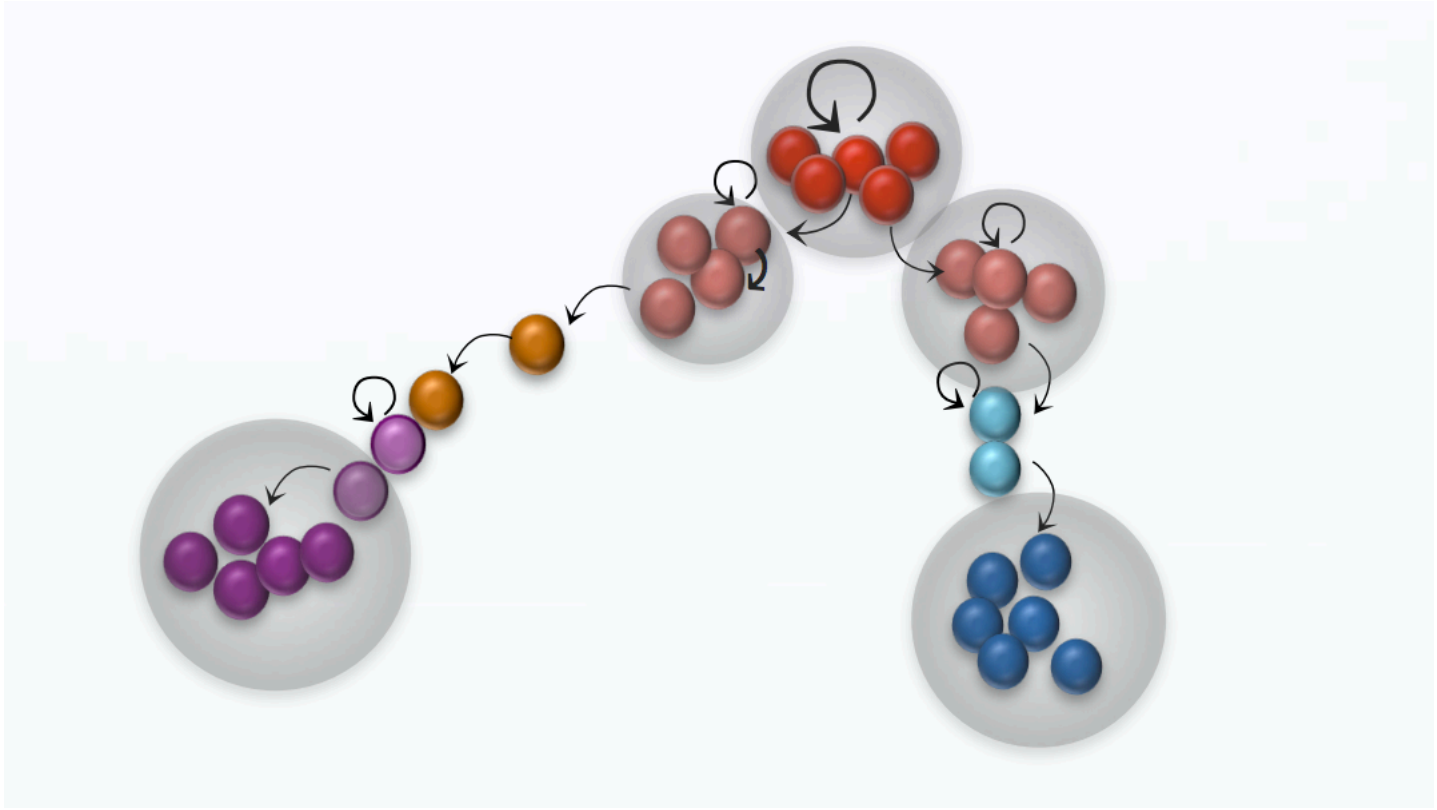
Linkage Clustering, Community detection

Density Centers



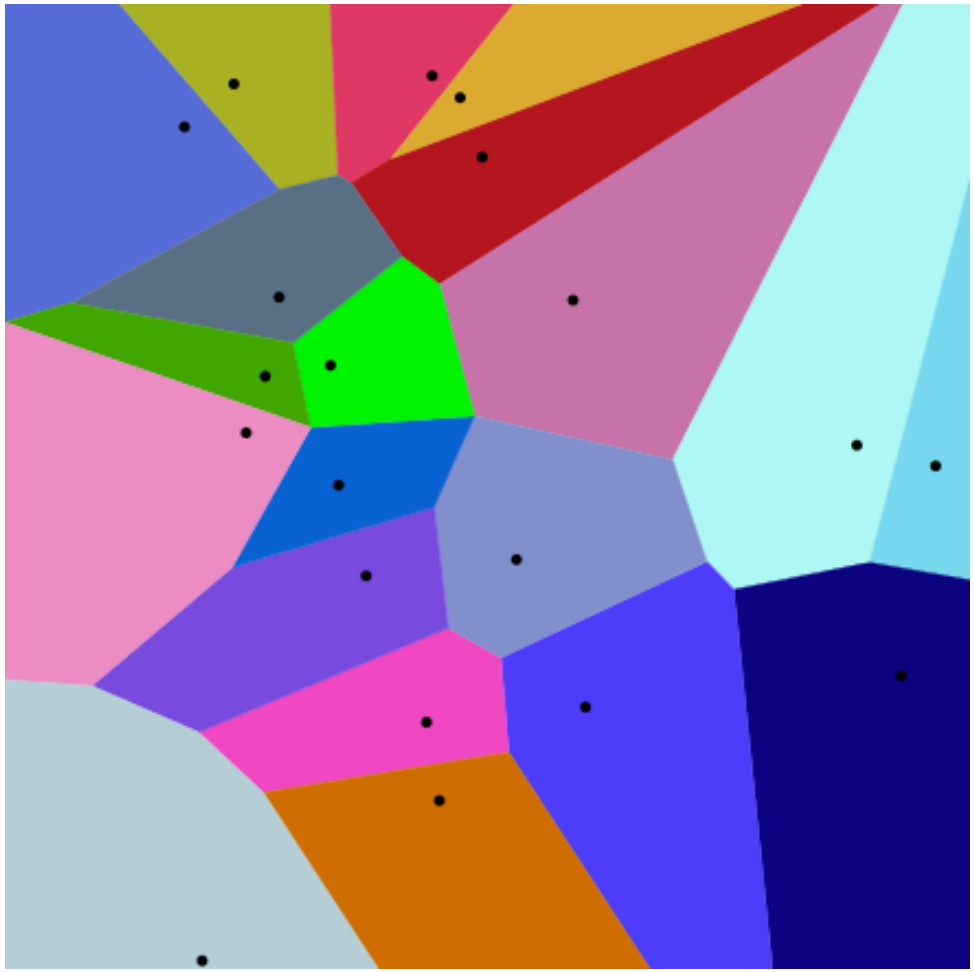
DBSCAN, Gaussian Mixture Model

Metastable States in A Manifold



PHENOGRAPH

A partitioning of the data space



Voronoi Diagram

K-MEANS (Macloed 1967), Fidduccia Matheyses

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 $X = \{x_1, x_2, \ldots x_n\}$
- Partition into k clusters

- Such that the within-cluster variance is minimized

$$\arg\min_{\mathbf{S}} \sum_{i=1}^{k} \sum_{\mathbf{x} \in S_i} \|\mathbf{x} - \boldsymbol{\mu}_i\|^2 = \arg\min_{\mathbf{S}} \sum_{i=1}^{k} |S_i| \operatorname{Var} S_i$$

K-Means Clustering

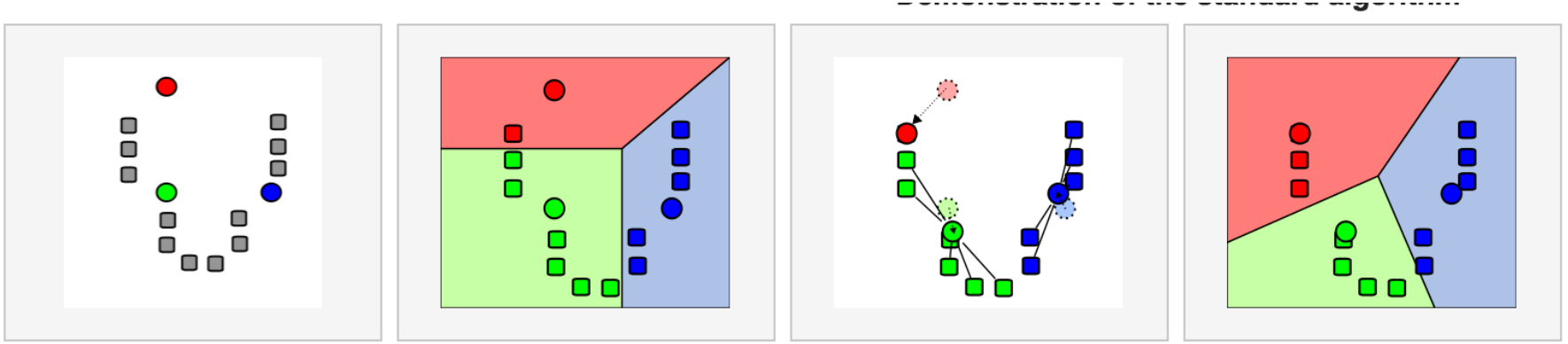
- Randomly partition data into k clusters
-
- **Update step**: Compute the means of clusters

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

- **Assignment step:** Reassign each x_i to cluster S_j that has the nearest mean u_j

Nearest mean minimizes squared Euclidean distance

Algorithm Iterations



Will this process stop?

Yes: K-means Converges

- Why?
- Each step LOWERS mean squared error
- Update of means lowers variance
- Reassignment of points to nearby means lowers variance

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

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-mediods helps with this
- Finds a local minima: repeat with different initializations

How do you pick K?

- If you keep increasing K the fit keeps getting better!
- K has to balance between overfitting, having too many parameters and modeling the data
- Akaike Information Criterion:

- Model selection

$$AIC = 2m - 2\ln(L)$$

$$L = P(x \mid \theta) = likelihood$$

Cluster Silhouette

- Measures the appropriateness of cluster assignments
- Within cluster dissimilarity
- Between cluster dissimilarity:

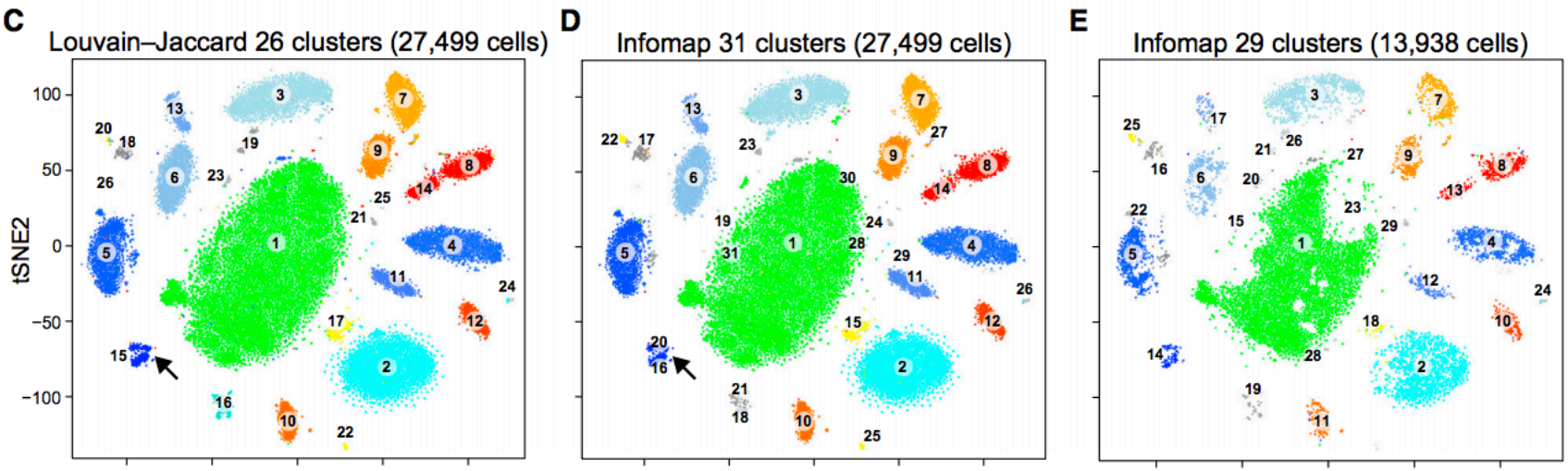
$$a(i) = \sum_{y \in S_i} \| x_i - y \|$$

- Silhouette score
- Can be used for picking k

$$b(i) = \min_{j \neq i} \sum_{y \in S_j} \| x_i - y \|$$

$$s(i) = \begin{cases} 1 - a(i)/b(i), & \text{if } a(i) < b(i) \\ 0, & \text{if } a(i) = b(i) \\ b(i)/a(i) - 1, & \text{if } a(i) > b(i) \end{cases}$$

Clustering in Single-Cell RNA-seq



Resource

Cell

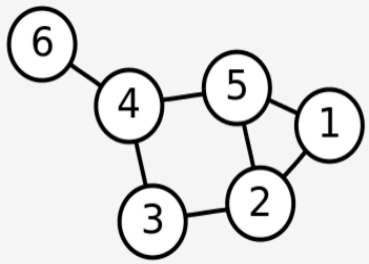
Comprehensive Classification of Retinal Bipolar Neurons by Single-Cell Transcriptomics

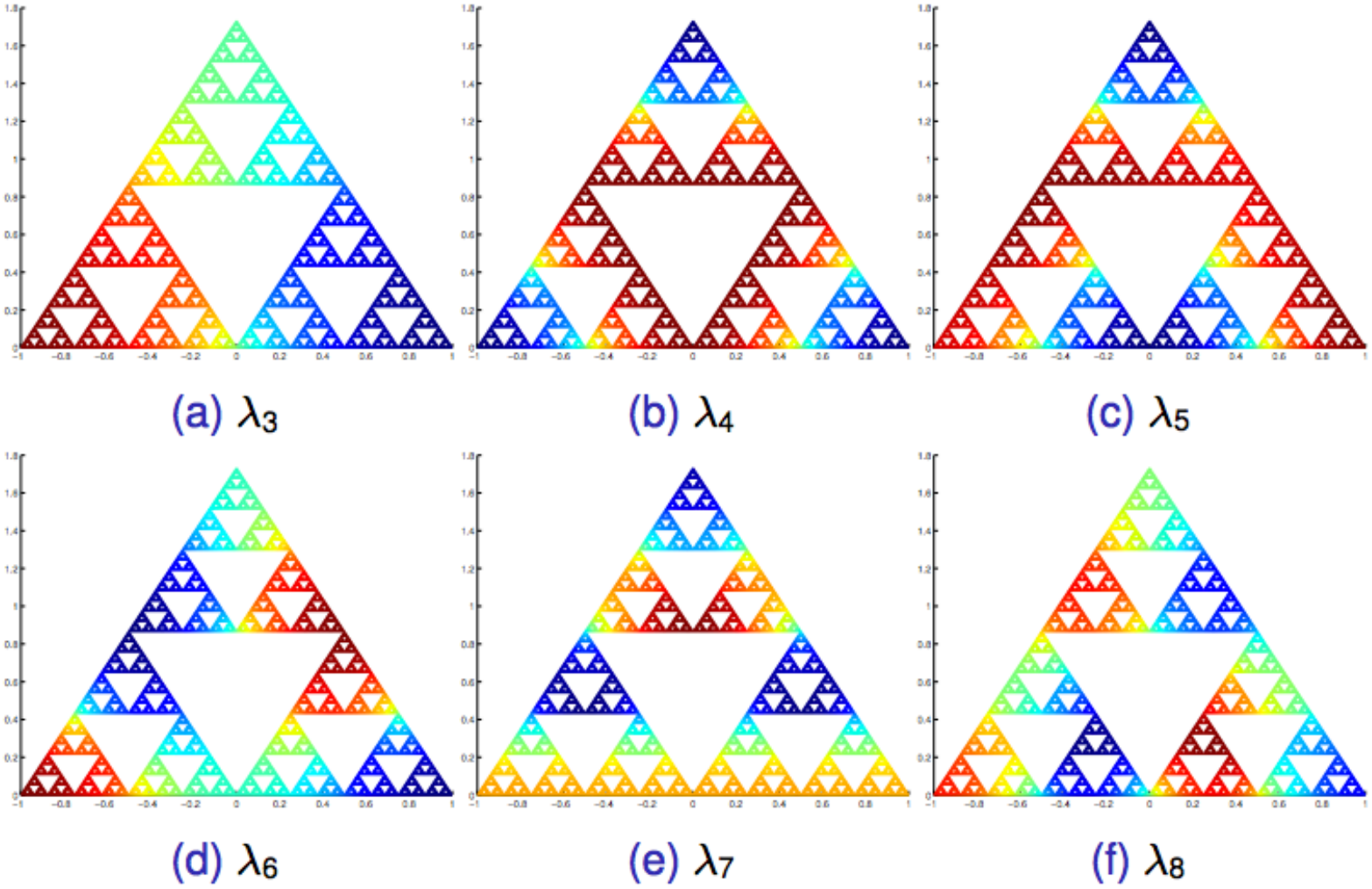
Graph Laplacian

- W = similarity or adjacency matrix
  - Adjacency matrix can be a binarized 0-1 matrix
- D = Degree Matrix

$$d_i = \sum_{j=1}^{n} w_{ij}.$$

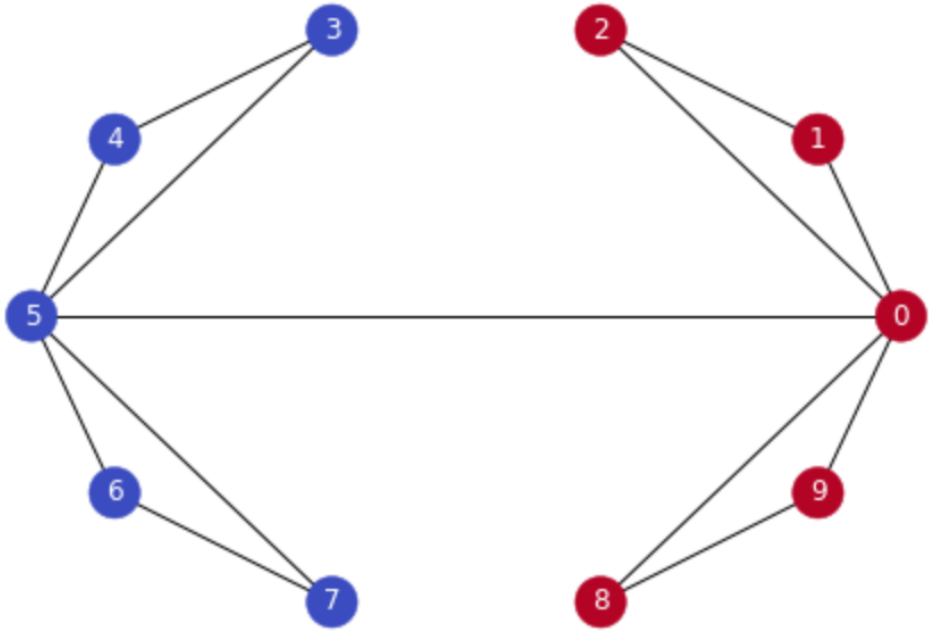
- Graph Laplacian: L = I

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



Same eigenvectors as affinity matrix

Fiedler Vector

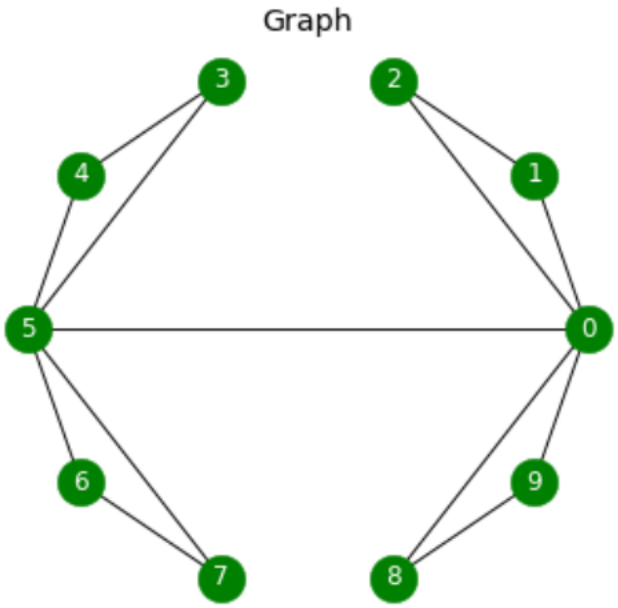


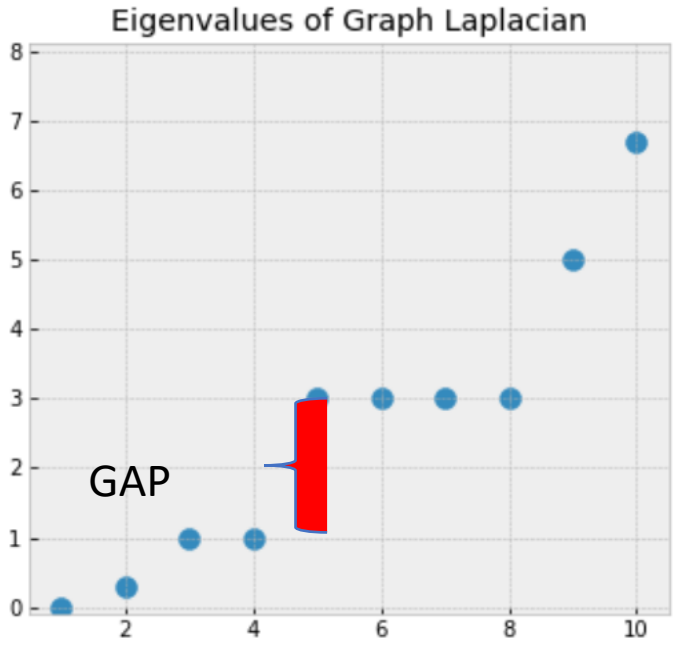
Second-smallest Eigenvalue is called Fiedler value, corresponding vector is the Fiedler vector

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.

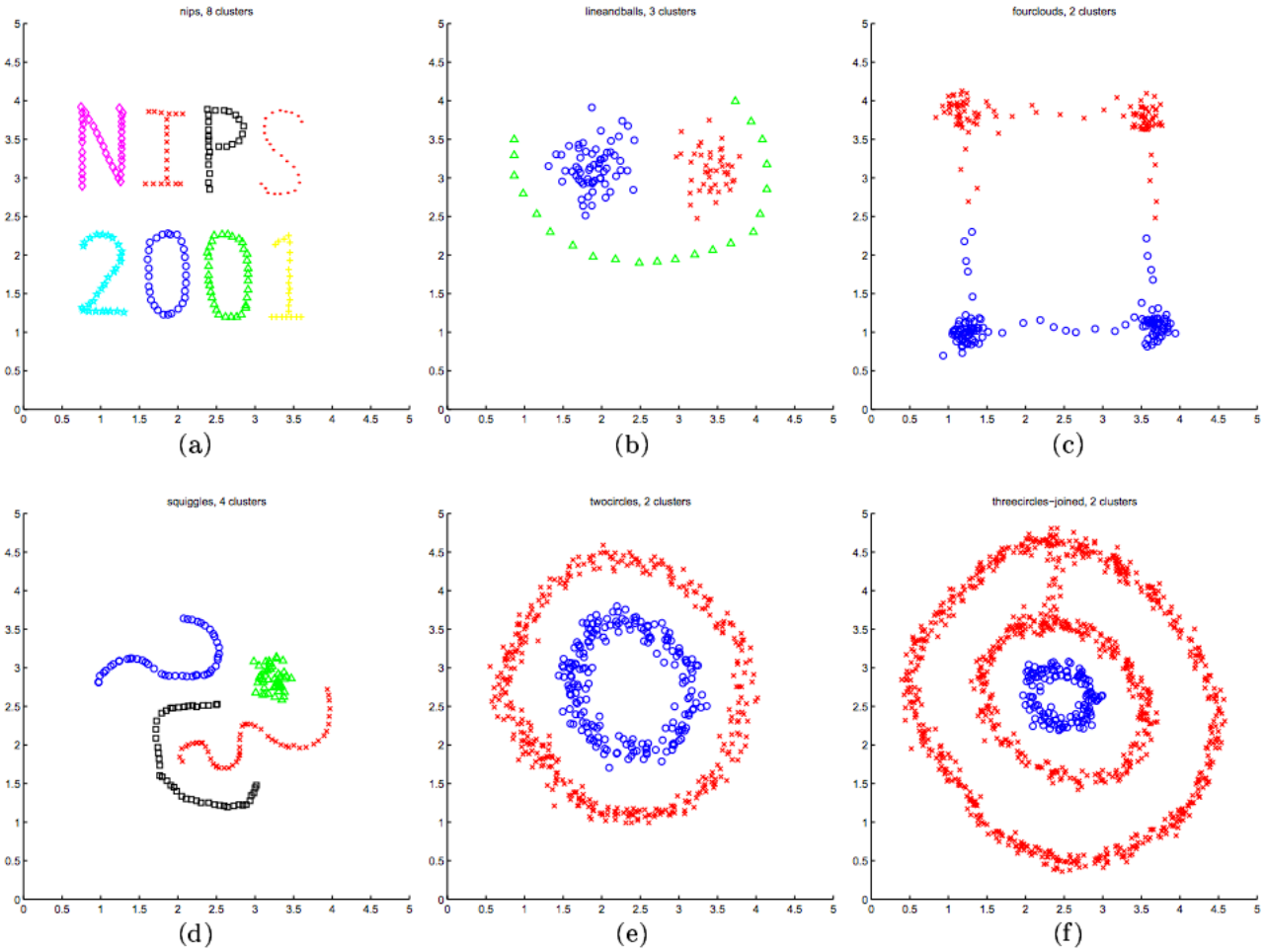
K-means Spectral Clustering

- Compute first K eigenvectors of L (of the smallest eigenvalue)
  - k-dimensional representation of datapoints
- Use K-means clustering on these representations
- Why?

Spectral Clustering Algorithm

1. Form the affinity matrix $A \in \mathbb{R}^{n \times n}$ defined by $A_{ij} = \exp(-||s_i - s_j||^2/2\sigma^2)$ if $i \neq j$, and $A_{ii} = 0$.
2. Define D to be the diagonal matrix whose (i,i)-element is the sum of A's i-th row, and construct the matrix $L = D^{-1/2}AD^{-1/2}$.[1]
3. Find $x_1, x_2, \ldots, x_k$, the k largest eigenvectors of L (chosen to be orthogonal to each other in the case of repeated eigenvalues), and form the matrix $X = [x_1 x_2 \ldots x_k] \in \mathbb{R}^{n \times k}$ by stacking the eigenvectors in columns.
4. Form the matrix Y from X by renormalizing each of X's rows to have unit length (i.e. $Y_{ij} = X_{ij}/(\sum_j X_{ij}^2)^{1/2}$).
5. Treating each row of Y as a point in $\mathbb{R}^k$, cluster them into k clusters via K-means or any other algorithm (that attempts to minimize distortion).
6. Finally, assign the original point s_i to cluster j if and only if row i of the matrix Y was assigned to cluster j.

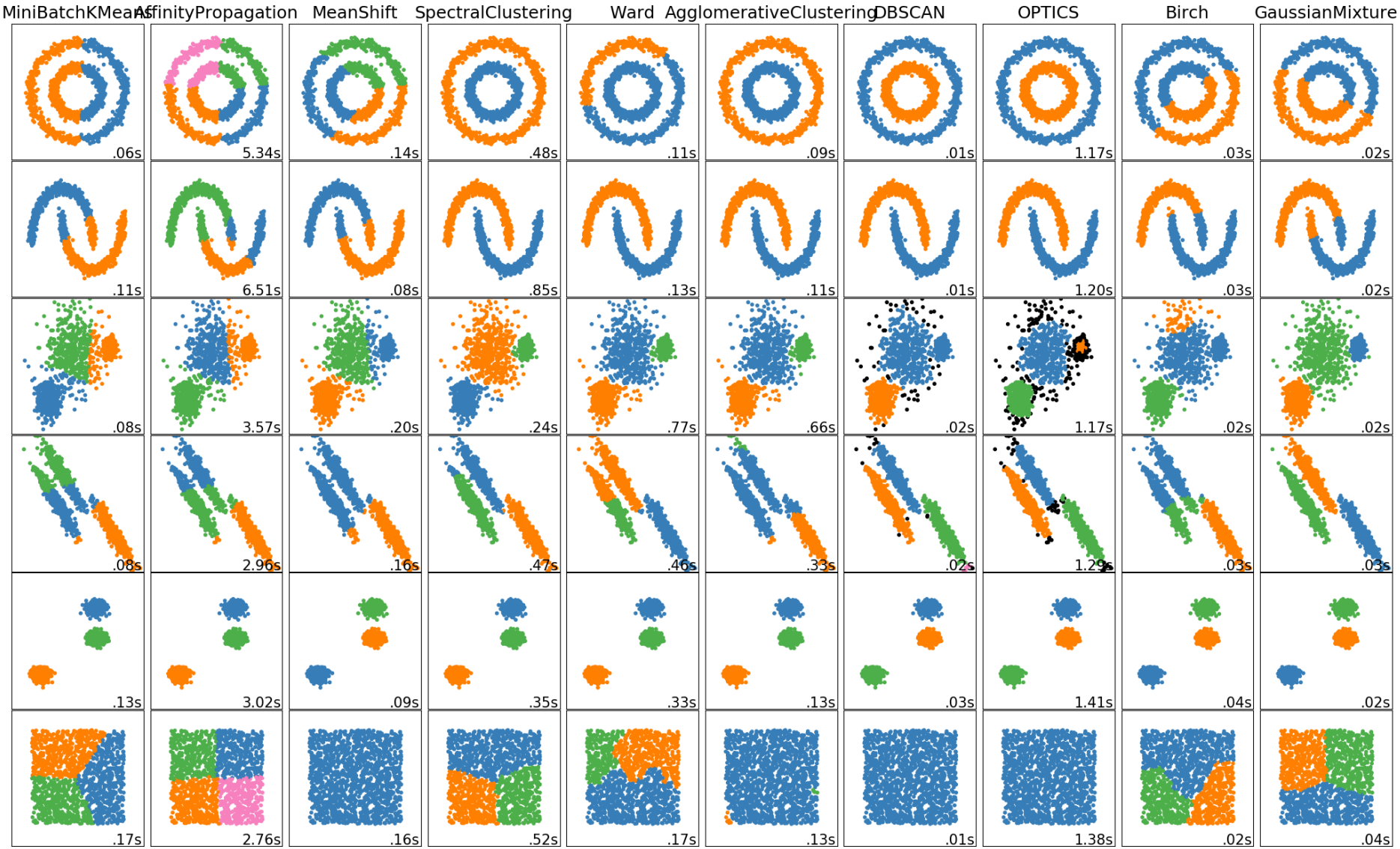
Illustration of Algorithm



What are advantages of this?

Advantages

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



https://scikit-learn.org/stable/modules/clustering.html

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 minimized within-cluster distances using eigenvectors of the laplacian