Comp683: Computational Biology

Lecture 9

Feb 7, 2024

Today + Announcements

- Single-cell clean-up
  - Dropout Imputation with MAGIC
- Branch preserving visualization for single-cell data

Announcement : No class on Tuesday due to wellness day.

Review Questions

1. What kind of clustering does the spade algorithm use and what are its limitations?
2. What is a common data artifact that we have with single-cell RNA-seq?

Step 2: Transform Marker Expressions

In the cell $\times$ marker matrix, we are counting the number of detected ions or joins between protein and heavy-metal tagged antibody. We will use a transformation, to compress the upper end of the spectrum and enhance the lower end.

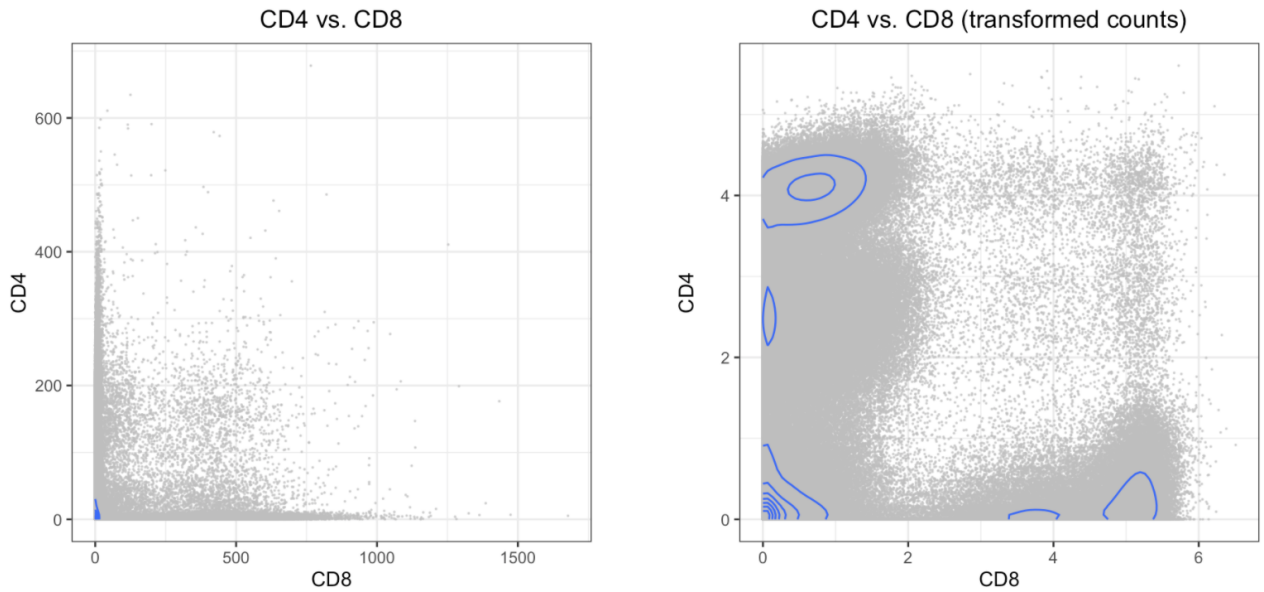


Figure: from `https://biosurf.org/cytof_data_scientist.html`

Imputation for Single-Cell Data

- Especially in scRNAseq, the datasets can be quite sparse, meaning that lowly-expressed genes fail to be detected.
- We can imagine an analog in mass cytometry, where signal can be a bit noisy (sometimes, we call it super staining when an antibody is too sticky)

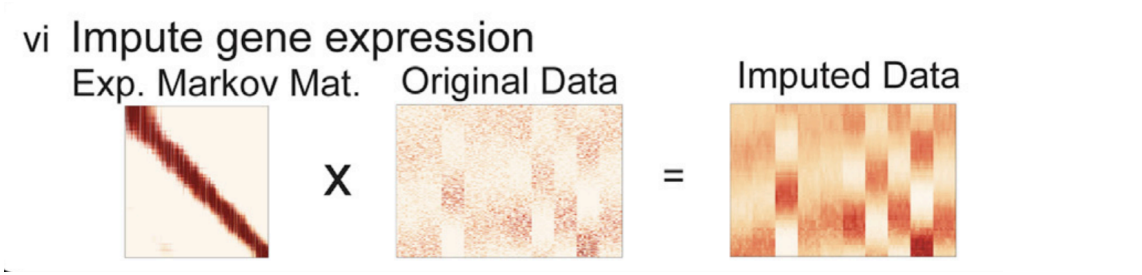


Figure: from van Dijk *et al.* Cell Systems. 2018

Accurately Representing Gene-Gene or Protein-Protein Interactions

The effect of lowly-expressed genes or proteins that are not adequately detected.

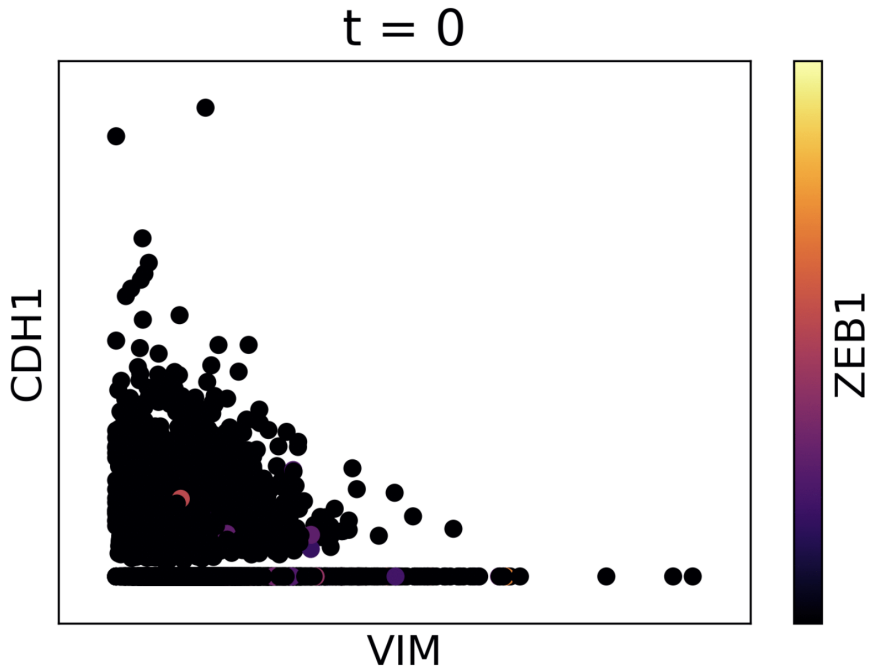
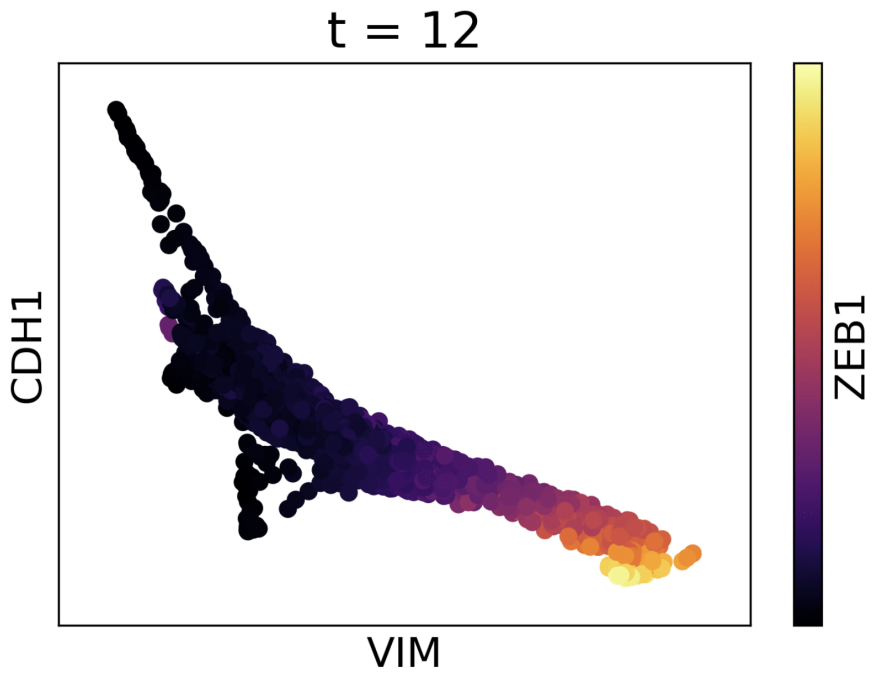


Figure: Note all of the 0 expression for CDH1

Estimating the Dropout, or Artificial Sparsity

If we can consider features of cells (in this case), that are otherwise similar to where dropout seems to have occurred, we can correct for the dropout and restore gene/gene or protein/protein interactions.



MAGIC Imputation for Single Cells

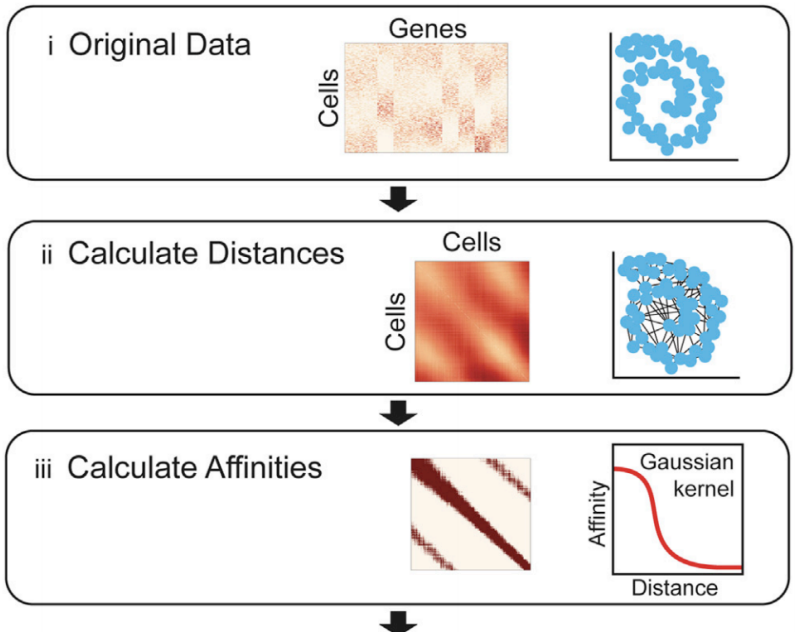
Here is an outline for the main idea.

- For a particular cell, find other cells that are most similar
- Construct a weighted affinity matrix between cells to more accurately capture between-cell similarities within a neighborhood
- Use affinities, and cell-cell neighborhood structure to correct the data

Step 1: Compute a Markov Affinity Matrix

Given the cell $\times$ feature matrix, $\mathbf{X}$, compute the cell-to-cell affinity matrix, $\mathbf{A}$ as follows. In particular, we compute affinities between each pair of cells (e.g. cell i and cell j).

$$A_{ij} = \exp(-(\text{Dist}(i,j)/\sigma)^2) \tag{1}$$



Notes on the σ parameter

In $A_{ij} = \exp(-(\text{Dist}(i,j)/\sigma)^2)$

- If σ is too small, the graph can become too disconnected.
- If σ is too large, cell-type resolution will be lost in the data and cell-types will be merged together.
- The solution is to adapt σ for each cell (σ_i) to take into account the density of its neighborhood. This prevents denser cell-types from dominating the imputation.
- In particular, σ_i is the distance between i and its kth nearest neighbor (default $k = 5$) [1].

[1] `https://github.com/KrishnaswamyLab/MAGIC/blob/master/python/magic/magic.py`

Question for you

Consider two cells, i and j, with corresponding feature vectors, $\mathbf{x}_i$ and $\mathbf{x}_j$. What problems do you see if the dataset dimension is large (let's say 20K features)? Also, what could be done to deal with this?

Affinity Matrix Cleaning

- Due to the adaptive σ_i, the affinity matrix is not symmetric!
- This can quickly be made symmetric with,

$$\mathbf{A} = \mathbf{A} + \mathbf{A}^T \tag{2}$$

- The next step is in row-stochastic normalization that renders the affinity matrix into a Markov transition matrix, $\mathbf{M}$

$$M_{ij} = \frac{A_{ij}}{\sum_k A_{ik}} \tag{3}$$

Markov Affinity Based Graph Diffusion : Exponentiating **M**

- By powering the matrix ($\mathbf{M}^t$), you can refine between-cell similarities related to the neighborhood structure. In other words, increase weight between pairs that have many common neighbors.
- M_{ij}^t represents the probability that a random walk of length t will reach node j after starting from node i.
- As t increases, false similarity between cells that was based strongly on noise gets filtered out. So, spurious neighbors will be down-weighted.

Try it Out

```
>>> A = np.matrix([[1/3,1/3,1/3],[0,0,1],[2/3,1/3,0]])
>>> A
matrix([[0.33333333, 0.33333333, 0.33333333],
        [0.        , 0.        , 1.        ],
        [0.66666667, 0.33333333, 0.        ]])
>>> matrix_power(A,2)
matrix([[0.33333333, 0.22222222, 0.44444444],
        [0.66666667, 0.33333333, 0.        ],
        [0.22222222, 0.22222222, 0.55555556]])
>>> matrix_power(A,3)
matrix([[0.40740741, 0.25925926, 0.33333333],
        [0.22222222, 0.22222222, 0.55555556],
        [0.44444444, 0.25925926, 0.2962963 ]])
>>> matrix_power(A,4)
matrix([[0.35802469, 0.24691358, 0.39506173],
        [0.44444444, 0.25925926, 0.2962963 ],
        [0.34567901, 0.24691358, 0.40740741]])
>>> matrix_power(A,5)
matrix([[0.38271605, 0.25102881, 0.36625514],
        [0.34567901, 0.24691358, 0.40740741],
        [0.38683128, 0.25102881, 0.36213992]])
```

Visualizing what is happening....

As t increases, dense areas of the data result in more possible paths and weights are then concentrated in this area (e.g. closest neighbors remain with high probability). Also, helps to filter out noise.

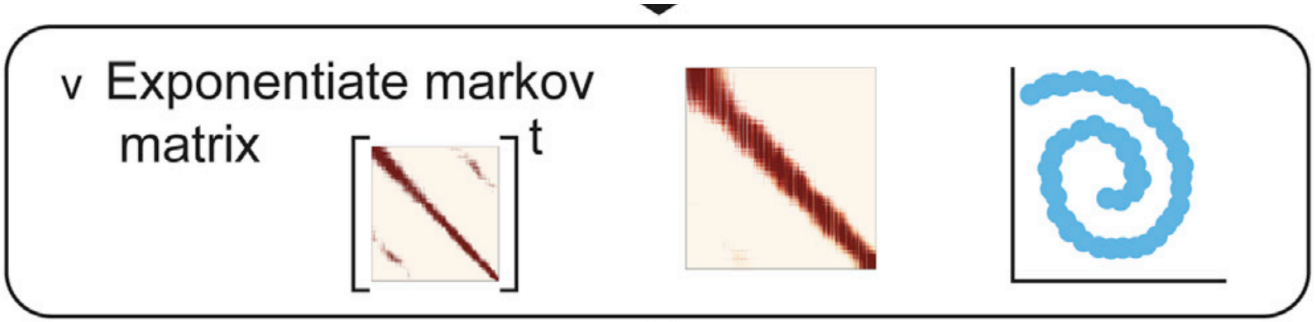


Figure: from van Dijk *et al.* Cell Systems. 2018

Imputation After Graph Diffusion

Finally, the original data $\mathbf{X}$ (of cells $\times$ features) is transformed as,

$$\mathbf{X}_{\text{impute}} = \mathbf{M}^t\mathbf{X} \tag{4}$$

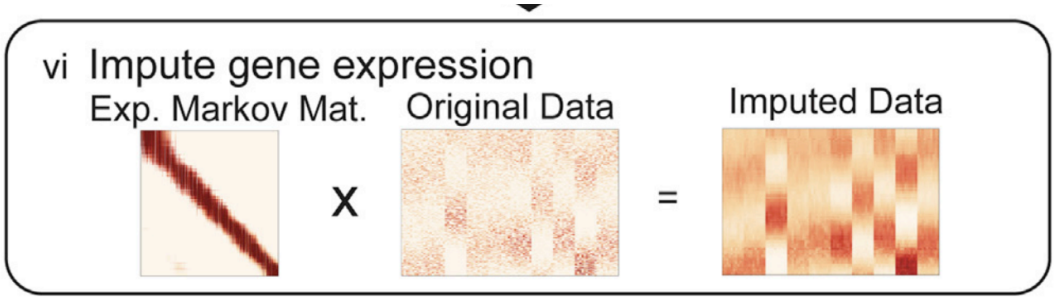


Figure: from van Dijk *et al.* Cell Systems. 2018

On a Practical Note, t

The t, or how you power the matrix is important to the results.

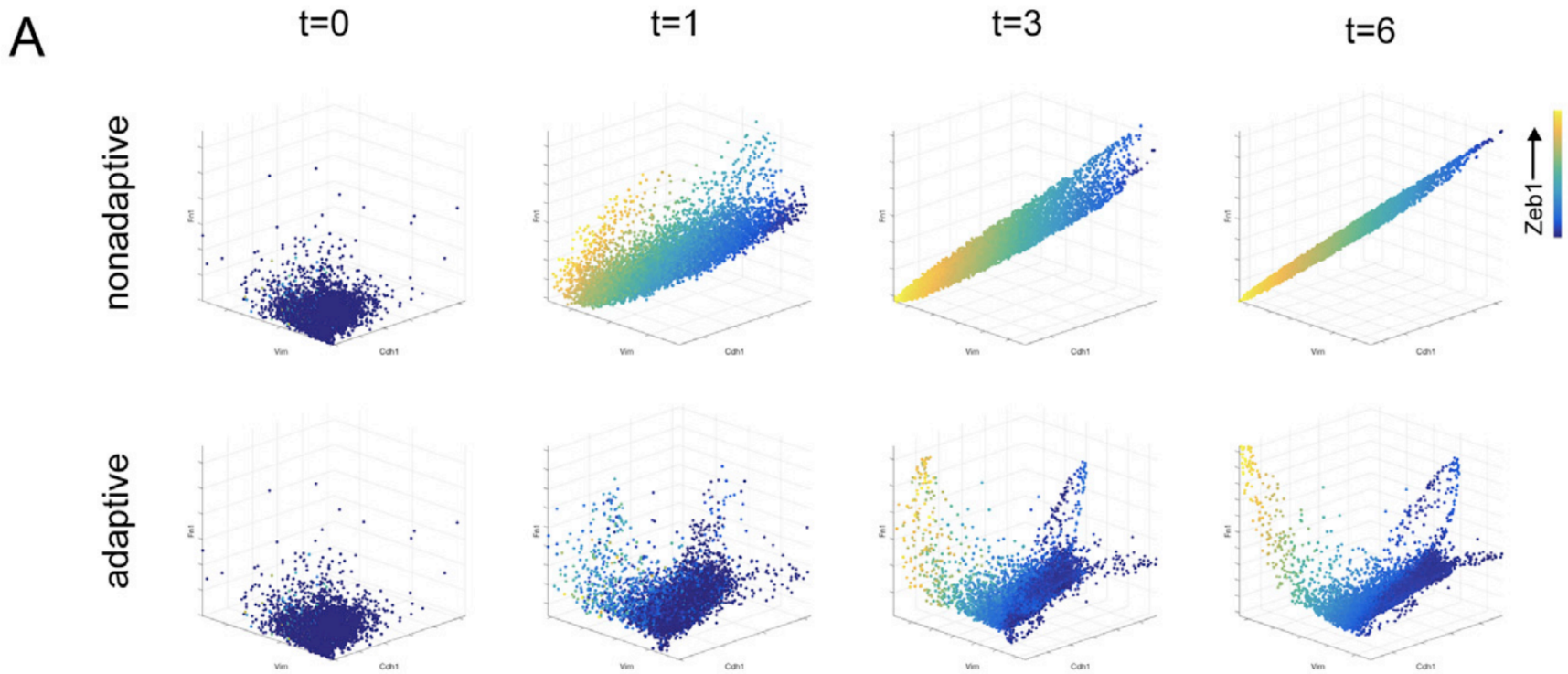


Figure: from van Dijk *et al.* Cell Systems. 2018. Adaptive means that you chose a σ_i for a particular cell, i.

Convergence of $\mathbf{X}_{\text{impute}}$ for different t

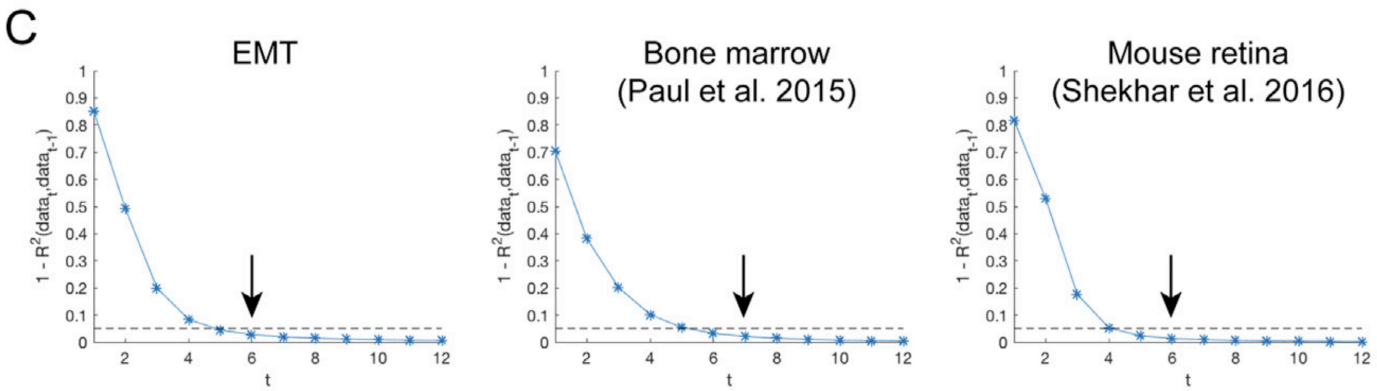


Figure: from van Dijk *et al.* Convergence of $\mathbf{X}_{\text{impute}}$ between t and $t-1$.

Effectiveness in Noise Reduction

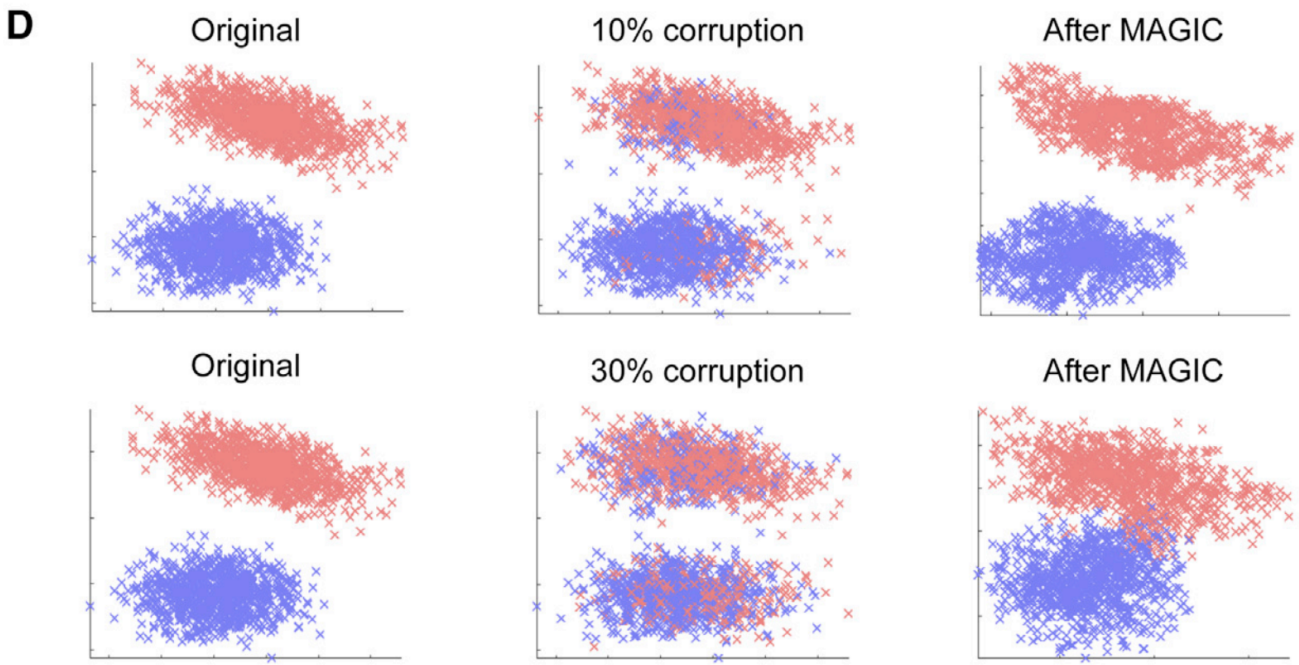


Figure: from van Dijk *et al.* MAGIC can restore noisy data, when features are artificially corrupted in a synthetic dataset.

Our Good Friend Flow Cytometry

Using MAGIC makes it easier to identify well-characterized cell-types.

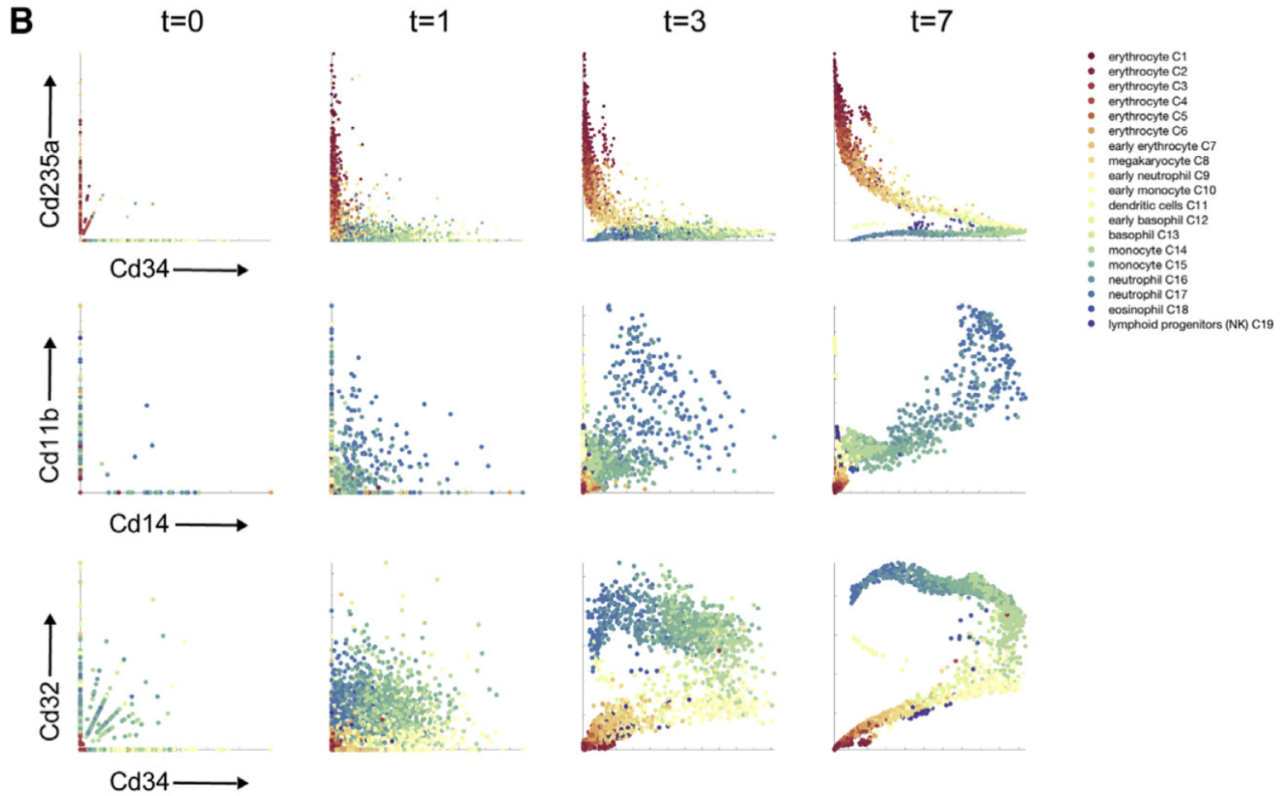


Figure: from van Dijk *et al.* Cell Systems. 2018.

Visualizing Single-Cell Data with PHATE

A Dimensionality Reduction Method for Single Cells

PHATE aims to preserve between-cell population similarities according to the way that cells differentiate.

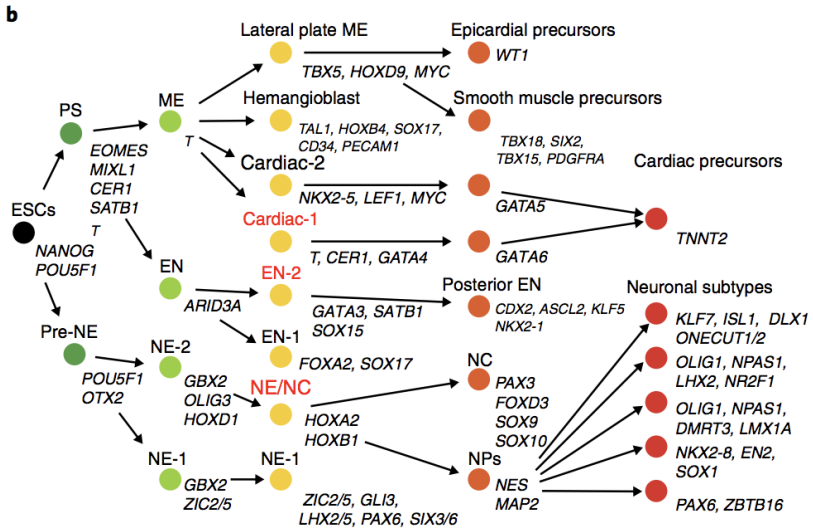


Figure: from Moon *et al.* Nature Biotechnology. 2019

Latent Structure

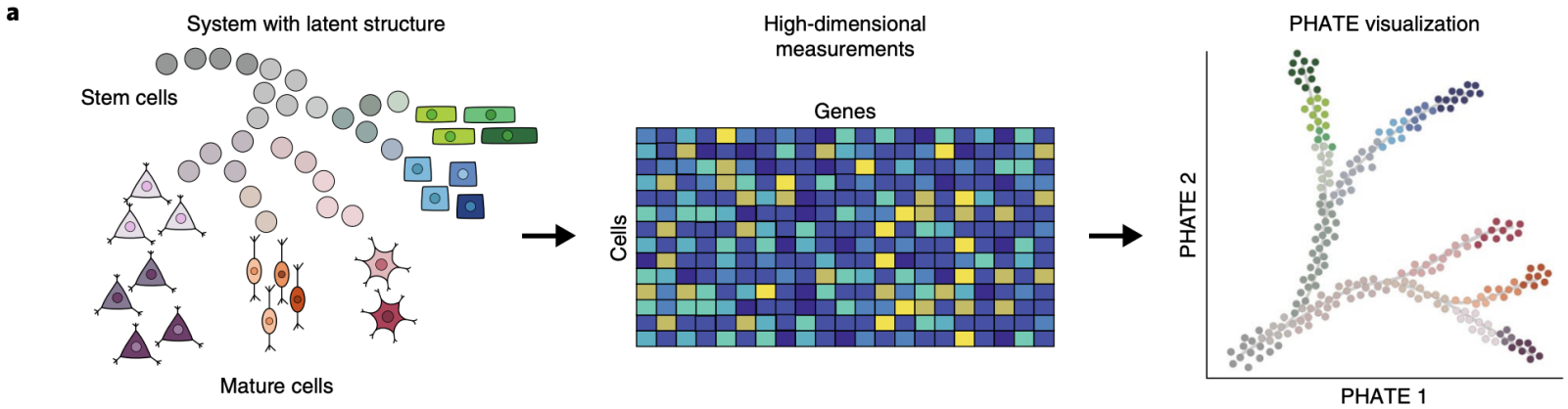


Figure: from Moon *et al.* Nature Biotechnology. 2019

PHATE Overview

Table 1 | General steps in the PHATE algorithm
Input: Data matrix, algorithm parameters (Methods)
Output: The PHATE visualization
(1) Compute the pairwise distances from the data matrix.
(2) Transform the distances to affinities to encode local information.
(3) Learn global relationships via the diffusion process.
(4) Encode the learned relationships using the potential distance.
(5) Embed the potential distance information into low dimensions for visualization.

Figure: from Moon *et al.* Nature Biotechnology. 2019

The First Few Steps are Identical to MAGIC

- **Step 1:** Cell $\times$ Cell distance matrix (based on Euclidean distance between cells)
- **Step 2:** Convert to Cell $\times$ Cell row-stochastic affinity matrix $\rightarrow$, $\mathbf{P}$, such that $\sum_j \mathbf{P}_{ij} = 1$.
- **Step 3:** Power $\mathbf{P} \rightarrow \mathbf{P}^t$.

A Complementary Method to Determine Optimal t

- The choice of t will determine how much noise you want to filter. Small eigenvalues of $\mathbf{P}$ correspond to noise, and will ultimately decrease towards 0.
- Let $[\eta(t)]_i = \lambda_i^t / \sum_{j=0}^{N-1} \lambda_j^t$ be the probability distribution defined by the non-negative eigenvalues of $\mathbf{P}^t$.

The von Neumann entropy $H(t)$ is then computed based on $[\eta(t)]_i$ and decreases as $t \to \infty$.

$$H(t) = -\sum_{i=1}^{N} [\eta(t)]_i \log([\eta(t)]_i) \tag{5}$$

Relationship Between t and $H(t)$

Higher t means less noise, but the implications of t are super powerful!

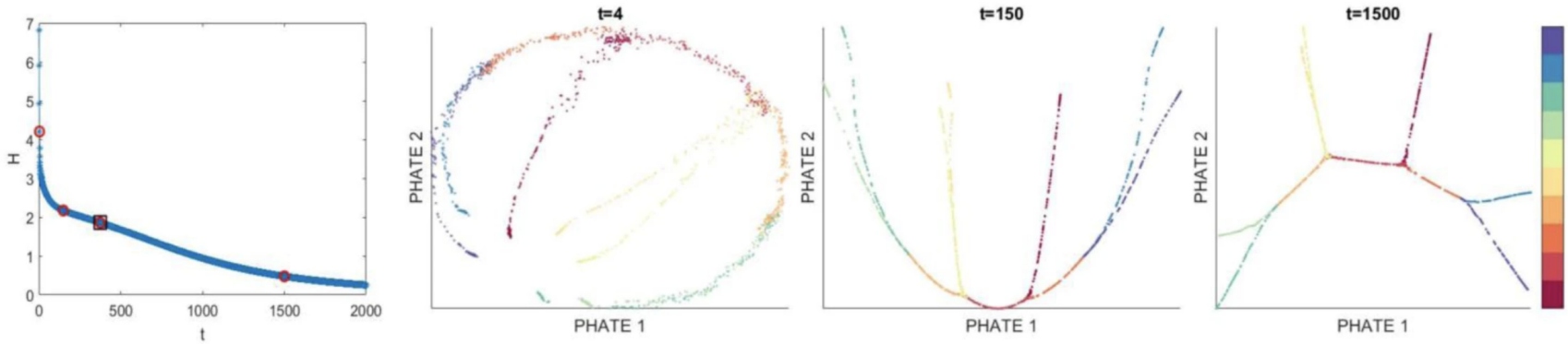


Figure: from Moon *et al.* Nature Biotechnology. 2019

Computing Potential Distances

- Potential distance is computed as a divergence between the associated diffusion probability distributions of the two cells to all other cells

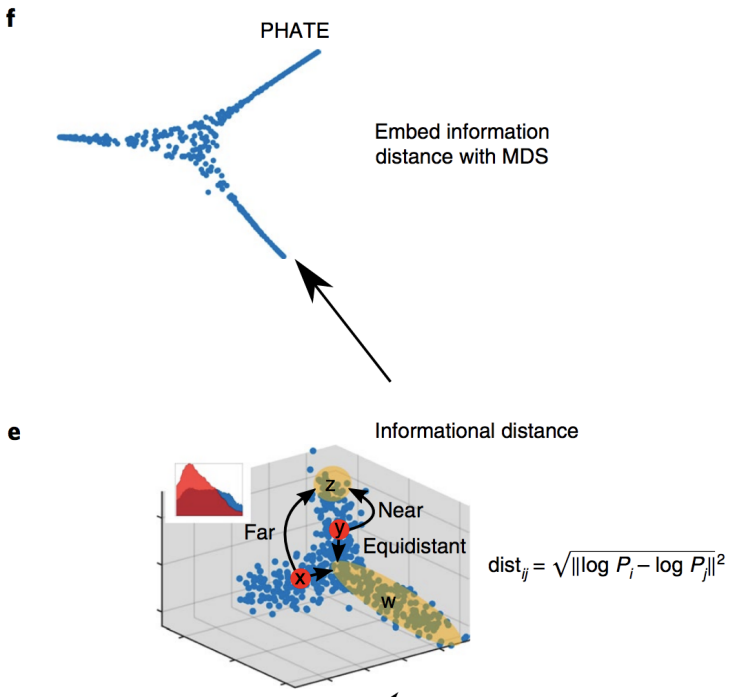


Figure: from Moon *et al.* Nature Biotechnology. 2019

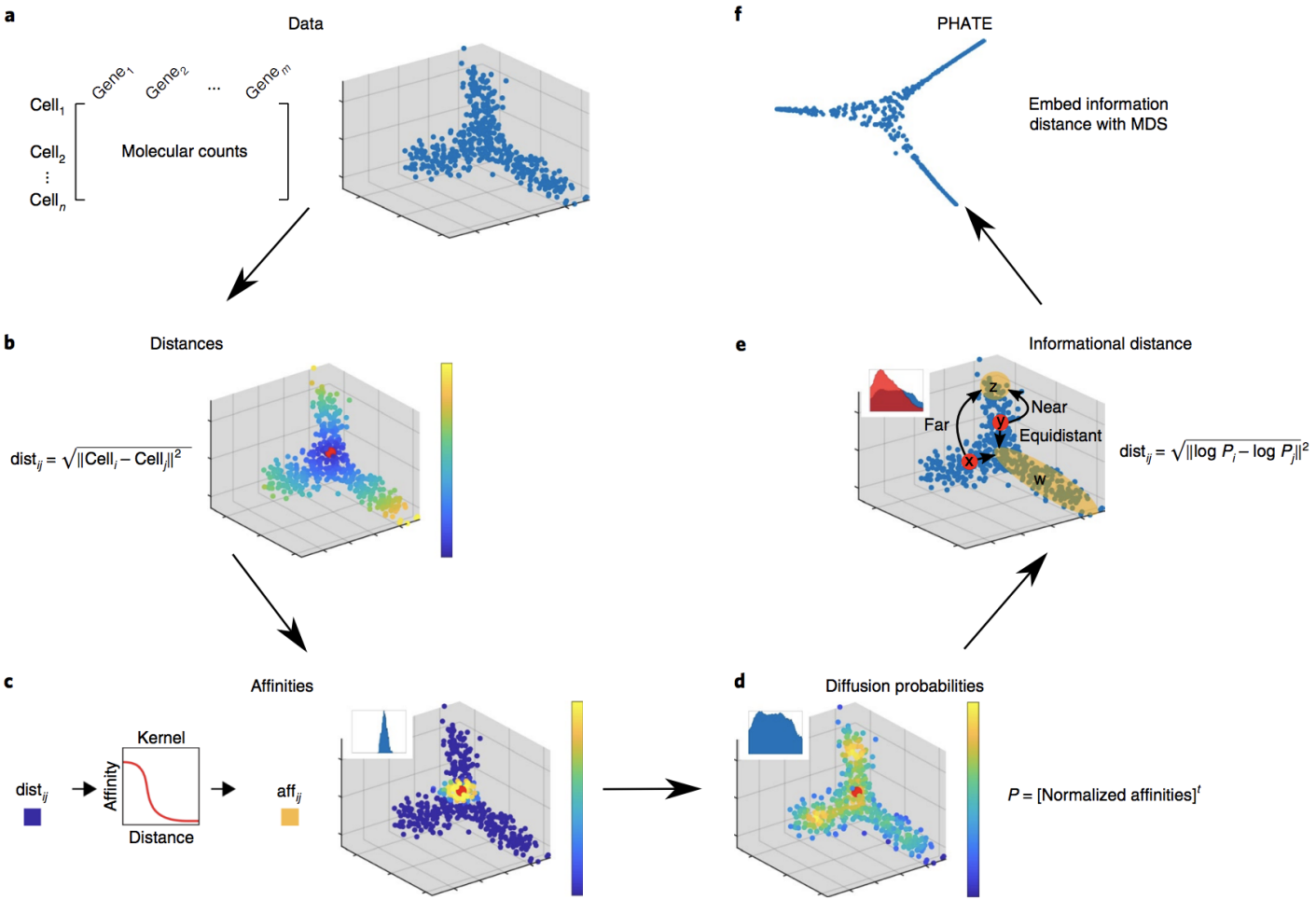
Potential Distances, Written Formally

- $U_{\mathbf{x}}^t = -\log(P_{\mathbf{x}}^t)$
- Then $\text{PD}(\mathbf{x}, \mathbf{y}) = ||U_{\mathbf{x}}^t - U_{\mathbf{y}}^t||_2$

Then use a classic algorithm called multi-dimensional scaling to project all points in two dimensions. Apply to,

$$\mathfrak{V}^t(\mathbf{x}, \mathbf{y}) = \left\| U_{\mathbf{x}}^t - U_{\mathbf{y}}^t \right\|_2, \mathbf{x}, \mathbf{y} \in \mathcal{X}$$

Recap


a
Data
Gene1
Gene2
...
Genem
Cell1
Cell2
Cellm
Molecular counts
b
Distances
distij = √||Celli − Cellj||2
c
Affinities
Kernel
distij
Affinity
Distance
affij
d
Diffusion probabilities
P = [Normalized affinities]t
e
Informational distance
z
Near
Far
y
Equidistant
x
w
distij = √||log Pi − log Pj||2
f
PHATE
Embed information
distance with MDS

PHATE preserves branch structure

As expected tSNE is preserving local similarities, but not necessarily distances between groups of points.

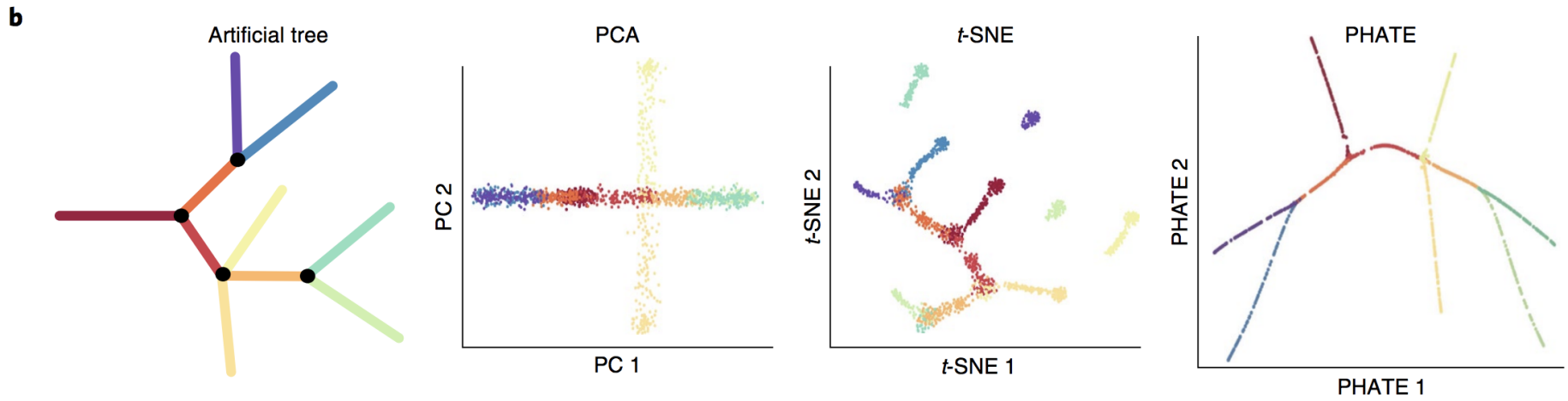


Figure: from Moon *et al.* Nature Biotechnology. 2019

And Preserves Cellular Differentiation Patterns!

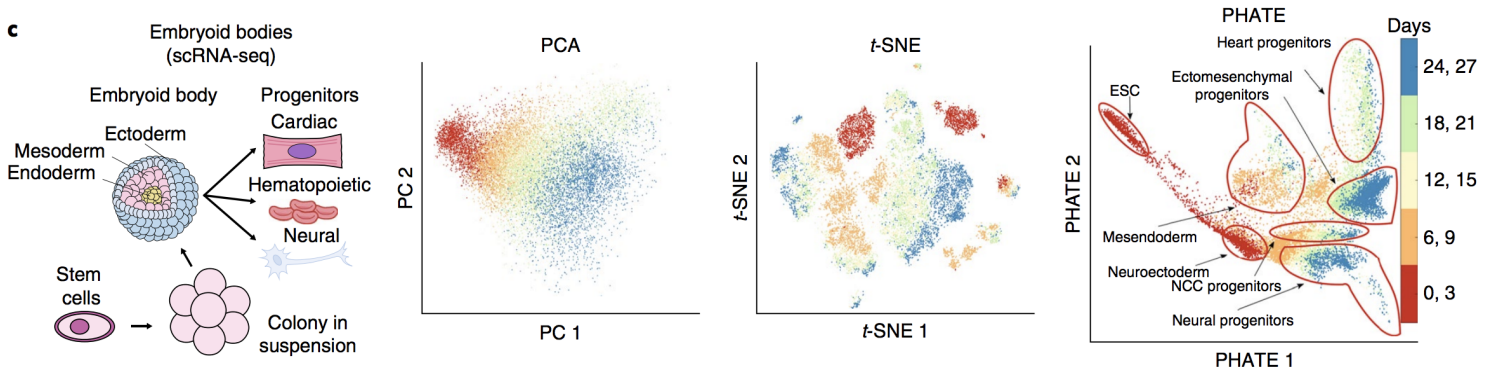


Figure: from Moon *et al.* Nature Biotechnology. 2019

Compare Phate to Other Dimension Reduction Techniques

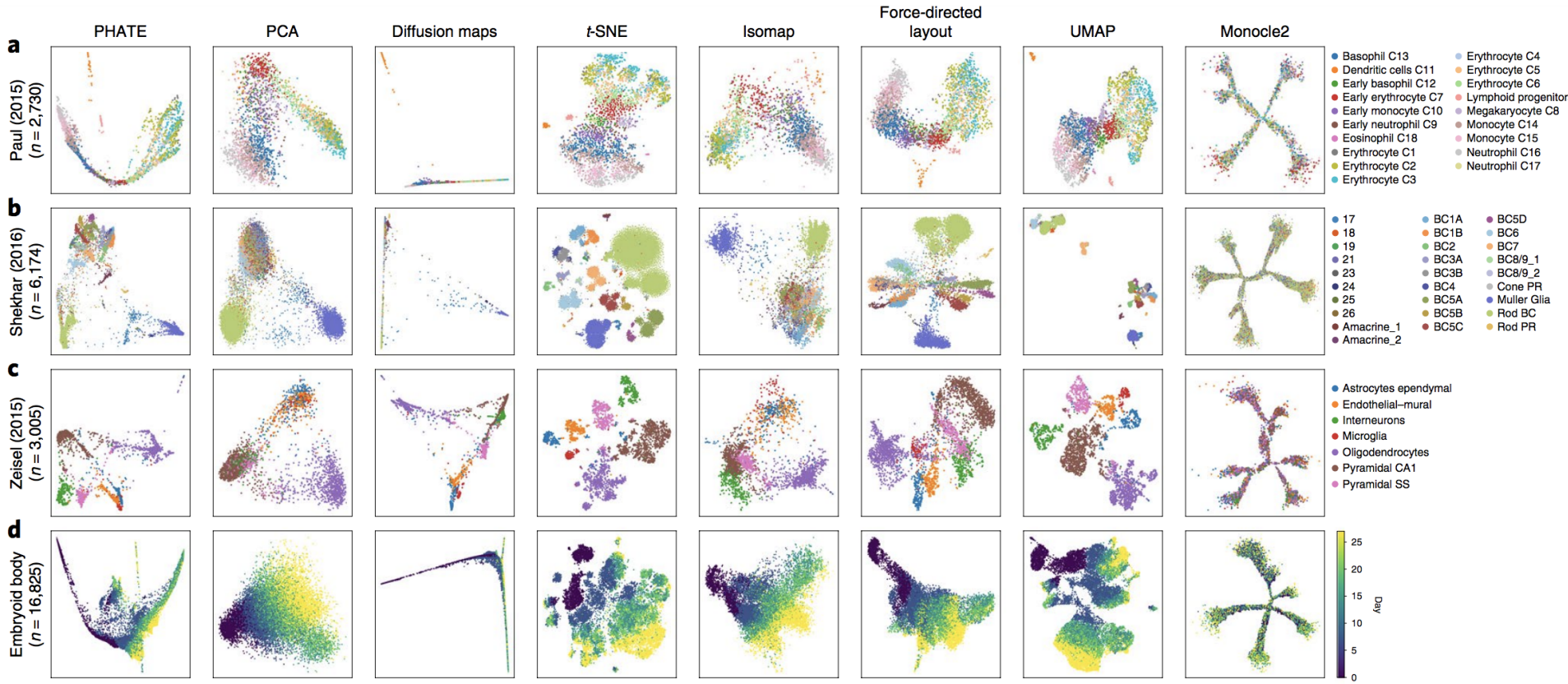


Figure: from Moon *et al.* Nature Biotechnology. 2019