Comp683: Computational Biology

Lecture 8

Feb 6, 2024

Today

- Finish up into to single-cell
  - Logistical issues : data transformation
- Drop out, imputation, noisy measurements

Review Questions

1. What are some pros and cons of manual analysis of single-cell data (manual gating)?
2. What is a very simple type of feature we can engineer from single-cell data to be used as input to a model?

What Are Cell Frequencies?

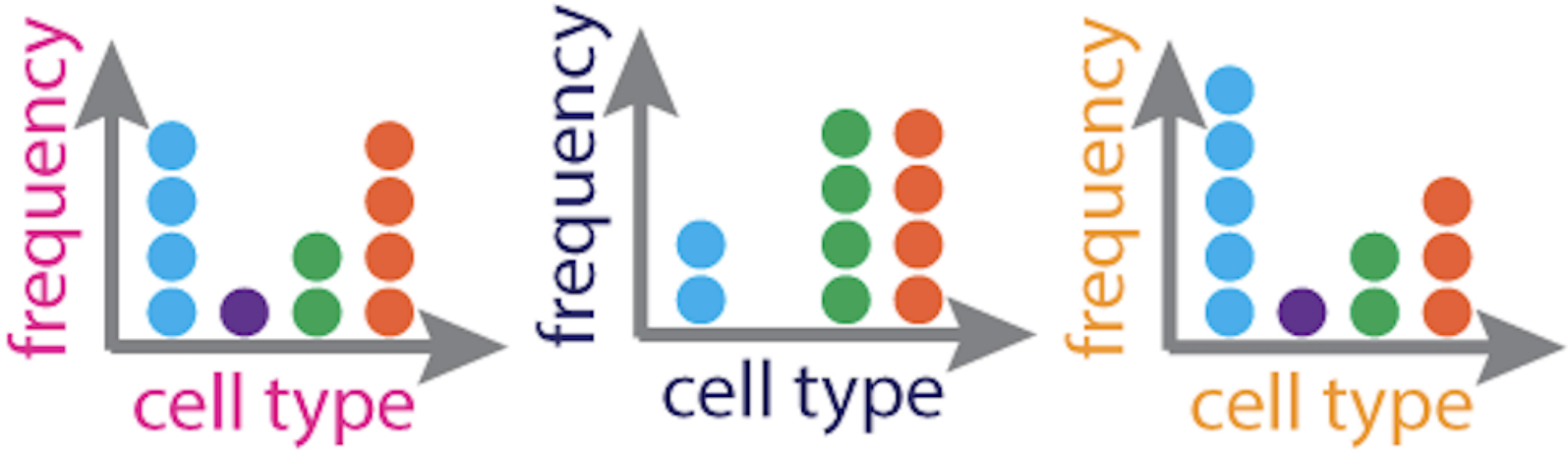


Figure: Simply count the number of cells in each profiled sample assigned to each specific type. This is a very simple hand engineered feature.

FCS file

- The collection of cells for each sample are stored in an ‘FCS‘ file. Cells are also called events.
- FCS stands for ‘Flow Cytometry Standard‘ and contains information about the experiment, like human interpretable names for markers, channel names, information about any performed normalization, etc.
- You will need to process each individual sample file.

Step 1: Separating Markers into Function vs Phenotypic vs Experimental

- There are 3 types of columns in an FCS file
- Phenotypic marker columns help us to characterize particular cell-populations (hint, usually starts with **CD**).
- Functional marker columns help us to quantify signaling or other function within a cell
- 'Junk' markers- help to keep track of which cells died, etc.
- After filtering out dead cells, we will only do analysis based on functional or phenotypic markers.

Example Marker Names from an FCS File

```
>>> sample.pnn_labels
['Time', 'Cell_length', 'DNA1', 'DNA2', 'CD45RA', 'CD133', 'CD19', 'CD22', 'CD11b', 'CD4', 'CD8', 'CD34', 'Flt3', 'CD20', 'CX
, 'CD45', 'CD123', 'CD321', 'CD14', 'CD33', 'CD47', 'CD11c', 'CD7', 'CD15', 'CD16', 'CD44', 'CD38', 'CD13', 'CD3', 'CD61', 'C
 'HLA-DR', 'CD64', 'CD41', 'Viability', 'file_number', 'event_number', 'label', 'individual']
```

Figure: For example, DNA1 and DNA2 help us to find dead cells. Markers like 'CD19' help us to find specific cell-types. 'CD19' in particular characterized B-cells! Event number is a junk marker example, which is just counting the cells.

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.

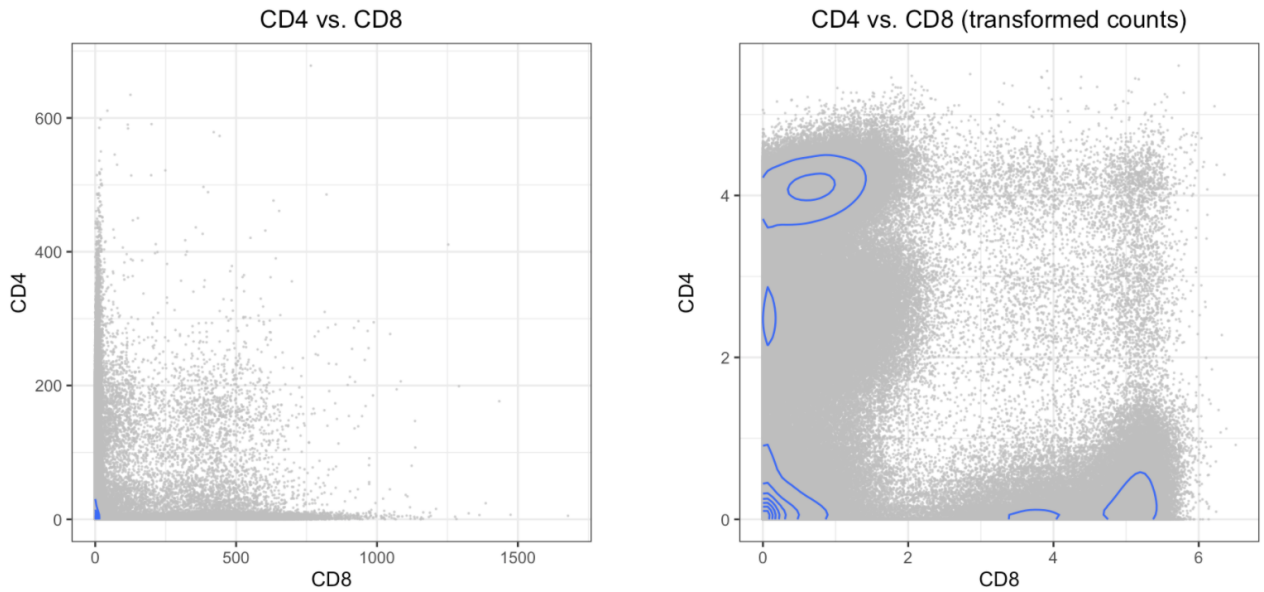


Figure: from `https://biosurf.org/cytof_data_scientist.html`

Arcsinh Transformation

In practice, especially with mass cytometry data, it is common practice to use an Arcsinh transformation, with co-factor aka scaling factor of 5. For count x, the transformed value $x' = \text{asinh}(\frac{1}{5}x)$

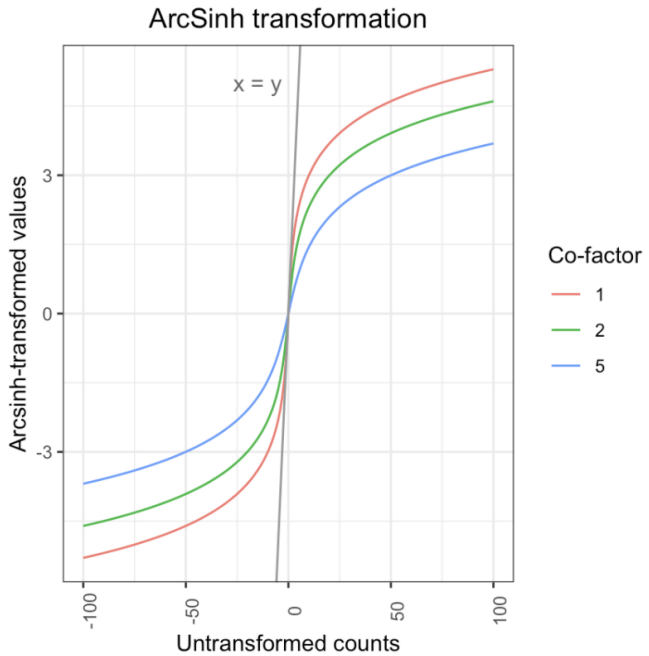


Figure: from `https://biosurf.org/cytof_data_scientist.html`

Effect of Normalization on Gating

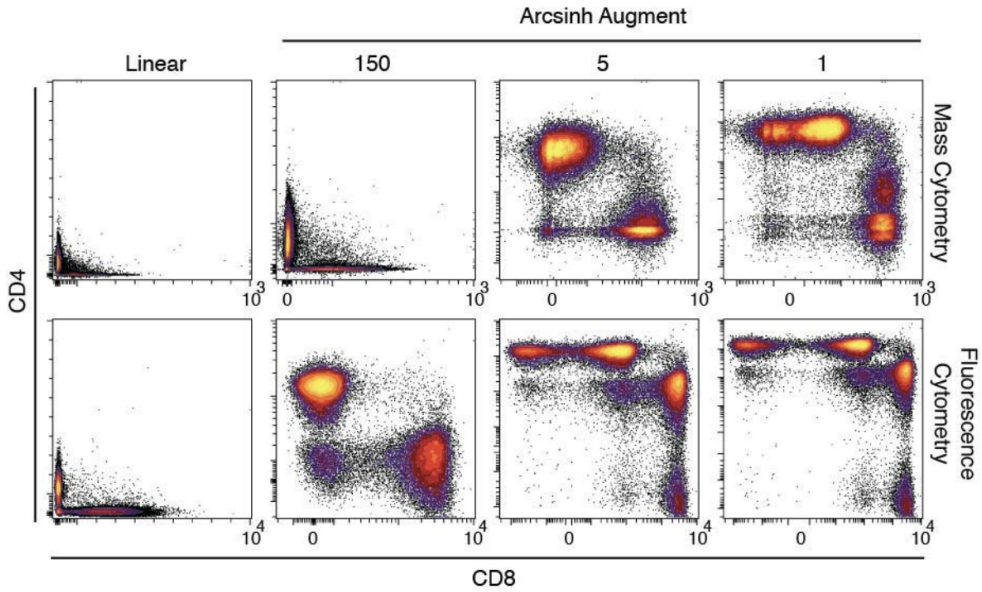


Figure: from Bendall *et al.* Science. 2011. The cofactor can cause sensitivity in the number of populations that emerge.

Tutorial

I have a tutorial for converting FCS files into matrices for both Python and R. `https://github.com/stanleyn/fcs_tutorial`.

Recap

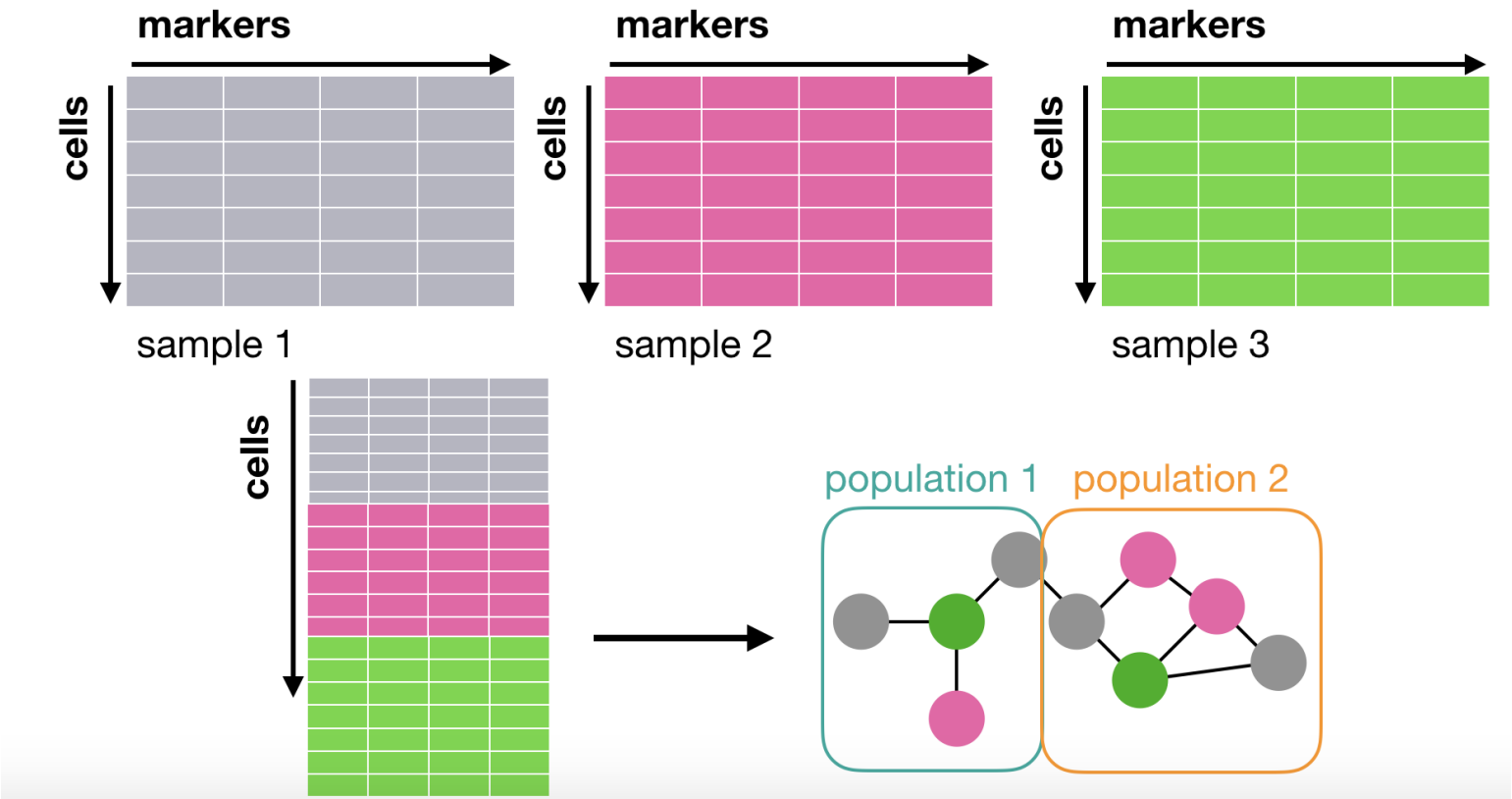


Figure: We are processing each FCS file, which corresponds to an individual sample. Ultimately cells are pooled across all samples.

To Automate Gating or to Not Automate Gating

Ask your collaborator to give you FCS files for each sample with only live cells included. If you ask them for gates, it will take them a lot of work. But you can make your own gates through unsupervised clustering!

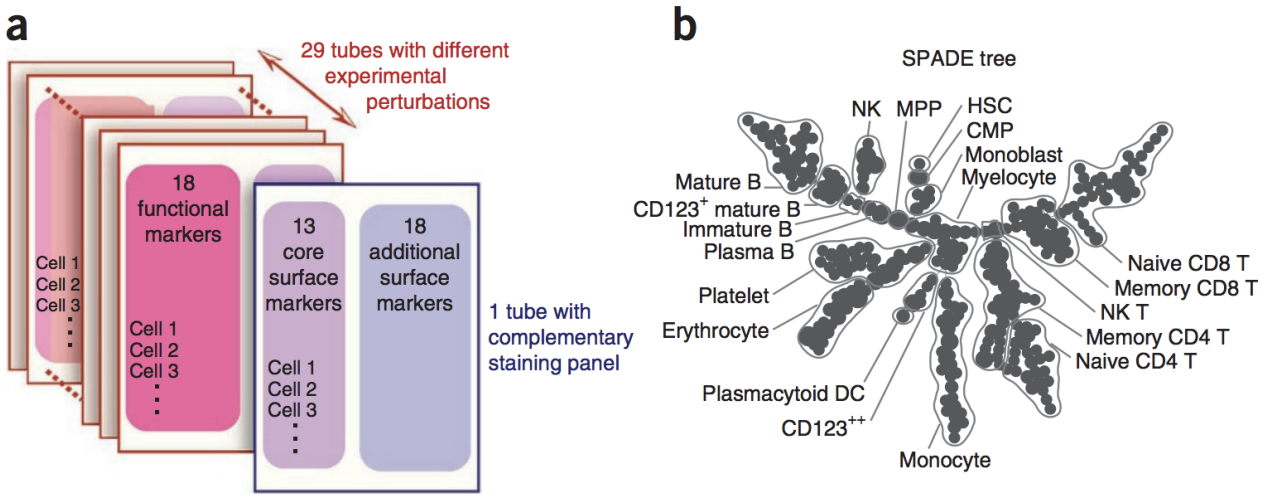


Figure: from Qiu *et al.* Nature Biotechnology. 2011. Do agglomerative hierarchical clustering based on the expression of measured markers.

Why Hierarchical Clustering?

It recapitulates our general understanding of cellular differentiation.

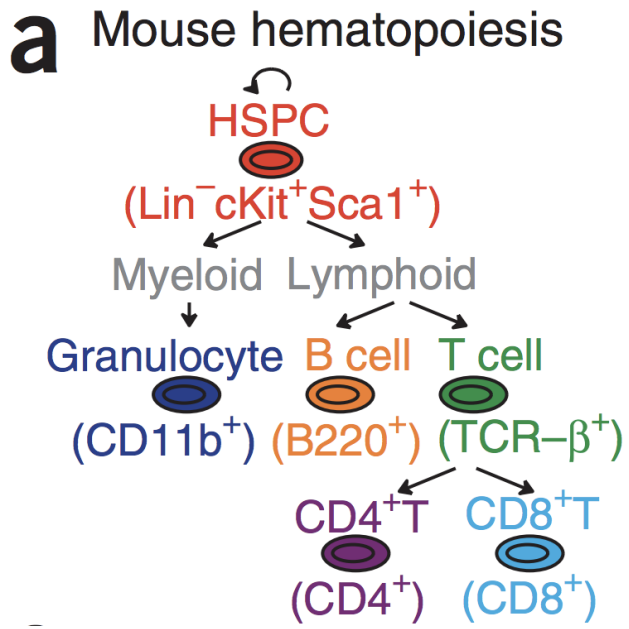


Figure: from Qiu *et al.* Nature Biotechnology. 2011. Cells can differentiate from Stem Cells to more specialized cell-types.

A Practical Consideration

- For N cells (and recall that N is large, like $>$100K), we need to compute an $N \times N$ distance matrix.
- If we have 100 samples, we will have around 100K $\times$ 100 total cells.
- We can't calculate all of those distances! So, right now, hierarchical clustering fails...

Question for You

Do you have any ideas about how you might reduce the size of this problem of the analysis of such a large number of cells across multiple samples?

Density-Dependent Downsampling

- Downsampling is a popular approach where a limited number of samples is samples for each FCS file.
- This is not ideal, because downsampling causes information loss.
- Spade does this is a density-dependent way, by sampling subsets of cells, and computing densities of their neighborhoods. Hence, Spade ensures that cells are represented across neighborhoods, especially in sparse neighborhoods.

Visualization in Spade : Cluster-Level

Construct a minimum spanning tree between clusters, based on the median expression of the markers.

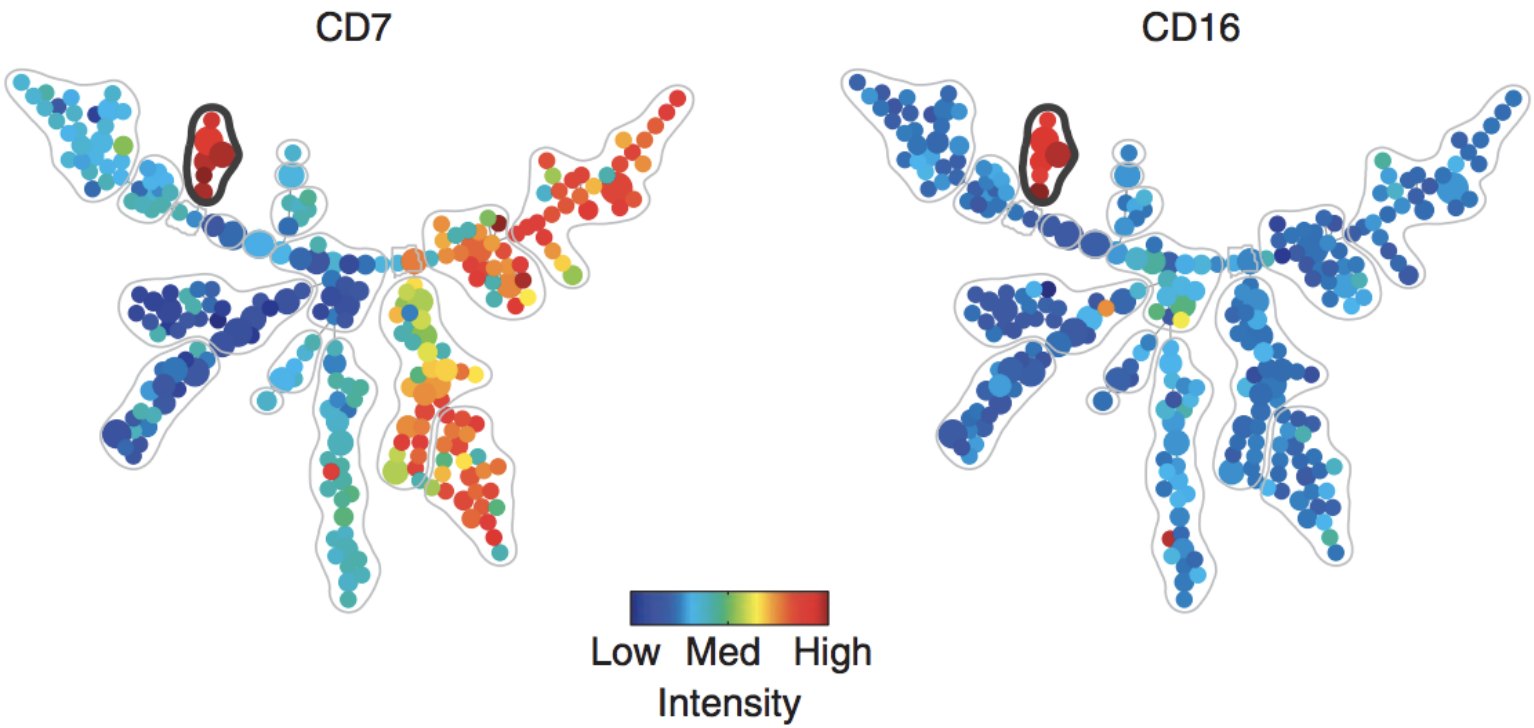


Figure: from Qiu *et al.* Nature Biotechnology. 2011.

Warning : Batch Effects

As with every biological assay, there is technical variation. For example, half the samples run on a machine in California, and the other half run on a machine in North Carolina. NC and CA samples might look very different from each other.

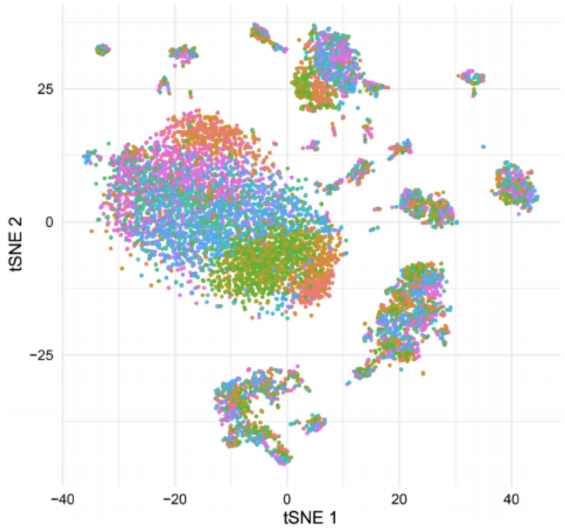


Figure: from Van Gassen *et al.* Cytometry A. 2019. Here cells are colored by batch.

Your job as a collaborator

- Batch effects are the most problematic when they correlate with the patient label that you are hoping to predict.
- If you could perfectly separate patients, it would be difficult to know if the success was because of a batch effect, or true biological signal.
- As a collaborator, you need to suggest for as much randomization as possible! E.g. have batches that have healthy and sick, for example

All of the Graph Knowledge Finally Applied....

Phenograph starts with a graph representation, sparsifies, and then clusters.

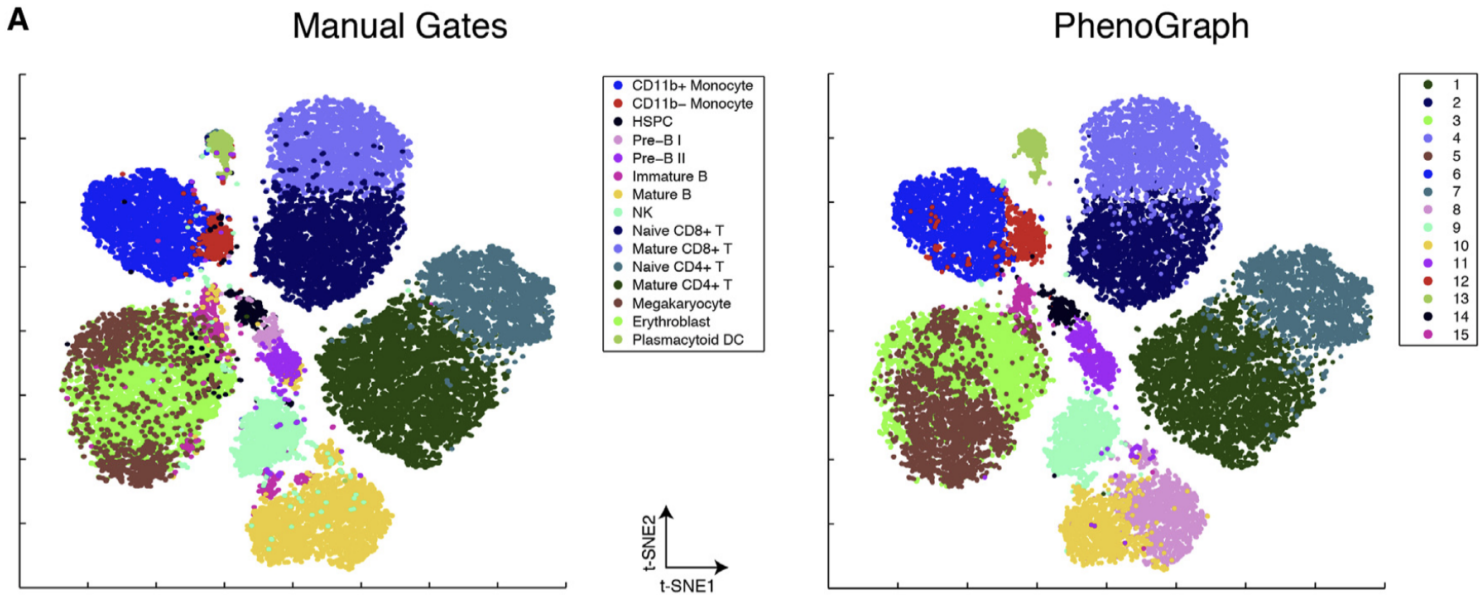


Figure: from Levine *et al.* Cell. 2015

A Direct Application of Louvain

- PhenoGraph is sold as being able to capture cell-populations of various sizes.
- A graph of cells is constructed for each samples
- Edges that exist are then turned into weighted edges, based on shared neighbors (a good contribution)
- The graph for each sample is partitioned with Louvain
- Clusters between samples are mapped with a metaclustering approach.

Communities Map to Cell-Populations

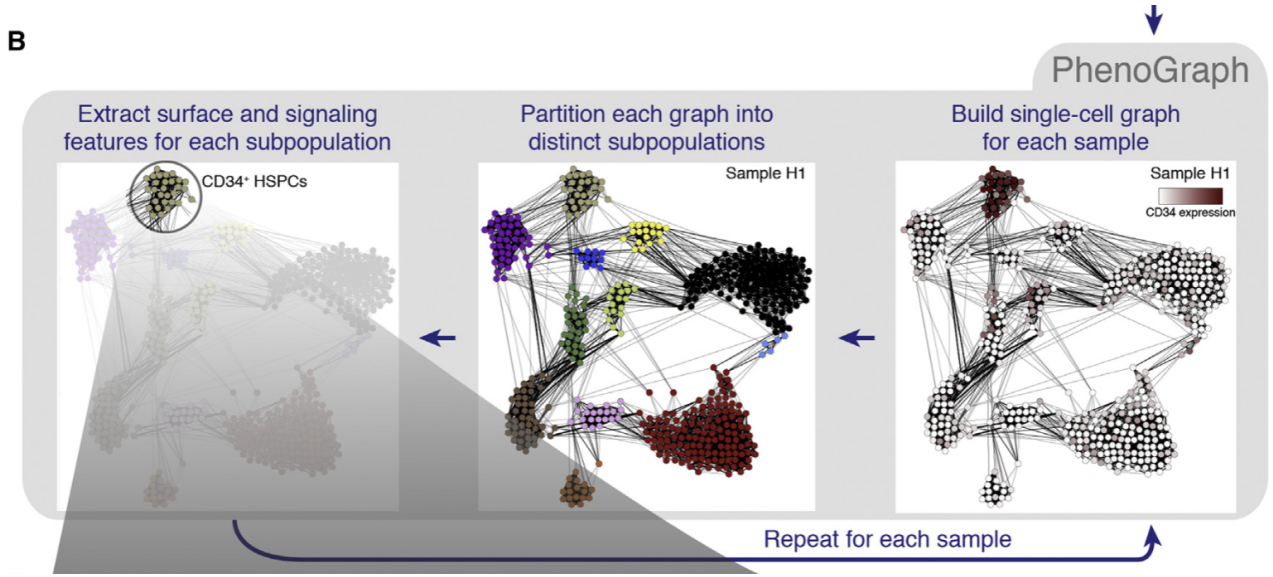


Figure: from Levine *et al.* Cell. 2015

Refining the kNN graph

Suppose an edge exists betweens nodes i and j. Then their edgeweight, w_{ij} is defined as the Jaccard Score of shared neighbors as,

$$W_{ij} = \frac{|v(i) \cap v(j)|}{|v(i) \cup v(j)|} \tag{1}$$

Here, $v(i)$ is the set of neighbors of node i. $|\cdot|$ is cardinality (number of neighbors)

Evaluating Similarity to Manual Gates

It's a bit reassuring to know that the performance (in terms of F-measure) is stable, regardless of the choice of k. [1]

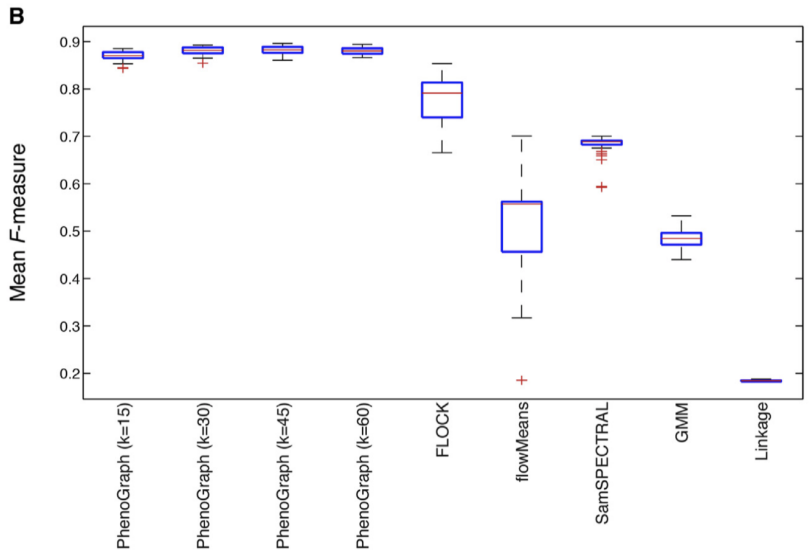


Figure: from Levine *et al.* Cell. 2015

[1] discussion of F measure `https://www.nature.com/articles/nmeth.2365.pdf`

Healthy vs AML with Cell Frequencies

Calculate the proportion of each sample's cells assigned to a particular metacluster. As you can see, there are differences in frequency between healthy (first few rows) and AML.

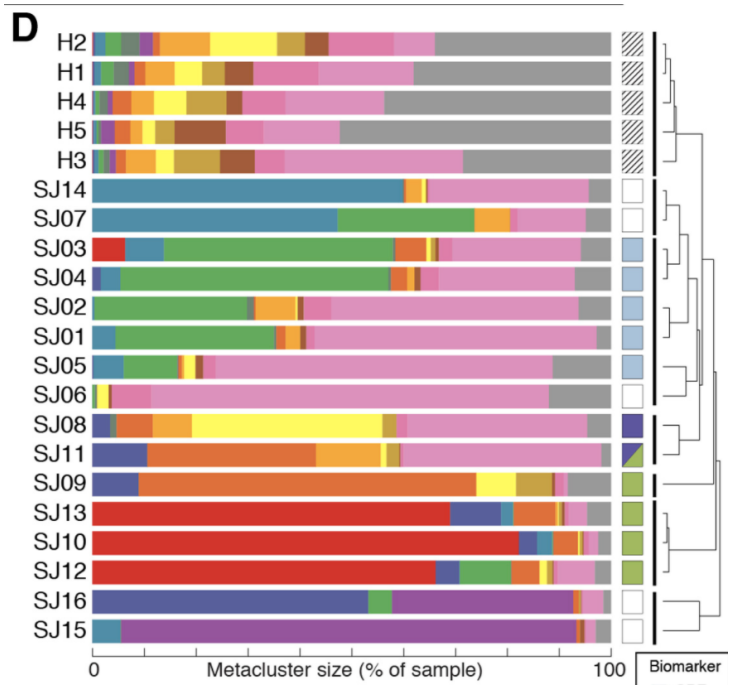


Figure: from Levine *et al.* Cell. 2015

You are ready for your first CyTOF dataset

If the data have signal, cell frequencies are a pretty powerful biologically interpretable feature that can allow us to classify patient samples.
To recap,

- Define per sample clusters
- Define metaclusters representing all samples
- Map cells from individual cells to metaclusters
- Compute frequency features
- You have now build a feature matrix that can be used for classification!

Example of a Powerful Frequency Feature

Frequency differences can be quite prominent in clinical settings. In this example, we are studying differences between patients who received steroid treatment after surgery, versus not.

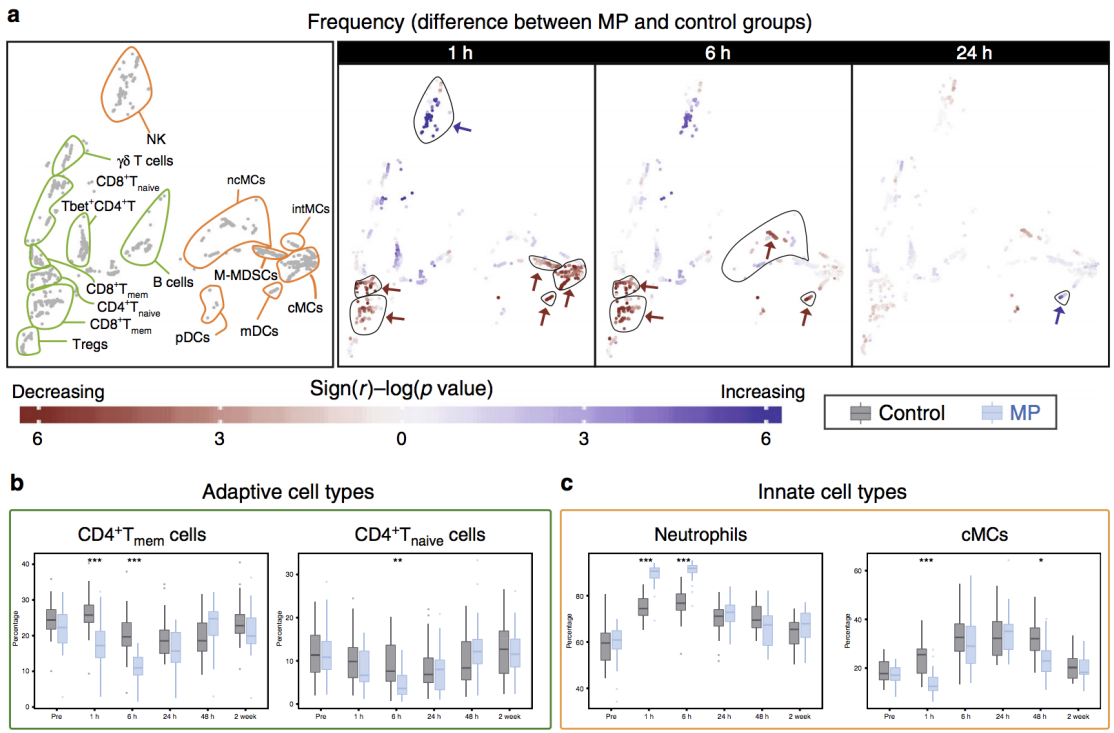


Figure: from Nature Communications. 2019.

Missing Data

- We don't live in a perfect world where we always have all of the data (could be missing features, entire samples, etc.)
- Imputation refers to an approach to 'fill in the blanks' for the missing information, based on the data that is available.
- De-noising is also a form of imputation, where you estimate and remove noise.

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)

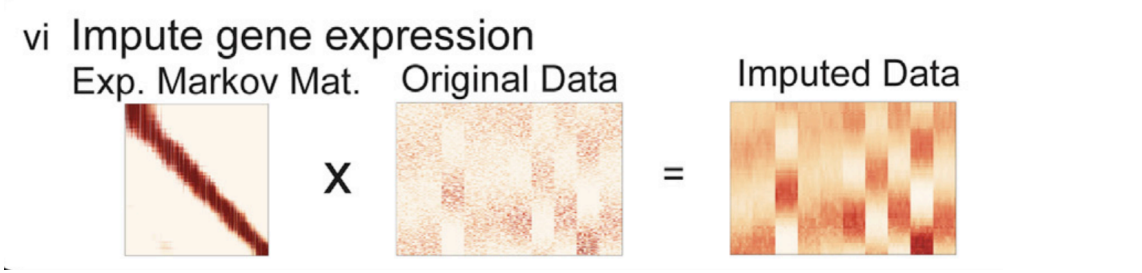


Figure: from van Dijk *et al.* Cell Systems. 2018

Accurately Representing Gene-Gene or Protein-Protein Interactions

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

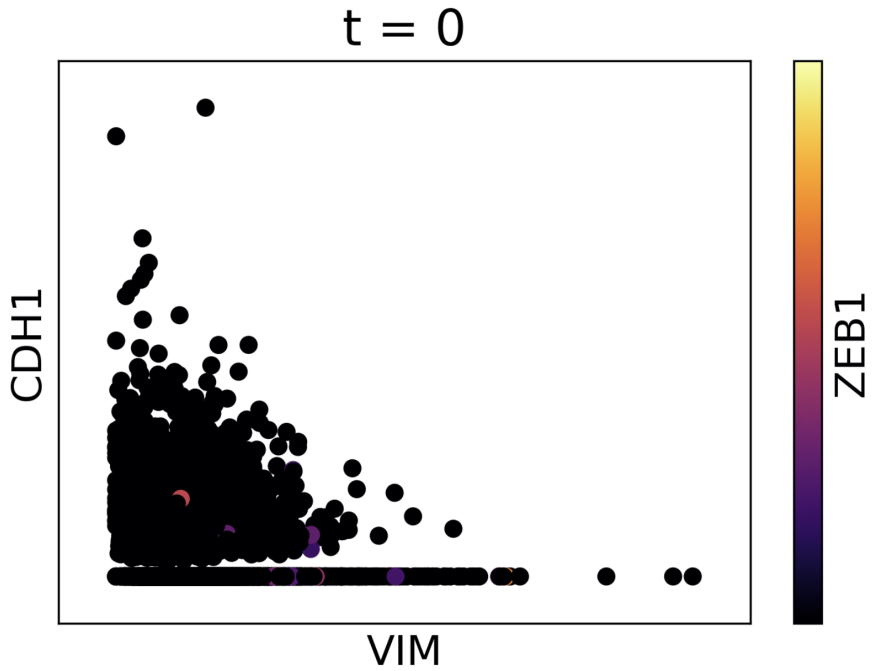
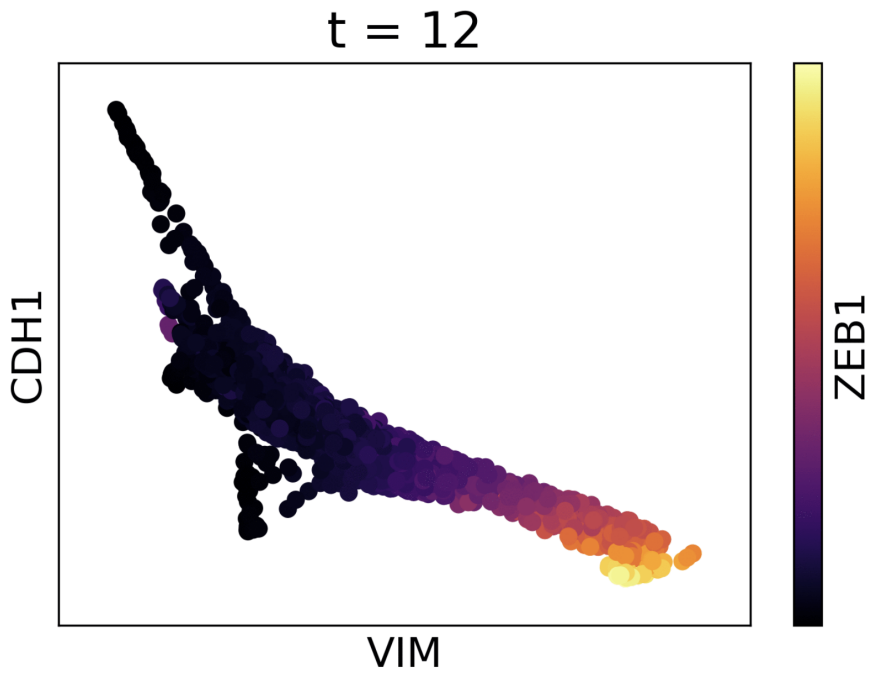


Figure: Note all of the 0 expression for CDH1

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

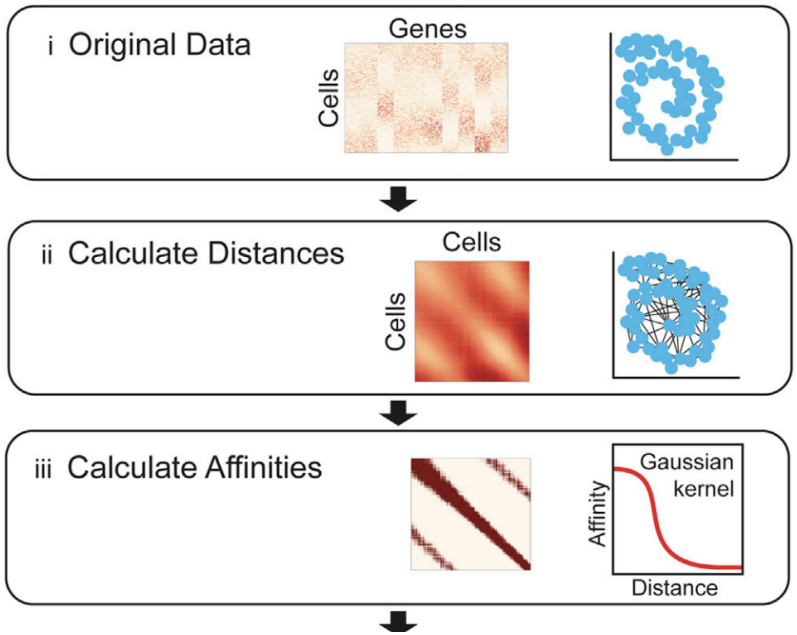
Here is an outline for the main idea.

- 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).

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



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 $k = 5$) [2].

[2] `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,

$$\mathbf{A} = \mathbf{A} + \mathbf{A}^T \tag{3}$$

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

$$M_{ij} = \frac{A_{ij}}{\sum_k A_{ik}} \tag{4}$$

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.

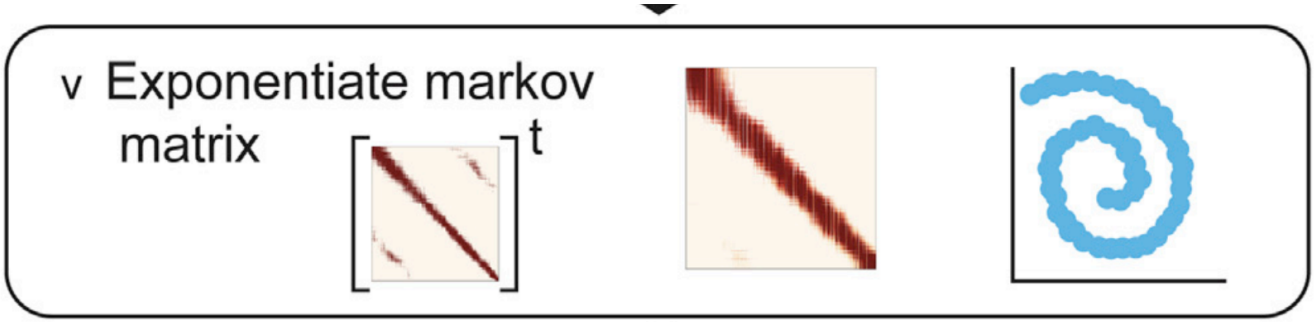


Figure: from van Dijk *et al.* Cell Systems. 2018