Dynamics on gene networks

Clayton W. Seitz

April 24, 2022

Data Summary

Research Strategy

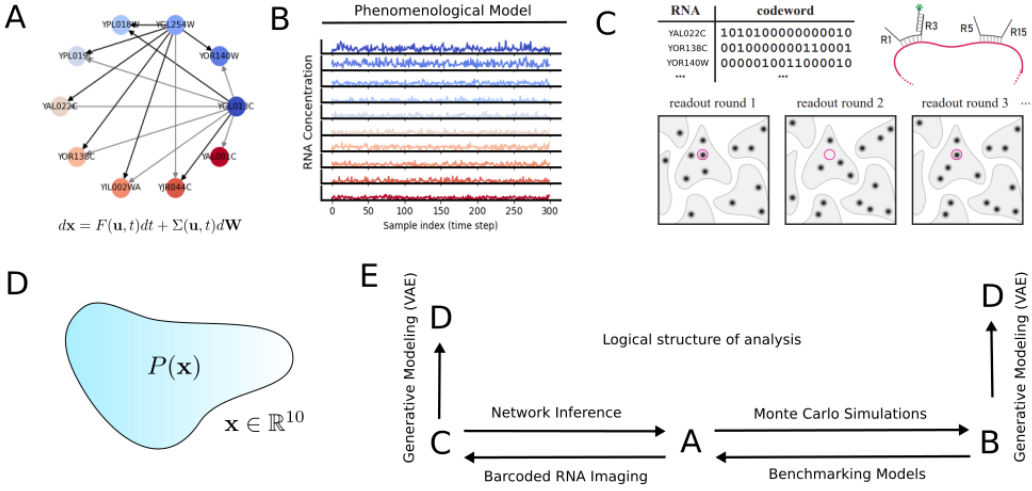


Figure 1: A *10-gene network sampled from Saccharomyces Genome Database (SGD).* **B** *Steady state values from Monte Carlo simulation.* **C** *Barcoding scheme for multiplexed RNA imaging (Alessio, 2021)* **D** *Cartoon of a joint distribution in higher dimensions* **E** *Relationships between components*

Simulating gene expression *in-silico*

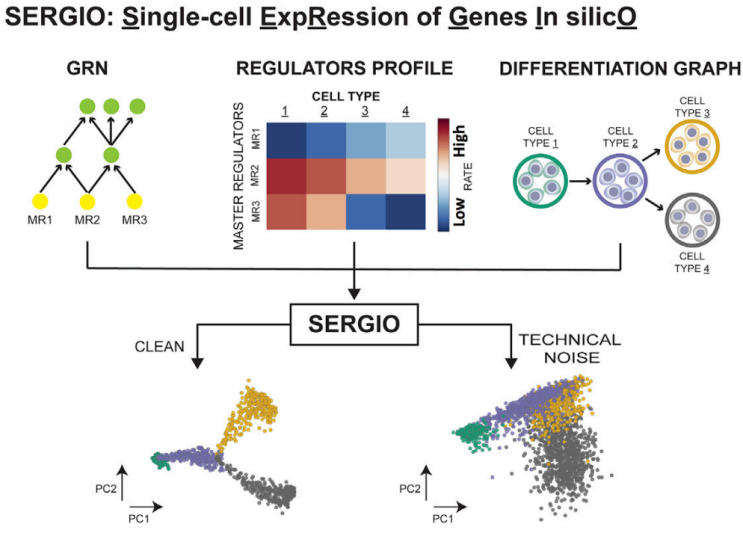


Figure 2: (Dibaeinia and Sinha, Cell Systems 2020)

Simulating gene expression *in-silico*

We can simulate the expression of a gene as a function of the levels of its regulators (TFs), as prescribed by a fixed gene regulatory network (GRN)

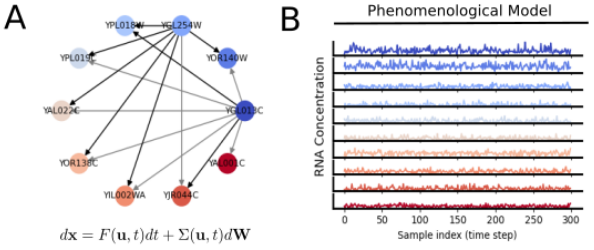
This is done via the chemical Langevin equation (Gillespie, 2000; Dibaeinia 2020):

$$\dot{x}_i = P_i(t) - \lambda_i x_i(t) + q_i \left(\sqrt{P_i(t)}\alpha + \sqrt{\lambda_i x_i}\beta \right)$$

The transcription rate of gene i depends non-linearly on its regulators:

$$P_i = \sum_j p_{ij} + b_i$$

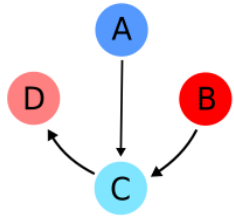
$$p_{ij} = m_{ij} \frac{x_{ij}^{n_{ij}}}{x_{ij}^{n_{ij}} + h_{ij}^{n_{ij}}}$$



Generating "cell types" for differentiation analyses

Genes A and B are so-called *master regulators*

$$\dot{x_A} = P_A - \lambda_A x_A + \phi(x_A)$$
$$\dot{x_B} = P_B - \lambda_B x_B + \phi(x_B)$$
$$\dot{x_C} = P_C(A, B) - \lambda_C x_C + \phi(x_A, x_B, x_C)$$
$$\dot{x_D} = P_D(C) - \lambda_D x_D + \phi(x_C, x_D)$$



ϕ summarizes the noise dependence. Tuning the basal expression rates P_A and P_B is one way of generating "cell types"

This is useful when benchmarking algorithms which detect genuine biological variability or experimental variability

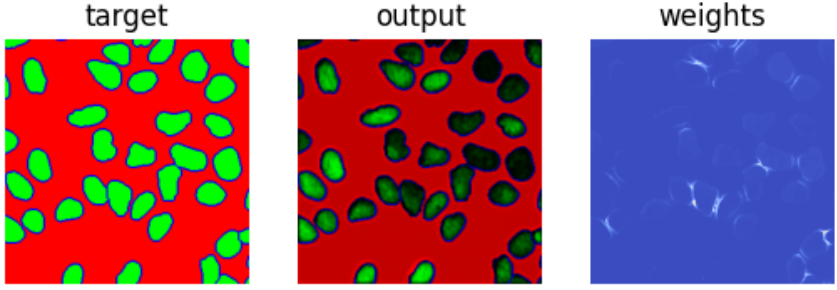
Convergence tests for Markov chains

GRN Inference from simulated data

Autoencoding simulated data

Training on BBBC039 U2OS Cells

BBBC039: 200 images, 160 train + 40 validation, 256 x 256 random crop

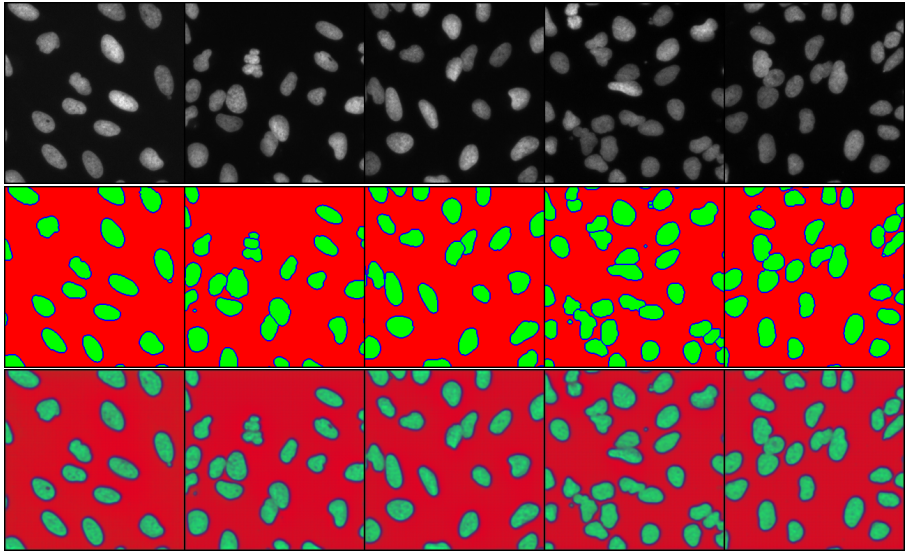


We train a 3-channel semantic segmentation model with **weighted** cross-entropy loss:

$$\mathcal{L} = \sum_{i,j} w_{ij} \log p_{ij}(\tilde{x}) = \sum_{i,j} w_{ij} \log \frac{\exp(-s_{ij}(\tilde{x}))}{\sum_{x \in \chi} \exp(-s_{ij}(\tilde{x}))}$$

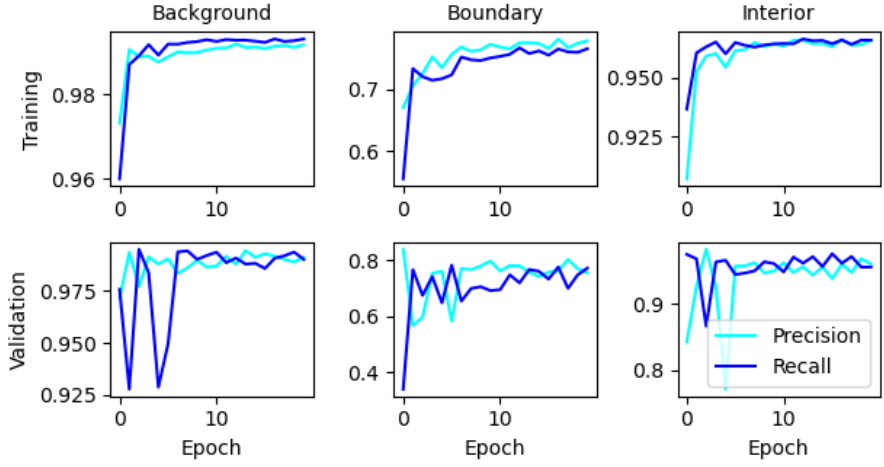
p_{ij} is the probability the model assigns a pixel to the true class $\tilde{x} \in \{\mathrm{a}, \mathrm{b}, \mathrm{c}\}$

Training on BBBC039 U2OS Cells



Training on BBBC039 U2OS Cells

Learning rate $\eta = 0.01$, Batch-size $B = 5$ (32 train iterations, 8 validation)



References I