Generating Data

Shalin Patel

January 22, 2021

Suppose we want to define an example GRN inference problem. In general, we approach this problem from the perspective of differential equations. We hope that there is a system of differential equations for a given regulatory network that is able to determine the time-based progression of expression. Specifically for a given gene g the expression of g, denoted by x_g is governed by

$$\partial x_g = F(x, \theta, t)$$

for a given F and parameters θ. In our case, we only consider $\{x_h \mid h \in N(g)\}$, or the expression profiles of neighboring genes in a proposed regulatory network.

We can do this is in julia with the `NeuralGRN.jl` package. To start load the nessary packages and define a few constants like initial conditions and the function F.

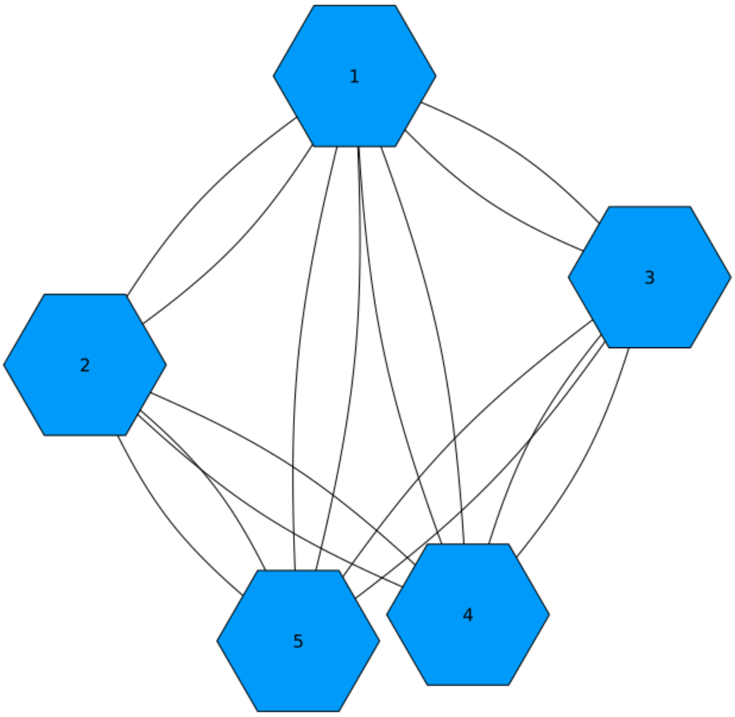
```
using NeuralGRN, LightGraphs, Plots, StatsPlots, LinearAlgebra, Distributions,
GraphRecipes
n = 5;
x_0 = rand(Normal(0, 0.2), n);
p = rand(Normal(), n, n);
propogate(u, p, t) = 10 * (u ⋅ p) * cos(5 * t) * exp(-0.5*t);
τ = 0:0.01:3 |> Array;
```

The F we chose for this example is

$$\partial x_g = 10(x_h \cdot \theta_g)\cos(5t)e^{-\frac{t}{2}}$$

More importantly, though, we need a computational graph that defines the relationships between different genes in our sample GRN. The following is a random graph, but any `SimpleGraph` would do.

```
g = erdos_renyi(n, 0.7);
plot(g, names=string.(1:n), nodesize = 0.4, fontsize = 10, fmt = :svg)
```

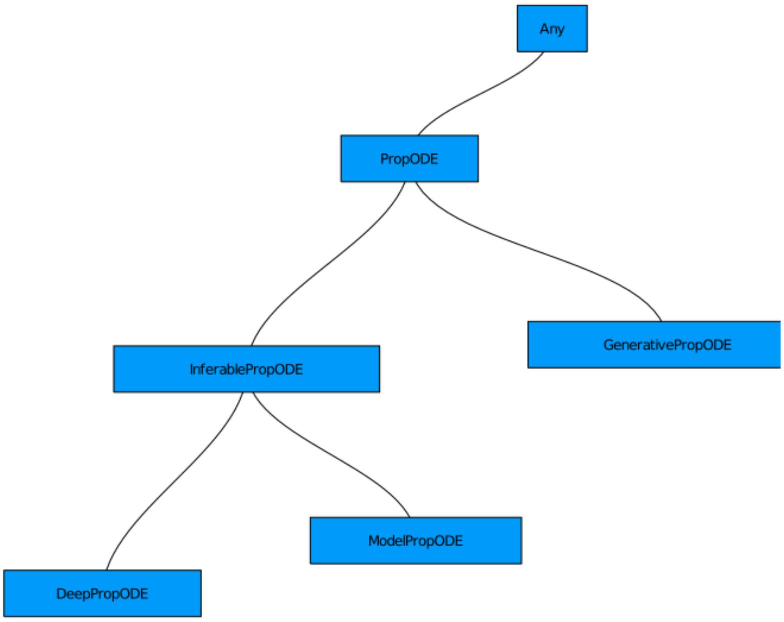


We now define a generative model using the inputs from above.

```
gen = GenerativePropODE(x_0, fill(propogate, n), p, g, τ);
```

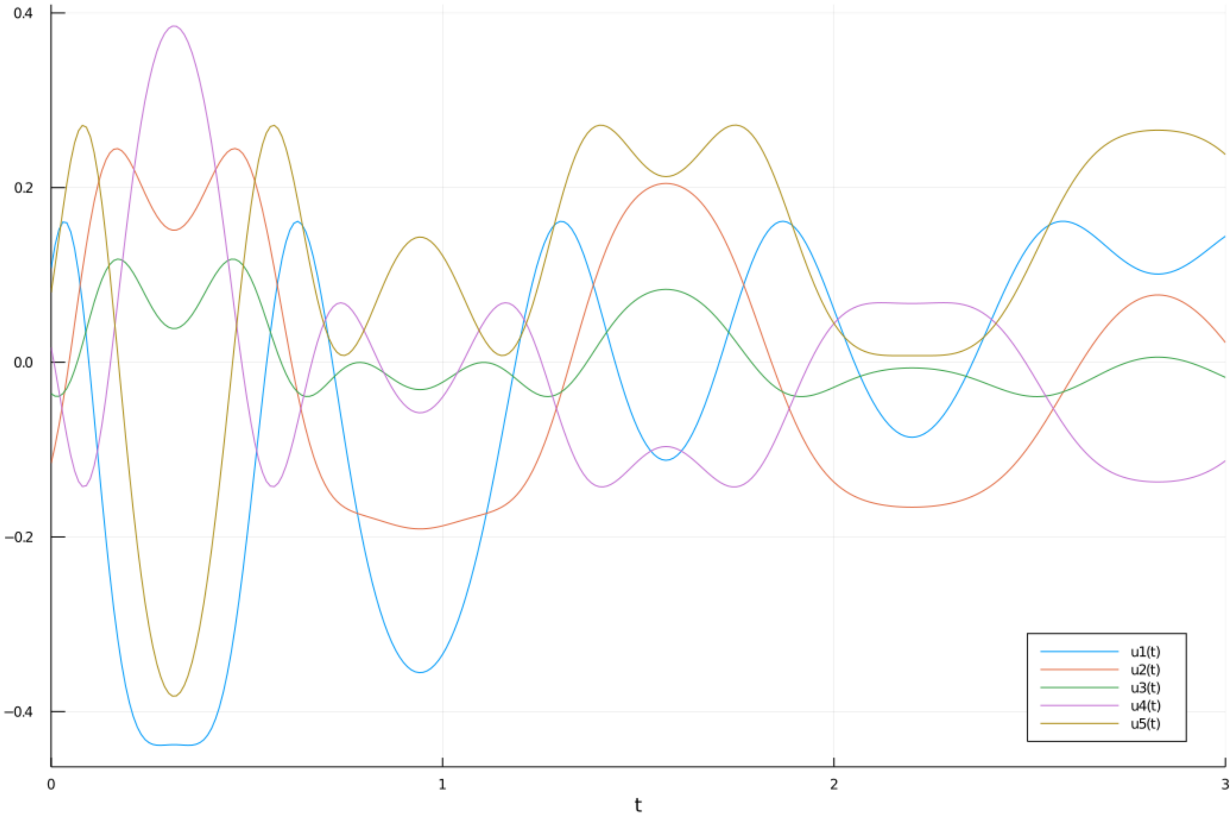
Using the `generate` function we can solve this system with the given parameters. This is all done courtesy of `DifferentialEquations.jl`. Note here that any subtype of `PropODE` can use the generate method which give inferred models based on real data the ability to project expression into the future. The following is a type diagram for the `PropODE` so you can see the different models in the works.

```
plot(NeuralGRN.PropODE, method = :tree, fontsize = 8, nodeshape = :rect, fmt = :svg)
```



We can run the inference and then plot the resulting data.

```
results = generate(gen);
plot(results, legend = :bottomright, fmt = :svg)
```


0.4
0.2
0.0
−0.2
−0.4
0
1
2
3
t
u1(t)
u2(t)
u3(t)
u4(t)
u5(t)