GATE: Software for the Analysis and Visualization of High-Dimensional Time-series Expression Data

Ben D. MacArthur[1,2,†], Alexander Lachmann[1,†], Ihor R. Lemischka[2] and Avi Ma'ayan [1,*]

[1] Department of Pharmacology and Systems Therapeutics, Systems Biology Center New York (SBCNY); [2] Department of Gene and Cell Medicine, Black Family Stem Cell Institute; Mount Sinai School of Medicine, One Gustave Levy Place, New York, NY 10029

*To whom correspondence should be addressed. †Contributed equally.

Supporting Text

GATE clustering and visualization- High-dimensional time-series are considered as an $N \times M$ data matrix $D \in \mathbb{R}^N \times \mathbb{R}^M$, where N is the number of observed molecular species (genes, proteins etc) and M is the number of observation time-points. Thus, each row of D gives the time-series expression of a single molecular species in the experiment. In order to examine the systems-level flow of information through the system we need to first provide a "spatial" representation of the data: that is, we need to appropriately project the data matrix D onto an appropriate geometric object. However, since regulatory networks are often very complex, and thus cannot be visualized clearly in two dimensions (that is, they are not planar graphs), network representations do not easily allow such visualization of the dynamic flow of information through the system. In order to visualize the flow of information through a system it is more informative to project the data onto a regular geometric object which can easily be represented in two dimensions. In this case, in order to visualize regulatory dynamics in a coherent manner we project the data matrix D onto regular hexagonal array $H.$ A hexagonal array is chosen since it ultimately presents the data in a form which is easier to visualize and interrogate than other tiling options (such as a rectangular tiling, for example). In order to provide a continuous geometric object with no boundaries we apply toroidal boundary conditions to the hexagonal array (that is, we associate the left and right hand sides of the array with each other, and the top and the bottom of the array with each other). Thus, we project the time-course data onto the surface of a hexagonally tiled torus. This projection ensures that there are no "special" places on the array and all molecular species are treated equally. In the absence of a defined regulatory network this projection may also be thought of as projecting the data onto a regular graph of degree 6. In mathematical terms, a projection of D onto H amounts to constructing a map $f: D \in \mathbb{R}^N \times \mathbb{R}^M \to H$ in which each row of the matrix D (each molecular time-series) is assigned to a unique hexagon $h_i \in H$. We denote as F the space of all such maps. However, not all maps $f(D) \in F$ will capture the systems-level regulatory dynamics equally well: for example, a random assignment of time-series to hexagons in the array will not (in general) capture the collective dynamics in the system since molecular species with similar expression patterns may not be near each other on the array. In order to construct the mapping which best captures collective dynamics we need to arrange the time-series on the array such that molecular species with similar expression patterns are near to each other on the array, while those with very different expression patterns are placed far apart. In order to do this we assign to each mapping $f(D) \in F$ the fitness

$$-1 \leq \mathrm{Fit}[f(D)] = \frac{\sum_{i=1}^{N} \sum_{j \in N_i} C_{ij}}{6N} \leq 1$$

where C_{ij} is the Pearson correlation coefficient between time-series i and j and N_i are the 6 neighbors of hexagon h_i. $\mathrm{Fit}[f(D)]$ measures how well a given arrangement captures the collective dynamics of the system as a whole: arrangements with low fitness do not capture systems level dynamics, while arrangements with high-fitness capture systems level dynamics well. The challenge, then, is to find the arrangement of the time-series on the array which has close to maximal fitness. This translates into a problem of combinatorial optimization: given the space F of maps $f: D \in \mathbb{R}^N \times \mathbb{R}^M \to H$ and the fitness function $\mathrm{Fit}(f)$, find $f_*(D) \in F$ such that

$$\mathrm{Fit}[f_*(D)] \cong \mathrm{Max}_{f(D)\in F}\{\mathrm{Fit}[f(D)]\}.$$

In order to solve this problem we employed a simulated annealing algorithm. This is a standard approach in combinatorial optimization. In practice we find that this approach gives near-optimal solutions within a few minutes of computational time on a standard desktop computer. In order to determine the intrinsic fit of the data to the array we also define the misfit parameter.

$$0 \leq \mathrm{Misfit}(D) \cong 1 - \mathrm{Fit}\,[f_*(D)] \leq 2$$

Data sets which are intrinsically mismatched to the hexagonal array geometry have a high misfit, while data sets which fit well have a low misfit. Practically, we find that, although each experiment has its own natural geometry, most short time-series fit this model remarkably well and have a correspondingly low misfit. Once the close to optimal map $f_*(D)$ from the data to the array has been identified a movie of the systems-level dynamics is generated by assigning a color to each hexagon $h \in H$ which changes over time according to the expression level of the molecular species to which it is assigned. In order to create a movie which interpolates smoothly between time-points we implemented a piecewise cubic Hermite interpolation prior to visualization. This smoothing of time-series expression data does not add (or remove) information, but simply allows a smooth transition between molecular snapshots enabling visualization of the dynamic flow of information through the system over time. In order to color each hexagon $h \in H$ appropriately we also normalized each time-series with respect to its day 0 expression value and such that all expression series range from 0 to 1.

Pseudo-code of clustering algorithm-

```
Function performClustering (maxTemperature)
Preprocessing
1.      Assign each gene with a position on the hexagonal grid randomly
2.      Compute all pairwise correlations of the gene time lines
3.      Start timer reducing temperature over time
oldFit = getGlobalFitness();              // compute the fitness function with pre-computed
while (temperature > 0){
        swap ();                          // swap position of two random genes
        newFit = getGlobalFitness();      // compute the fitness function
        r = getRandom();                  // random number between 0 and 1
        if (r < getProb(newFit, oldFit, (temperature/maxTemperature))){
                oldFit= newFit;
        }
        else{
                undoSwap();               // switch back position of genes
        }
}
finalize();                               // run into  local optimum
```