1 Introduction

This project follows the general methods used in *Martin et al.* to infer gene regulatory networks from time series expression microarray data. All code was written by me based on the described methods. It then begins to explore some of the dynamics of the network. Lacking access to any of the datasets used in *Martin et al*, The dataset used is the yeast mitotic cell cycle from *Cho et al.*

1.1 Overview of GRN Inference

Elucidation of gene regulatory networks is a critical problem in molecular biology. Many computational methods exist, among them boolean networks, Bayesian networks, and differential equations. Boolean networks were chosen for this product due to their simplicity in modeling. Boolean networks take as inputs discrete states where each node (gene) is represented by 1 or 0; 1 indicating that the gene is expressed, 0 indicating that the gene is not expressed. The next discrete state depends only on a function of the previous state.

2 Medthods

Two time-series gene expression datasets were used in *Martin et al.*: IL-2 stimulated murine T-cells and LPS-stimulated macrophages. For this project, 17 time points in the yeast mitotic cell cycle were used. After importing the text file (EDS16data.txt), the data were normalized by row using a function using Matlab's built in norm funtion.

```
normed=rownormalize(EDS16data)
```

Listing 1: rownormalize.m

```
function normed = rownormalize(matrix)
normed=zeros(size(matrix,1),size(matrix,2));
for i=1:size(matrix,1)
    normed(i,:)=matrix(i,:)/norm(matrix(i,:));
end
end
```

As in *Martin et al.* the expression profiles were then clustered into groups called "metagenes" in order to save computational power. The k-means algorithm was used for this purpose. In order to choose a suitable k, values between 4 and 100 were tested for "internal consistency" as defined by *Martin et al.* First, singular value decomposition was performed on the clustered data to yield $X = USV$, "where U and V are orthogonal matrices and S is a diagonal matrix whose entries describe the importance of the columns of U and V." A high ratio of $S_1 : S_2$ means that the majority of the data in a cluster can be captured in a single row (high internal consistency). The average internal consistency for each group of clusters was plotted in Figure 1.

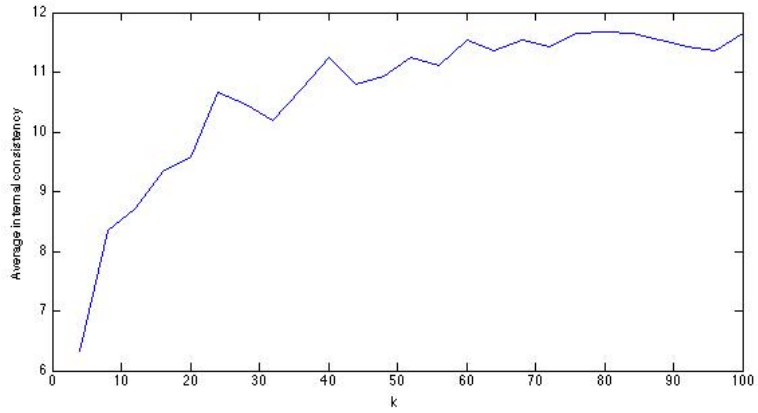
Listing 2: bestk.m

```
function ics = bestk(matrix)
ics=zeros(25,1);
for i=4:4:100
    try
        ics(i/4)=consistency(matrix,i);
    catch
        ics(i/4)=0;
    end
end
plot(ics);
end
```

Listing 3: consistency.m

```
function cons = consistency(matrix,k)
a=kmeans(matrix,k);
total=0;
for i=1:k
    [U,S,V] = svd(matrix(a==i,:));
    total=total+S(1,1)/S(2,2);
end
cons=total/k;
end
```

Figure 1: Average internal consistency for increasing values of k



The average internal consistency increases quickly at first and then begins to level off. A local maximum at $k = 24$ was chosen for the clustering.

```
clusters=kmeans(normed,24);
```

Martin et al. then used support vector regression to model each cluster and thresholded the data such that each metagene is represented by a 1 (expressed) 50% of the time. I don't prefer this method due to the assumption that genes are expressed exactly half the time (an examination of the original data shows this is intuitively not the case). Instead, I binarized the expression profile (binarize.m) of each gene in the normalized dataset using k-means clustering with k=2. The cluster with the higher average had a value of 1 assigned to all time points and the lower cluster had a value of 0 assigned to all time points.

Listing 4: binarize.m

```
newEDS=zeros(size(EDS16data,1),size(EDS16data,2));
for i=1:size(EDS16data,1)
    try
        x=kmeans(EDS16data(i,:),2);
        y=mean(EDS16data(i,x==1));
        z=mean(EDS16data(i,x==2));
        if y>z
            a=1;
        else
            a=2;
```

```
        end
        newEDS(i,x==a)=1;
    end
end
clearvars a i x y z;
```

Once the original dataset was binarized, each gene "voted" for its metagene's expression at time t (binarizemetagene.m). In the event of a tie, the expression was assigned a value of 0.5 and thus excluded from the succeeding analysis.

Listing 5: binarizemetagene.m

```
function metamatrix = binarizemetagene(bmatrix,clusters)
metamatrix=zeros(max(clusters),size(bmatrix,2));
for i=1:max(clusters)
    clustersize=length(find(clusters==i));
    for j=1:size(bmatrix,2)
        clustersum=sum(bmatrix(clusters==i,j));
        if clustersum>clustersize/2
            metamatrix(i,j)=1;
        elseif clustersum==clustersize/2
            metamatrix(i,j)=0.5;
        end
    end
end
end
```

Once the data preprocessing was complete, I could begin inferring functional relationships between metagenes.

Network inference was done by evaluating candidate boolean functions one by one. The routine isfunction.m takes as input a possible combination of activators and inhibitors for a target node. For each time t, if the boolean function of the state of the potential regulators can accurately describe the state of the target node at time t+1, the function is retained.

Listing 6: isfunction.m

```
function bool = isfunction(matrix,node,activators,inhibitors)
bool=0;
```

```
for i=1:size(matrix,2)-1
    if or(prod(matrix(inhibitors,i))==1,prod(matrix(activators,i))==1)
        if (1-prod(matrix(inhibitors,i)))*prod(matrix(activators,i))~=matrix(node,i+1
            %not a function
            bool=0;
            break
        else
            bool=1;
        end
    end
end
end
```

To simplify the procedure, the boolean function was represented mathematically as
$targetstate = (1 - product(allinhibitorstates)) * product(allactivatorstates)$
Inferfunction.m takes as input a set of possible regulators and determines if a boolean function of that set exists that could explain the behavior of a node throughout the network. To be returned as a possible function, it must be confirmed by at least time step and not be contradicted by any time steps.

Listing 7: inferfunction.m

```
function f = inferfunction(matrix,target,regulators)
f={};
for i=0:length(regulators)
    combs=combnk(regulators,i);
    for j=1:size(combs,1)
        if isfunction(matrix,target,combs(j,:),setdiff(regulators,combs(j,:)))==1
            f={regulators,combs(j,:),setdiff(regulators,combs(j,:))};
            break
        end
    end
    if ~isempty(f)
        break
    end
end
end
```

Infernetwork.m takes as input a binarized matrix, and provides every combination of 1, 2, or 3 nodes that can regulate each target node to inferfunction.m

Listing 8: infernetwork.m

```
function network = infernetwork(matrix)
network=cell(size(matrix,1),1);
for i=1:size(matrix,1)
    for j=1:3
        combs=combnk(1:size(matrix,1),j);
        for k=1:size(combs,1)
            f=inferfunction(matrix,i,combs(k,:));
            if ~isempty(f)
                network{i}=[network{i};f];
            end
        end
    end
end
end
```

Trimregulators.m takes the list of boolean functions for each node and eliminates each superset of regulators for which a subset already exists. If a gene a has its expression inhibited by a complex of gene products from b and c, it is not necessary to examine whether it is also inhibited by a combination of gene products from b, c, and d.

Listing 9: trimregulators.m

```
function trimmed = trimregulators(network)
trimmed=cell(size(network,1),1);
for i=1:size(network,1) %loop through nodes
    for j=1:size(network{i},1) %loop through list of activators/inhibitors
        regulators=network{i}{j,1};
        if ~isempty(trimmed{i,1})
            if ~isempty(regulators)
                subsetexists=0;
                for k=1:size(trimmed{i},1)
                    if all(ismember(trimmed{i}{k,1},regulators))
                        subsetexists=1;
                        break
```

```
                    end
                end
                if subsetexists==0
                    trimmed{i}=[trimmed{i};network{i}(j,:)];
                end
            end
        else
            trimmed{i}=network{i}(j,:);
        end
    end
end
end
```

If a network is defined by one boolean function corresponding to the expression of each node, the number of possible networks can be counted with countnetworks.m

Listing 10: countnetworks.m

```
function n = countnetworks(mynetwork)
n=1;
for i=1:size(mynetwork,1)
    if size(mynetwork{i},1)>0
        n=n*size(mynetwork{i},1);
    end
end
end
```

A random network can be selected with samplenetworks.m, which returns a column vector corresponding to the index of the chosen boolean function within each cell.

Listing 11: samplenetworks.m

```
function sample = samplenetworks(network)
sample=zeros(size(network,1),1);
for i=1:size(network,1)
    sample(i)=ceil(rand*size(network{i},1));
end
end
```

Taking as input the initial condition, the randomly sampled network, and the number of time steps, runnetwork.m produces the binary state of each gene at

each time step in the randomly selected network.

Listing 12: runnetwork.m

```
function sim = runnetwork(initial,network,samples,T)
nextstate=initial;
sim=zeros(length(initial),T+1);
sim(:,1)=initial;
for i=1:T
    for j=1:length(nextstate)
        if samples(j)>0
            if isempty(network{j}{samples(j),3})
                inhibitors=0;
            else
                inhibitors=prod(initial(network{j}{samples(j),3}));
            end
            if inhibitors==1
                nextstate(j)=0;
            else
                if isempty(network{j}{samples(j),2})
                    activators=0;
                else
                    activators=prod(initial(network{j}{samples(j),2}));
                end
                if activators==1
                    nextstate(j)=1;
                end
            end
        end
    end
    sim(:,i+1)=nextstate;
    initial=nextstate;
end
end
```

Findnetwork.m randomly samples 100,000 networks and compares their dynamics through all time steps with the observed network. Networks with an average of 70% agreement across all time steps are returned.

Listing 13: findnetwork.m

```
function mynetworks = findnetwork(network,initial,terminal,T,matrix)
mynetworks=cell(length(initial),0);
for i=1:100000
    sample=samplenetworks(network);
    h=runnetwork(initial,network,sample,T);
    howmatch=sum(sum(h==matrix));
    if howmatch>=286
        newnetwork=cell(length(initial),1);
        for j=1:length(initial)
            newnetwork{j}=network{j}(sample(j),:);
        end
        mynetworks=[mynetworks,newnetwork];
    end
end
end
```

Attractors.m takes a network and runs it until it reaches a steady state. It returns the time point at which the steady state first occurs along with the network simulation up to that point.

Listing 14: attractors.m

```
function [t,allstates] = attractor(network,initial)
allstates=initial;
t=inf;
for i=1:100
    nextstate=initial;
    for j=1:length(initial)
        if isempty(network{j}{3})
            inhibitors=0;
        else
            inhibitors=prod(initial(network{j}{3}));
        end
        if inhibitors==1
            nextstate(j)=0;
        else
            if isempty(network{j}{2})
                activators=0;
            else
```

```
                activators=prod(initial(network{j}{2}));
            end
            if activators==1
                nextstate(j)=1;
            end
        end
    end
    for k=1:size(allstates,2)
        if allstates(:,k)==nextstate
            t=i;
        end
    end
    allstates=[allstates,nextstate];
    if t~=inf
        break
    end
end
end
```

3 Results

Because of a lack of access to the data used by *Martin et al.*, one set of time series data for the yeast mitotic cell cycle was used. These data generated a total of 9.6267e+61 possible networks, of which analyzing all is computationally prohibitive. A smaller, more robust list of potential networks could be generated by using additional time series data sets, particularly with external stimulation. Of the 100,000 randomly sample networks, four met the criteria of aligning with the observed data 70% of the time. They are shown below.

All four networks converged on a steady state at timepoint 2.

4 Conclusion

The lack of consensus among the four generated networks, combined with the early convergence on a steady state, indicate major problems with the results. Because the methods in *Martin et al.* were followed closely, the issues likely lie with the limited training data and computational power. More training data would

Metagene	Network1		Network 2		Network 3		Network 4	
	Activators	Inhibitors	Activators	Inhibitors	Activators	Inhibitors	Activators	Inhibitors
1	21	13	[4,19]	3	[5,14]	20	[6,10]	14
2	[7,15]	21	5	[4,22]	[7,15]	21	[3,22]	21
3	20	2	[12,21]	5	[4,24]	10	[1,15]	19
4	8	[3,6]	[12,22]	24	18	[9,21]	[12,15]	20
5	1	[16,18]	[15,24]	3	[7,22]	14	16	[6,17]
6	[6,17]	16	[2,12]	14	[5,21]	16	[15,21]	20
7	3	[9,13]	3	[20,22]	3	[6,20]	16	[3,20]
8	[7,14]	20	1	16	[1,23]	13	[1,19]	24
9	8	[3,22]	[9,12]	14	10	[6,17]	21	13
10	[3,14]	22	[3,5]	14	14	13	[5,7]	22
11	14	[5,12]	8	[1,13]	3	[19,22]	17	[2,3]
12	13	1	14	8	[1,17]	8	21	[3,9]
13	[8,15]	10	[12,22]	19	14	[17,24]	[6,10]	7
14	[7,21]	10	[6,10]	19	[3,16]	17	[3,8]	4
15	14	[17,20]	14	[17,20]	18	[1,21]	24	[10,13]
16	[3,18]	24	[2,18]	24	[6,16]	24	[4,24]	3
17	[1,24]	6	16	[11,20]	[4,24]	3	[6,18]	24
18	[2,12]	3	[1,14]	3	[6,22]	3	[10,12]	16
19	[10,14]	20	[2,12]	17	[10,19]	16	[8,18]	21
20	[12,21]	3	[3,8]	13	[5,21]	20	[6,14]	15
21	[2,12]	7	[1,22]	19	[3,22]	21	[4,18]	9
22	20	[9,13]	[1,17]	24	16	19	[6,15]	8
23	10	[12,20]	8	[11,20]	15	[13,19]	3	[17,24]
	10	[]	16	[]	1	[]	19	[]

limit the possible networks generated, and more computational power would enable analyzing all generated networks. Issues acknowledged in the original paper include loss of information when clustering and binarizing. Additionally, boolean network representations oversimplify the complex dynamic relationships between genes, as networks do not exist in discrete states and there is a wide range of expression levels. Nevertheless, boolean networks are a useful first step in simplifying gene regulatory networks in order to discover useful patterns in prohibitively complex data.

References

Shawn Martin, Zhaoduo Zhang, Anthony Martino and Jean-Loup Faulon. (2007). Boolean dynamics of genetic regulatory networks inferred from microarray time series data. *Bioinformatics* (2007) 23 (7): 866-874. doi: 10.1093/bioinformatics/btm021. First published online: January 31, 2007.

Cho, Campbell, et. al., "A Genome-Wide Transcriptional Analysis of the Mitotic Cell Cycle", *Molecular Cell*(1998) 2, 65-73.