1 INTRODUCTION

Gene network data is a collection of interactions among genes. A well-structured gene network map can provide a global view of biological signaling, and illustrate gene clusters. In this paper, we demonstrate how to use SAS Data Step Graphics Interface (DSGI) to graphically represent directed/undirected, weighted/unweighted gene networks. The displaying of the data was inspired by Martin Krzywinski's schemaball (Schema viewer for SQL database, http://mkweb.bcgsc.ca/schemaball/). Gene nodes were aligned along a circle and connections were represented by curves. The program allows modifying gene colors for distinguishing gene functions, line widths for showing gene co-expression levels, and adding arrows for denoting gene regulation directions.

2 METHODS AND RESULTS

2.1 Example of input data

Gene network data can be represented by a symmetric matrix $A = [a_{ij}]$, where $a_{ij}$ can either be the gene expression correlation coefficient between gene $i$ and gene $j$, or a binary number (1 if the gene $i$ and gene $j$ are co-expressed, and 0 otherwise). As an example, a network dataset of 43 genes associated with one disease, is shown in Figure 1 (arbitrary gene IDs and gene symbols were used). 20 genes have known functions (Information="Known") and 23 genes have unknown functions (Information="Unknown").

2.2 Generate two SAS datasets: gene_list and correlation_list

SAS datastep was first applied to generate two SAS datasets according to the preceding gene network data. One dataset "gene_list" contained gene symbols and the known/unknown function information for the 43 genes (Figure 2). The genes were sorted by the column “rank”, which indicated the order how gene nodes would be aligned on a circle (Figure 5). The other dataset "correlation_list", shown in Figure 3, included 70 pairs of genes whose correlation coefficients (column “cor”) were greater than 0.83 (in this study, correlation coefficients greater than 0.83 were considered significant). “rank1” was the rank for “gene1” and “rank2” was the rank for “gene2”. For every pair of genes listed in the “correlation_list” dataset, an arc would be drawn to connect the two gene nodes (Figure 5). The arc height (distance to the circle) was chosen to be half of the distance between two gene nodes, so curves would not overlap and could be easily distinguished.
2.3 SAS IML Procedure and DSGI for graphical output

SAS IML procedure and DSGI were used to create the graphic gene network map. The programming consisted of 3 steps (see the comments inside the codes for details):

Step 1 DSGI to retrieve the size of output graph and “symput” the information into macro variables.

```sas
proc iml;
start calarc(i,j) global(x5,y5,b,starta,enda,dir); *Define a module calarc;
pi = constant('pi'); *The coordinates of the graph center;
h0 = x0; v0 = y0; *The radius of the circle;
r = min(4*h0,4*v0)*0.8; *The number of gene nodes;
*Calculate the start and the end points, radius and center of a curve;
y1 = y0 + r*cos(2*pi/n); *i: rank for gene1, and j: rank for gene2;
y2 = y0 + r*cos(2*pi/n);
x1 = x0 + r*sin(2*pi/n);
x2 = x0 + r*sin(2*pi/n);
x4 = (x1+x2)/2; *(x4,y4) is the middle point between two gene nodes;
y4 = (y1+y2)/2;
run;
```

Figure 2, “gene_list” dataset

<table>
<thead>
<tr>
<th>Information</th>
<th>Genesymbol</th>
<th>Genes</th>
<th>rank</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 Unknown</td>
<td>Symbol_i</td>
<td>g0132</td>
<td>1</td>
</tr>
<tr>
<td>2 Known</td>
<td>g0134</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>3 Known</td>
<td>g0740</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>4 Known</td>
<td>g0748</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>5 Known</td>
<td>g0784</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>6 Known</td>
<td>g0974</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>7 Known</td>
<td>g1225</td>
<td>7</td>
<td></td>
</tr>
<tr>
<td>8 Known</td>
<td>g1529</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>9 Unknown</td>
<td>g1530</td>
<td>9</td>
<td></td>
</tr>
<tr>
<td>10 Known</td>
<td>g1891</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>11 Known</td>
<td>g2083</td>
<td>11</td>
<td></td>
</tr>
<tr>
<td>12 Unknown</td>
<td>g2300</td>
<td>12</td>
<td></td>
</tr>
<tr>
<td>13 Unknown</td>
<td>g2301</td>
<td>13</td>
<td></td>
</tr>
<tr>
<td>14 Unknown</td>
<td>g2311</td>
<td>14</td>
<td></td>
</tr>
</tbody>
</table>

Figure 3, “correlation_list” dataset

<table>
<thead>
<tr>
<th>gene1</th>
<th>gene2</th>
<th>cor</th>
<th>rank1</th>
<th>rank2</th>
</tr>
</thead>
<tbody>
<tr>
<td>g3255</td>
<td>g0132</td>
<td>0.86524</td>
<td>31</td>
<td>1</td>
</tr>
<tr>
<td>g3235</td>
<td>g0132</td>
<td>0.89567</td>
<td>25</td>
<td>1</td>
</tr>
<tr>
<td>g3251</td>
<td>g0132</td>
<td>0.87684</td>
<td>29</td>
<td>1</td>
</tr>
<tr>
<td>g3236</td>
<td>g0132</td>
<td>0.84051</td>
<td>26</td>
<td>1</td>
</tr>
<tr>
<td>g1529</td>
<td>g0132</td>
<td>0.86602</td>
<td>8</td>
<td>1</td>
</tr>
<tr>
<td>g3210</td>
<td>g0132</td>
<td>0.88243</td>
<td>22</td>
<td>1</td>
</tr>
<tr>
<td>g3262</td>
<td>g0740</td>
<td>0.83869</td>
<td>35</td>
<td>3</td>
</tr>
<tr>
<td>g2754</td>
<td>g0748</td>
<td>0.85718</td>
<td>19</td>
<td>4</td>
</tr>
<tr>
<td>g3238</td>
<td>g0374</td>
<td>0.83801</td>
<td>28</td>
<td>6</td>
</tr>
<tr>
<td>g3236</td>
<td>g0974</td>
<td>0.87282</td>
<td>26</td>
<td>6</td>
</tr>
<tr>
<td>g3210</td>
<td>g1529</td>
<td>0.91825</td>
<td>22</td>
<td>8</td>
</tr>
<tr>
<td>g3235</td>
<td>g1529</td>
<td>0.87019</td>
<td>25</td>
<td>8</td>
</tr>
<tr>
<td>g3936</td>
<td>g1529</td>
<td>0.84284</td>
<td>38</td>
<td>8</td>
</tr>
<tr>
<td>g3936</td>
<td>g2366</td>
<td>0.90075</td>
<td>38</td>
<td>15</td>
</tr>
</tbody>
</table>

Step 2 IML procedure to calculate the curves

In the IML procedure, a “calarc” module was first defined. The input parameters are the ranks and the correlation coefficient of a pair of genes, and the output parameters are “x5” and “y5” the coordinates of the curve center; “b” the curve radius; “starta” and “enda” the start and end angles of the curve; and “dir” the direction to draw the curve, from gene1 to gene2 or from gene2 to gene1. The major variables used in “calarc” are illustrated in Figure 4.

The information of gene ranks and correlation coefficients stored in the “correlation_list” dataset was input into a matrix “x”. Records in “x” were input to “calarc” by row, and “calarc” outputs were saved into a matrix “tt” consecutively. Finally, the records in “x” and “tt” were concatenated into a dataset “drawarc0”.

Step 3 IML procedure to draw the graph

```sas
data _null_; rc=ginit(); *Initialize DSGI;
rc=graph('clear'); *Open a graphic segment;
call gask('window', 0, llx, lly, urx, ury, rc); *Get window/image size;
h0=urx/2; *Define the center of the map;
v0=ury/2;
rc=gterm(); *Terminate DSGI;
call symput ("h0",h0); *Symput center values into macro variables;
call symput ("v0",v0);
run;
```
d=sqrt((x1-x2)**2+(y1-y2)**2)/2;
alpha=arsin(d/r);
dd=d-2*sin(alpha/2)**2;
beta=2*atan(dd/d);
b=d/sin(beta);
*xThe radius of the curve;
x5=(x4-h0)/(r*cos(alpha))*(r*cos(alpha)+b*cos(beta))+h0;
y5=(y4-v0)/(r*cos(alpha))*(r*cos(alpha)+b*cos(beta))+v0;
*(x5,y5) is the center of the curve;

*Calculate the start and the angles of the curve;
x1s=x1-x5;
x2s=x2-x5;
y1s=y1-y5;
y2s=y2-y5;
if y1s=0 then y1s=0.001;  *Avoiding invalid tan(0);
a1=atan2(y1s,x1s)+2*pi;
if y2s=0 then y2s=0.001;
a2=atan2(y2s,x2s)+2*pi;
a1=mod(a1,2*pi);
a2=mod(a2,2*pi);

"a1" and "a2" are the start or end angles of the curve;

*Calculate the direction how the curve would be drawn, "starta" is the start angle and "enda" is the end angle;
if (a2-a1)>0 then do;
  starta=a1/pi*180;
  enda=a2/pi*180;
  dir=0;
  "dir=0 means the curve will be drawn from "gene1" to "gene2";
end;
else do;
  starta=a2/pi*180;
  enda=a1/pi*180;
  dir=1;
  "dir=1 means the curve will be drawn from "gene2" to "gene1";
end;
if (enda-starta)>180 then do;
  tmp=starta;
  starta=enda;
  enda=tmp+360;
  dir=mod((dir+1),2);
end;
finish calarc;  *End of the calarc module;

use correlation_list var {"rank1","rank2","cor"};
read all into x;  *Import information from "correlation_list" into "x";
tt=shape(0,nrow(x),6);  *Import information from "correlation_list" into "x";
do k = 1 to nrow(x);
  i=x[k,1];  *i=rank1 for gene1;
  j=x[k,2];  *j=rank2 for gene2;
  run calarc(i,j);  *Run calarc;
  tt[k,1]=x5;  *(x5,y5) the coordinates of the curve center;
  tt[k,2]=y5;
  tt[k,3]=b;  *Curve radius;
  tt[k,4]=starta;  *Curve start angle;
  tt[k,5]=enda;  *Curve end angle;
  tt[k,6]=dir;  *Direction of how to draw the curve;
end;
dout= x || tt;  *Concatenate matrices "x" and "tt";

*Save the information of how to draw curves into "drawarc0" dataset;
create drawarc0 from dout [colname={"i","j","cor","x5","y5","b","starta","enda","dir"}];
append from dout;
quit;
Figure 4. Illustration of major parameters used in the “calarc” module.

Step 3 DSGI to plot genes on a circle and draw curves
The genes were first lined up along a circle according to the “gene_list” dataset. Gene names were rotated to point to the circle center. Available gene symbols in italic font style were aligned under gene names. If the number of gene nodes, n, was even and |rank1-rank2|=n/2, the curve radius would be infinitely large. In this situation, the two gene nodes would be connected by a straight line. Otherwise, curves would be drawn according to the drawarc0 dataset.

data _null_;
rc=ginit();
rc=graph('clear');
h0=&h0;                          *The coordinates of the graph center;
v0=&v0;
pi=constant pi');
tr=2*pi*r/30/&n;                *Define the radius of small rings for gene nodes;
r= min(&h0,&v0)*0.8             *The radius of the circle;
n=43;                           *The number of gene nodes;
gfnt='centx';                    *Define the font for gene names;
ffnt='centxi';                   *Define the font for gene symbols (italic);
*Draw genes along the circle (according to “geneslist” dataset);
do ii=1 to nn;
   set gene_list nobs=nn point=ii;   *nn=The number of rows in “gene_list” dataset;
   x=h0+r*cos(2*pi/nn*ii);
   y=v0+r*sin(2*pi/nn*ii);
   rc=gset('filtype', 'solid');
   rc=gdraw( 'pie', x,y,tr,tr, 360);     *Define text alignment and angle for each gene node;
   if ii>nn/4 and ii<=3*nn/4 then do;
      x0=cos(2*pi/nn *ii-pi/2);
      y0=sin(2*pi/nn *ii-pi/2);
      halign="right";
   end;
   else do;
      x0=-cos(2*pi/nn *ii-pi/2);
      y0=-sin(2*pi/nn *ii-pi/2);
      halign="left";
   end;
   rc=gset('texalign',halign, 'base');
   rc=gset('texup',x0,y0);
   rc=gset('textfont',gfnt);
   rc=gset('texheight',2);
   rc=gdraw( 'text',x,y,genes);
   rc=gset('texalign',halign, 'top');
   rc=gset('textfont',ffnt);
   rc=gdraw( 'text',x,y,genesymbol);
end;
*Draw curves (according to “drawarc0” dataset);
do jj=1 to nnn;
    set drawarc0 nobs=nnn point=jj;   *nnn=The number of rows in “drawarc0” dataset;
    *If |i-j|= n/2 draw an arc to connect two genes;
    if abs(i-j)= n/2 then
    rc=gdraw('arc',x5,y5,b,starta,enda);
    *If |i-j|= n/2 draw a straight line to connect two genes;
    else do;
    x1=h0+r*cos(2*pi/n*i);
    y1=v0+r*sin(2*pi/n*i);
    x2=h0+r*cos(2*pi/n*j);
    y2=v0+r*sin(2*pi/n*j);
    rc=gdraw('line',2,x1,x2,y1,y2);
    end;
*Create the graph and terminate DSGI;
    rc=graph('update');
    rc=gterm();
    stop;
run;

2.4 Options to modify the gene network map
In this section, a few sample codes were provided for modifying the network map according to different visualization schemes.

2.4.1 Represent gene nodes with different colors, curves with different colors and line widths
Adding the following codes in step 3 after retrieving row records from “gene_list” dataset, genes with known functions would be colored red, and genes with unknown functions would be colored green.

*Choose the color for each gene node according to “information”;
select (information);
when ("known") clr=4   *4 red;
when ("unknown") clr=3  *3 green;
end;
rc=gset('texcolor', clr);

Curves can also be colored. For example, they can be colored red to represent the connections between genes with known functions, blue to represent the connections between genes with known and unknown functions, and green to represent the connections between genes with unknown functions. This can be done by first extracting the known/unknown information stored in “gene_list” dataset to merge with “drawarc0” dataset,

proc sql;
    create table drawarc as
        select a.*, b.information as information1, c.information as information2
        from drawarc0 a left join correlation_list b on a.rank1=b.rank
        left join correlation_list c on a.rank2=c.rank;
quit;

then adding the following codes in step 3 before drawing curves.

if information1^=information2 then
    rc=gset('lincolor', 2);       *2=blue;
else if information1="known" then
    rc=gset('lincolor', 4);       *4=red;
else if information1="unknown" then
    rc=gset('lincolor', 3);       *3=green;

In addition, larger correlations can be represented by wider curves.

wid=(cor-.73)*20;
rc=gset('linwidth', wid);

The network map with customized gene colors, curve colors and curve widths is shown in figure 5.
2.4.2 Add arrows to show regulation directions
Suppose it had been discovered that all seven genes co-regulated with gene g1529 (rank=8) were regulated by g1529, and we wanted to display this information by arrows. Arrow markers was first created by the GFONT procedure, with a font name “arrows”, letter “a” to represent “►” and letter “b” to represent “◄”.

Then adding the following codes in step3 after drawing curves, arrows would be drawn to point to genes regulated by g1529.

```c
*For curves drawn starting from gene node g1529, angle "enda" is used to calculate the arrow angle. Arrow sign "◄" is used;
if (i=8 and dir=0) or (j=8 and dir=1) then do;
x6=x5+b*cos(enda/180*pi);
y6=y5+b*sin(enda/180*pi);
rc=gset('texup',cos(enda/180*pi),sin(enda/180*pi));
rc=gset('texalign','left','half');
rc=gdraw('text',x6,y6,"b");
end;
*For curves drawn ending with gene node g1529, angle "starta" is used to calculate the arrow angle. Arrow sign "►" is used;
if (i=8 and dir=1) or (j=8 and dir=0) then do;
x6=x5+b*cos(starta/180*pi);
y6=y5+b*sin(starta/180*pi);
rc=gset('texup',cos(starta/180*pi),sin(starta/180*pi));
rc=gset('texalign','right','half');
rc=gdraw('text',x6,y6,"a");
end;
```

This piece of code could be easily modified to represent arrows not only at the tips of but also along the curves. Figure 6 shows a gene network map with multiple arrows added along curves.

**Figure 5**, Gene network map with colored genes, colored and weighted curves.
3 CONCLUSIONS
We have shown how to use DSGI to automatically generate comprehensive gene network maps. The code can be easily modified to graphically display other kinds of network datasets. Several ways to display different information through controlling different features of texts (i.e., font, size and color) and lines (i.e., color, size) and adding arrows were explained. In addition, SAS DSGI is capable of managing the dimension, position and scale of the graphic output, inserting a previously created graph, or even adding html hyperlinks at specific regions or points.

4 REFERENCES

CONTACT INFORMATION
Wenjie Song
Regeneron Pharmaceuticals, Inc.,
777 Old Saw Mill River Rd.,
Tarrytown, NY 10591
wenjie.song@regeneron.com

Ke Yang
Regeneron Pharmaceuticals, Inc.,
777 Old Saw Mill River Rd.,
Tarrytown, NY 10591
ke.yang@regeneron.com