PharmaSUG2011 - Paper HS08

Visualizing Healthcare Provider Network using SAS® Tools

John Zheng, Columbia, MD

ABSTRACT

Healthcare provider network or patient-provider network is one kind of affiliation networks. Analyzing such networks allows us to gain additional insights on healthcare provider groups that share patients and patients that belong to the same group. This paper develops methods to visualize provider network data in healthcare using the Annotate facility. Link analysis is used to explore the connections between healthcare providers and patients, and hence connections between peer providers. In addition, this paper also takes advantage of the spatial information in the patient-provider data to gain additional insights. In particular, the connections between providers are projected on a Google map using Google Map Generator.

INTRODUCTION

The healthcare provider network, like other membership networks, is a kind of affiliation networks, or networks that consist of two disjoint sets of nodes and all edges must be between the two sets. Prior work has been published on using SAS to visualize and to compute network statistics of classical (unipartite) social networks (Hoyle 2006, Hornibrook 2009, Ellis 2009). Instead of directly applying social network analysis and visualization techniques on the affiliation network, we first convert the affiliation network into a classical network of providers that is defined by patient-sharing. In this paper, special attention is paid to uncovering those unnaturally dense cliques within the network and the hidden connections in patient-provider networks.

When dealing with the large networks such as the patient-provider network, the first visualization challenge comes from laying out the nodes for big data sets such as this. The popular multidimensional scaling (MDS) can produce spatial layout of the nodes based on their “similarity”, but failed to scale well with large number of nodes. To overcome the difficulties in laying out a very large network with MDS, Hoyle uses a variation of MDS(Hoyle 2009) . Alternatively, when spatial information (such as address or zip code) is available, one can also skip the MDS entirely by projecting all nodes on a Google map using Google Map Generator provided by SAS/GRAPH. Besides those two approaches, this paper also discusses a component analysis method, which dissects the original network into many smaller, manageable sub-networks.

THE REPRESENTATION OF SOCIAL NETWORK DATA IN SAS

Social network data can be represented in two formats, the matrix format and the edge-list format. The matrix format stores social network data in an n-by-n square matrix with all nodes listed across the top and down the side. The value of cell(i,j) in the matrix indicates whether the node i and the node j is connected. The edge list, on the other hand, stores a pair of connected nodes along with attributes of this connection in a single row of a table. For example, a connection between provider i and j with 5 shared patients is stored as (i,j,5). The matrix format stores every pair of nodes whether or not they are connected. Whereas the edge list format only stores edges that exist, hence uses much less space when it comes to storing a sparse network, which is often the case in social network datasets.

The patient-provider network can be represented by an m-by-n matrix with patients being the row nodes and providers being the column nodes. Each cell (i,j) records whether the patient i is cared by provider j in the observation period. Table 1 shows an example of a patient-provider network represented by matrix A. In this example, patient 2 is cared by providers 1and 2. Provider 1 cares patients 1 and 2.

A	Provider1	Provider2	Provider3
Patient1	1		
Patient2	1	1	
Patient3			1
Patient4		1	

Table 1. A Matrix Representation of the Patient-provider Network

We can obtain a classical network of providers by multiplying the transpose of the matrix A with A. The resulting new matrix is an n-by-n square matrix of providers, D, with each cell representing the number of patients shared by the row provider and the column provider.

$$\mathbf{D} = \mathbf{A}^T \times \mathbf{A} = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} \times \begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix} = \begin{bmatrix} 2 & 1 & 0 \\ 1 & 2 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Although the matrix algebra can be carried out using PROC IML, PROC IML is not suitable for manipulating large matrixes with thousands of nodes. We recommend storing the affiliation network data in the edge-list format and converting the affiliation network into a classical network using PROC SQL instead.

In the edge-list format, the patient-provider network in Example 1 can be represented as:

PatientID	ProviderID
1	1
2	1
2	2
4	2

Table 2. An Edge-list Representation of the Patient-Provider Network

To construct a classical network of providers, we can join the table with itself using PatientID and then group the results by NPI (used as ProviderID) and count the number rows to get the number of patients shared. The SQL query for this relational algebra operation is as follows.

```
select tbl1.NPI as NPI1, tbl2.NPI as NPI2,
count(*) as PatientsCnt
from table_name as tbl1, table_name tbl2
where tbl1.PatientID = tbl2.PatientID
group by NPI1, NPI2
```

The result of the relational algebra is an edge-list representation of the classical provider network:

ProviderID1	ProvdierID2	Shared Patients #
1	1	2
1	2	1
2	1	1
2	2	2
3	3	1

Table 3. An Edge-list Representation of the Classical Provider Network

PROVIDER NETWORK VISUALIZATION

Once the affiliation network is converted to a classical network, we may use tools SAS offers to display the network structure to suit various visualization needs. Constellation Applet can be used in conjunction with SAS's XML generating macro %DS2CONST to create social network diagrams. Though convenient, the Constellation applet can only handle limited numbers of nodes. Moreover, the automatic layout changes from call to call, preventing meaningful comparison across multiple network diagrams. As a general purpose graph drawing facility, the Annotate Facility provides greatest flexibility for visualizing network data, and can also be used to create social network diagrams. But users of the Annotate facility must provide coordinates for nodes and edges. The Annotate facility is used in this paper to produce the network diagram for its versatility and for its ability to handle large networks.

A main task in visualizing a social network is to lay out the nodes in a meaningful way. In this paper, we choose the popular MDS algorithm as our layout engine. The MDS attempts to place nodes in such a way that "similar" nodes are close to each other on a 2-dimensional map. The MDS algorithm takes as its input a similarity matrix that records the similarity of every pair of nodes in the network. The classical network of providers obtained in the previous section provides a basis for constructing a similarity matrix. Choosing a similarity measure is a key to visualizing the provider network, and we define our similarity measure as follows.

THE SIMILARITY MEASURE

Though we could use the number of shared patients as "similarity" between two providers, it does not consistently measure "similarity." For example, 10 shared patients between two providers have different implications when the two providers care only 10 patients than when they care 100 patients. To better capture the "similarity" between two providers, we calculate overlapping ratio as their similarity measure:

$$\text{Overlapping Ratio} = \frac{2 \times \text{count of patients shared}}{\text{provider 1's patient count} + \text{provider 2's patient count}}$$

By definition, two providers have an overlapping ratio of 0 if they share no patients and an overlapping ratio of 1 if they share all their patients. This overlapping ratio is used as a similarity measure to lay out the network as well as to identify significant connections.

Because the MDS procedure requires a square matrix as its input, we need to convert the edge-list representation of the provider network to a similarity matrix. Here is SAS code for converting the edge-list representation to the similarity matrix.

```
data SimilarityTbl(drop=npi: OverlapRatio);
array D(1:&DimMax.);
do until(last.npi1);
    set DyadTbl;
          by npi1 npi2;
          if first.npi1 then call missing(of D(*));
          D(input(npi2, NpiIdx.)) = OverlapRatio;
          Var = npi1;
          if last.npi1 then output;
end;
run;
```

Finally, the following SAS code use PROC MDS to layout the network of providers based on the similarity matrix.

```
proc mds
data=SimilarityTbl
out=MdsTbl
dimension = 2
level=ratio similar=1 fit=1;
quit;
```

It's straightforward to rescale the coordinates from the MDS to be percent of graphics area and to generate an annotate file for providers. In addition, we can add other graphical elements to enhance the social network visualization. For example, we can use the color of nodes to represent provider specialty, the size of the node to represent the number of patients cared by the provider, and the thickness of the lines to represent the similarity measure. The code for generating the node dataset is shown below.

```
%mk_pt(annobubble, MdsTbl, input(npi,CordX.), input(npi,CordY.), pie
, color = "cx"||put(npi,NpiTaxColor.), size = sqrt(input(npi,NpiBeneCnt.)/1000));
```

and the code for generating the edge dataset

```
%mk_ln(annolines, DyadTbl, input(npi1,CordX.), input(npi1,CordY.)
, input(npi2,CordX.), input(npi2,CordY.), color = 'gray'
, size = OverlapRatio*4, ThreCond = OverlapRatio>=0.1);
```

The node and edge annotate datasets generating macros ***%mk_pt()*** and ***%mk_ln()*** are given in the appendix. To facilitate the generation of annotation datasets, a series of SAS formats are used to look up the coordinates, patient count, and node colors of providers. We obtain the annotation dataset by combining node dataset and the edge dataset, and use PROC GSLIDE to plot the annotation dataset. Figure 1 shows a network with 26 providers. The color schemes for plotted specialty are: Physicians- *Navy*, Hospitals- *Sky Blue*; Other Service Providers- *Powder Blue, and* Nursing Providers- *Turquoise etc.*

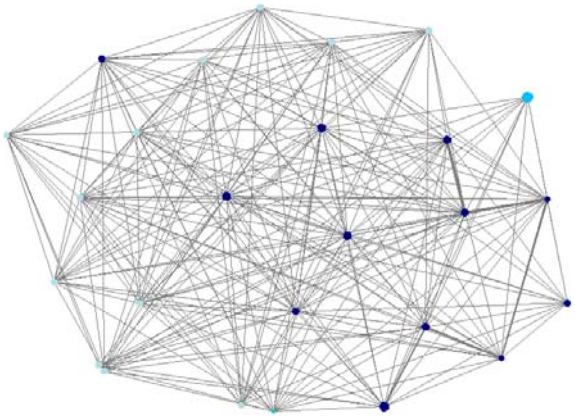


Figure 1. The Visualization of a Network with 26 Providers

STRATEGIES FOR LAYING OUT LARGE NETWORKS

The size of the network poses a serious challenge in applying the MDS layout algorithm. PROC MDS cannot easily reach convergence when dealing with a matrix with more than a few hundred nodes. So it is essential to reduce the size of the network submitted to PROC MDS. We used a few different strategies.

CORE-PERIPHERAL LAYOUT

One strategy, as suggested by Hoyle, is to use MDS to generate coordinates for a small number of "core" nodes. After the core nodes are placed in a center circle by MDS, an additional algorithm places peripheral nodes in an outer band. Figure 2 shows an example of such a core-peripheral layout. In this example, there are 1,100 providers. 190 core providers (chosen by the degree centrality) are laid out in the center, and the remaining 910 peripheral providers are arranged roughly in a circular outer band. Links to core nodes are shown in blue lines, otherwise in gray lines.

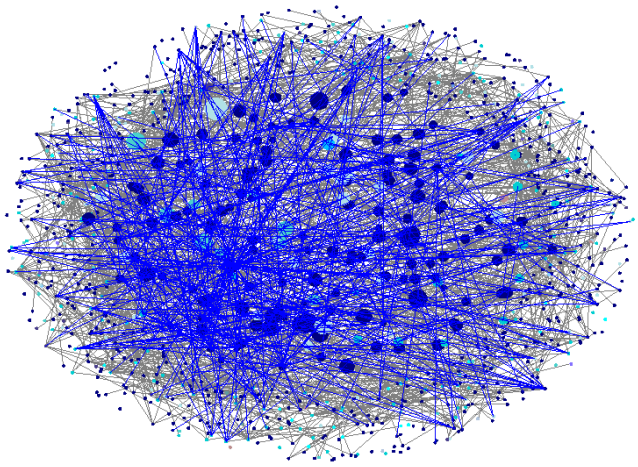


Figure 2. A Core-Peripheral Network of Providers

NETWORK COMPONENT ANALYSIS

However a dense network like Figure 2 can barely tell an insightful story. Our second strategy is to identify components in the original network and draw each component separately. A component is a set of connected nodes that are not connected to any node outside the component. If the original network can be divided into disjoint components, then fewer nodes are entered for each PROC MDS layout, thus alleviating the size problem.

To identify the components in a network, we start with one node, and then add all its neighbors, then neighbors of neighbors, and so on, until no new node is added. This set of connected nodes is labeled as first component. The similar component searching process is repeated for the remaining nodes until all nodes are assigned. The component searching macro %***ComponentSearch()*** can be found in the appendix.

In a well-connected network such as healthcare provider network, the biggest component can easily consist of more than 95% of all nodes. We can further break up the network into smaller clusters, by eliminating weak connections, e.g. we can ignore the connections with less than 1% patients overlapping ratio. With a threshold of minimum 1% patients overlapping ratio, we obtain the following top 5 components.

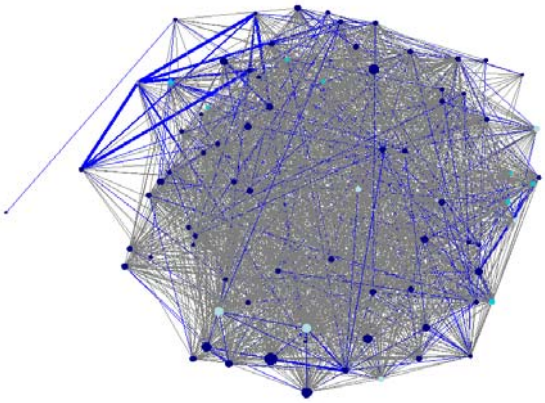
Node Count	Percentage	Accumulative Percentage
111	10.09%	10.09%
82	7.45%	17.55%
50	4.55%	22.09%
40	3.64%	25.73%
34	3.09%	28.82%

Table 4. The Top 5 Components of a Provider Network

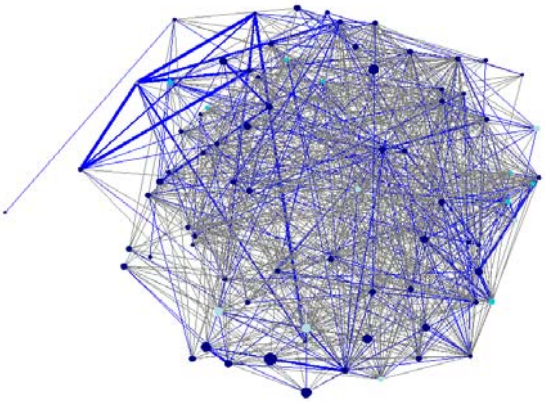
HIGHLIGHTING STRONGER CONNECTIONS

Another useful way to enhance the network visualization is to apply connections threshold. In the following example, the component network with 82 providers was used. From a series of given network diagrams with different

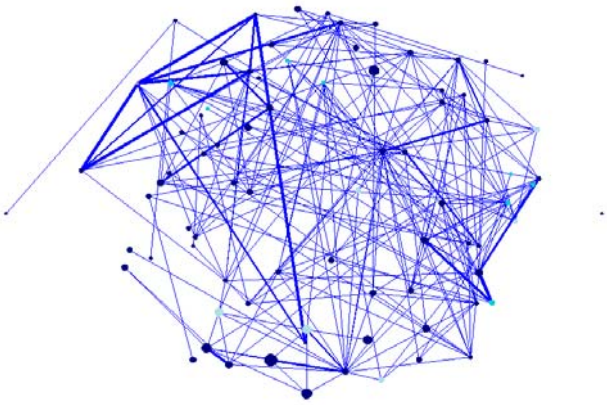
thresholds, we can clearly see that less significant connections are eliminated and important connections emerge. In the last diagram, every connection has at least an overlapping ratio of 10%. The clique of 5 providers at the top-left corner of the diagram was flagged for further study.



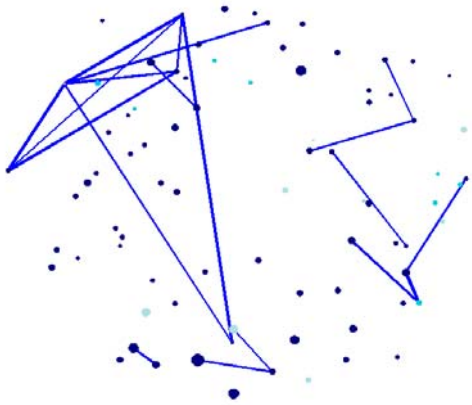
No Minimum Overlapping Ratio



Minimum Overlapping Ratio = 2.5%



Minimum Overlapping Ratio = 5%



Minimum Overlapping Ratio = 10%

Figure 3. An Example of Sub-component Network of Providers with Different Minimum Overlapping Ratio

SOCIAL NETWORK STATISTICS

The social network statistics can be calculated using the edge-list dataset presented in previous sections. In our provider network example, there are a total of 21,724 nodes, 735,488 ties, 134,365,111 triplets (a triplet is a set of three nodes that are connected by at least two ties), and 5,632,309 triangles (a triangle is a triplet connected by three ties). The global clustering coefficient, as measured by 3* number of triangles / number of triplets, is 0.3907. The average local clustering coefficient, measured by the mean of (number of triangles connected to vertex i) / (number of triples centered on i), is 0.6412, which is more dense than the similar-sized SAS-L network presented in Hoyle 2009.

Figures 4 and 5 illustrate the distribution of two social network statistics: the degree centrality and overlapping ratio. Note that the providers' maximum patient overlapping ratio was used in the smoothed histogram of overlapping ratio. The degree centrality reveals how central a provider is in terms of service utilization. According to the degree centrality diagram, radiologists have significantly higher degree centrality than other specialties, which is explained by the frequent use of radiologists by patients cared by other service providers. But having a higher degree centrality does not mean that a provider shares most of its patients with another provider. The latter is captured by the overlapping ratio. From figure 5, radiologists once again have the highest overlapping ratio among the 4 specialty categories. But interestingly, "Other Providers" now have higher patient overlapping ratios than both nursing providers and physicians, while they have overall smallest degree centrality measure.

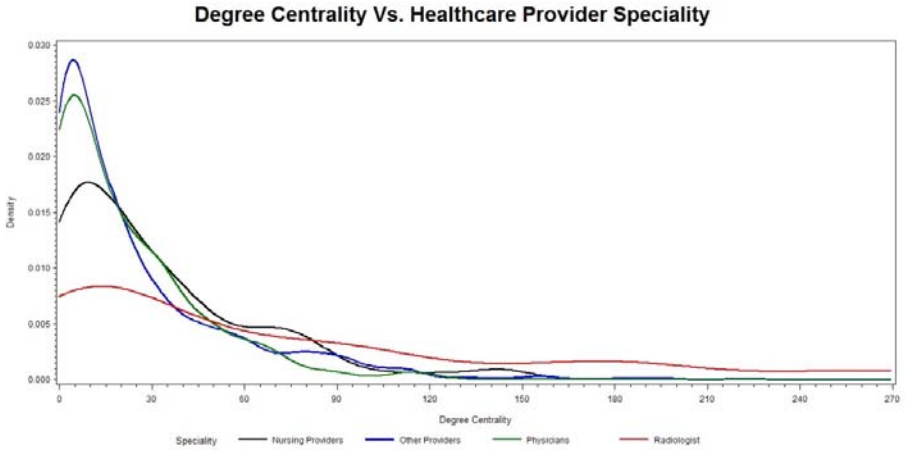


Figure 4. Kernel Density of Degree Centrality by Provider Specialty

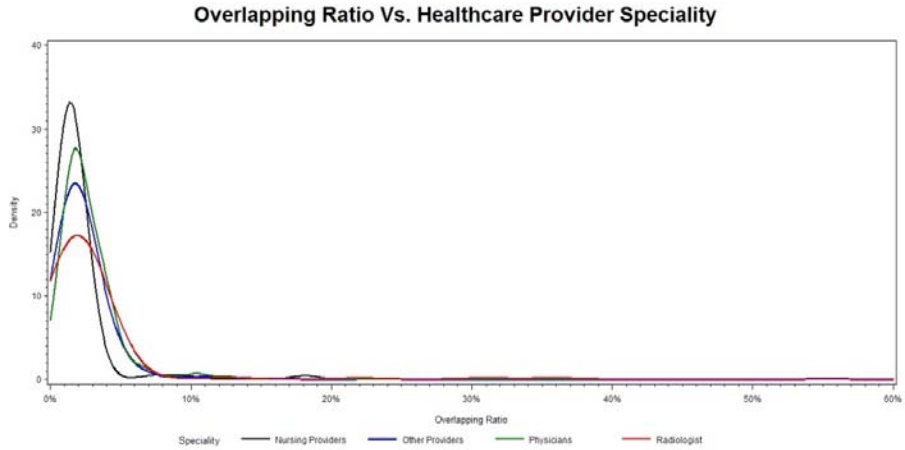


Figure 5. Kernel Density of Overlapping Ratio by Provider Specialty

Figure 6 shows the distribution of overlapping ratios of all provider ties. 6a plots the distribution of overlapping ratios whereas 6b plots the distribution of shared patients count. Comparing the two, we find that the number of shared patients concentrated in a small range between 0 and 30 while overlapping ratios are distributed more evenly. This supports the idea that overlapping ratio is more useful in discerning ties of different degree of similarity.

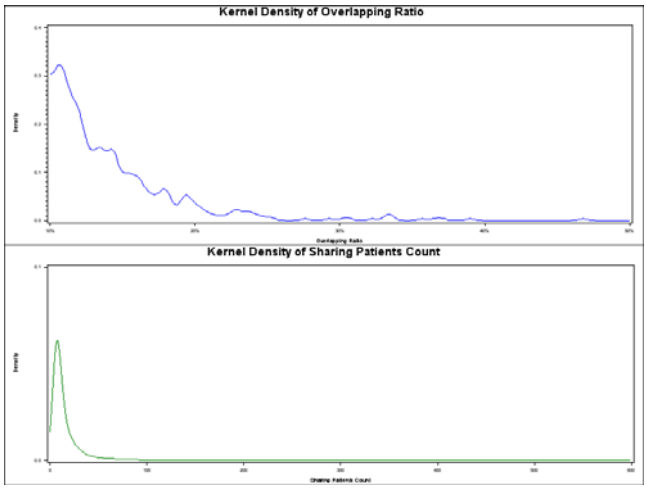


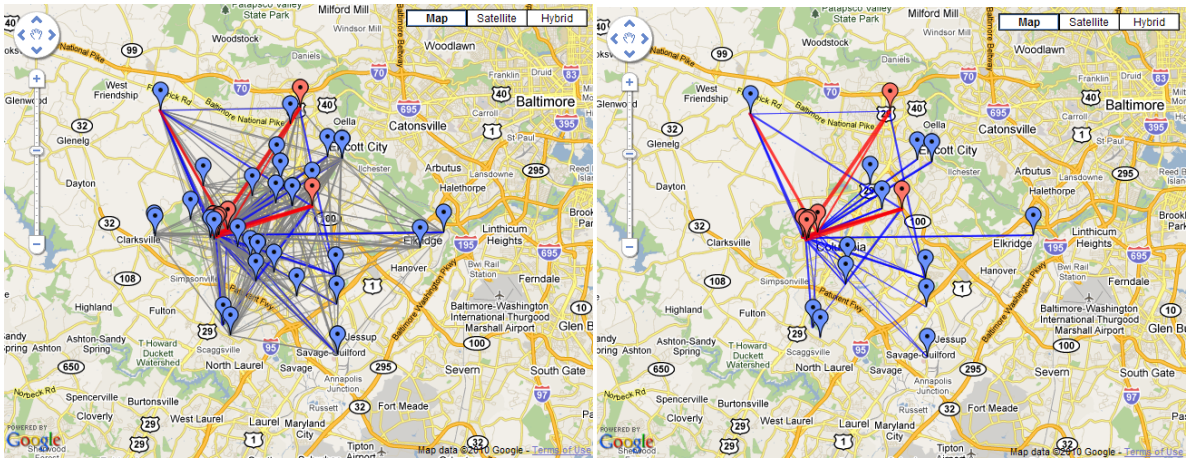
Figure 6. Comparing Kernel Density of Overlapping Ratio and Shared Patients Count

VISUALIZATION USING GOOGLE MAP

In this section, we look closer into each previously indentified component by drawing it on the Google map using Google Map Generator (GMG). We extract the spatial information such as the 9-digit zip code of the practice address from the provider data and add it to the annotate dataset. SAS GMG can convert an Annotate data set into either a KML file or a GPOLY JavaScript file, which is then used to generate the Google map. For details on how to configure SAS GMG, one can refer to Massengill (2010). The macros for generating the node and edge annotate datasets for SAS GSLIDE can be reused for SAS GMG, except that different coordinates (now generated by PROC GEOCODE) and color formats are provided. The code for generating the node and edge annotate dataset is shown as follows.

```
%mk_pt(annogpt, dd_mds, input(npi,NpiZipX.), input(npi,NpiZipY.), point
    , color = put(npi,NpiGoogColor.) , size = 1);
%mk_ln(annoglines, dd_sim, input(npi1,NpiZipX.), input(npi1,NpiZipY.)
    , input(npi2,NpiZipX.), input(npi2,NpiZipY.), color = colors
    , size = 0.5, ThreCond = 1 );
```

The following figure shows an example of a provider network component projected on Google Map. The flagged providers are represented by red icons on the map. Connections with flagged providers having overlapping ratio>=10% are in red lines and less than 10% are in blue lines. Connections between unflagged providers are drawn in gray lines. Once again the line weight is proportional to the overlapping ratio between the pair of providers. Note that the 9-digit zip codes are fictional and randomly selected for illustration purpose.



All Connections Connections with Flagged Providers Only

Figure 7. A Component of the Provider Network Projected on Google Map

CONCLUSION

In this paper, we show how to convert patient-provider network into a classical (unipartite) network for visualization and social network analysis. We construct provider network with connections measured by the percentage of patients shared by two providers (namely overlapping ratio), and then visualize the network with the Annotate facility. To attain manageable network sizes for visualization, we dissect the provider network into several components and draw each component separately. This new approach provides a viable way of visualizing large networks. We further enhance network visualization by applying different thresholds for connections and thus highlighting most significant connections and cliques. We also show how to gain spatial insights by projecting the provider network on a Google map using Google Map Generator provided by SAS/GRAPH.

REFERENCES

• Hornibrook, Shane &Degrees. of Separation: Social Network Analysis Using The SAS System (NESUG2006) http://www.nesug.org/proceedings/nesug06/an/da21.pdf

• Hoyle, Larry Implementing Stack and Queue Data Structures with SAS® Hash Object (SGF 2009) http://support.sas.com/resources/papers/proceedings09/084-2009.pdf

• Hoyle, Larry Visualizing Two Social Networks Across Time with SAS : Collaborators on a Research Grant vs. Those Posting on SAS-L. (SGF2009) http://support.sas.com/resources/papers/proceedings09/229-2009.pdf

• Ellis, Alan R. Using SAS to Calculate Betweenness Centrality http://www.insna.org/PDF/Connections/v29/2009_I-1_P-26-32.pdf

• Massengill, Darrell Google Maps and SAS/GRAPH® (SGF2010)
http://support.sas.com/resources/papers/proceedings10/025-2010.pdf

• Massengill, Darrell Odom, Ed PROC GEOCODE: Now with Street-Level Geocoding (SGF2010)
http://support.sas.com/resources/papers/proceedings10/332-2010.pdf

• Newman, M. E. J. The structure and function of complex networks. SIAM Review 45, 167-256 (2003)

• Brandes, U. A faster algorithm for betweenness centrality. Journal of Mathematical Sociology(2001), 25(2): 163-177.

ACKNOWLEDGMENTS

The author would like to thank Ahmed Al-Attar, Jia Wang for their help in preparing and testing the code, and Gary McQuown for his constructive suggestions.

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:

Name: John Zheng
Phone: (571)-329-1496
E-mail: hqzheng@gmail.com

APPENDIX

```
*nodes annotate dataset generation macro;
%macro mk_pt(annodsn, indsn, x, y, function, color="green", size=0.7 );
data &annodsn(keep=function x y size color xsys ysys hsys when style rotate);
  length function color $8 style $20 rotate 8.;
  retain xsys ysys '2' hsys '3' when 'a' size .7;
  set &indsn;
  function="&function";
  %if "&function"="pie" %then %do;
        rotate=360 ;
  %end;
  style='psolid';
  x=&x;
  y=&y;
  color=&color;
  size=&size;
  output;
run;
%mend;
*edges annotate dataset generation macro;
%macro mk_ln(annodsn, indsn, fromx, fromy,tox,toy, color="blue", size=0.5,ThreCond=1);
data &annodsn(keep=function x y size color xsys ysys hsys when line);
  length function color $ 8 ;
  retain xsys ysys '2' hsys '3' when 'a' size .5 line 1;
  set &indsn;
  /* Move to starting point */
  function='move';
  x=&fromx;
  y=&fromy;
  output;
  /* Draw to the ending point, if threshold condition met */
  x=&tox;
  y=&toy;
  if &ThreCond. then function='draw';
  else function='move';
  line=1;
  size=&size;
  color=&color;
  output;
run;
%mend;
```

```
/* Component Searching Macro */
%macro ComponentSearch(EdgeListDsn,/* Input Edge-List Dataset               */
                       ComponentDsn,  /* Output Component Dataset              */
                                      /*     -with Node Assigned with Component */
                       Node1,         /*    Starting Node Variable Name        */
                       Node2          /*    Ending Node Variable Name          */
                       );
%local ComponentIdx LoopIdx NodeCnt;

%let ComponentIdx=1; /*Initialize Component Index*/

/*Initialize Output Component Dataset*/
proc sql;
create table &ComponentDsn as
select distinct &node1 as Node , . as ComponentIdx  /*Initialize Component Index*/
       from &EdgeListDsn
order by Node;
quit;

/* First Node w.o. Component Assigned*/
data _NodeListFound;
set &ComponentDsn.(where=(missing(ComponentIdx)) obs=1);
run;

/* Main Searching Loop */
%do %until(%nobs(_NodeListFound)=0);
       /*%nobs refer to http://www.datasavantconsulting.com/roland/nobs.sas */

       %let LoopIdx=0;     /* Reset Loop Index for next Compoent */
       %let NodeCnt=0;     /* Reset Node Count */

       /* Searching all Nodes for current Component */
       %do %until(%nobs(_NodeListFound) = &NodeCnt ); /* Loop till No New Node Found*/

                     %let NodeCnt=%nobs(_NodeListFound);

                     data _LstNodeList; set _NodeListFound;run;
                     /* All Nodes Connected with Current Nodes List */
                     proc sql;
                     create table _NodeListFound as
                     select distinct &node2 as Node from &EdgeListDsn
                                  where &Node1 in (select Node from _LstNodeList)
                     order by Node;
                     quit;

                     %let LoopIdx=%eval(&LoopIdx+1); /* Next Loop */
       %end;

       /* Update Component Index for Found Nodes in Output Dataset */
       data &ComponentDsn.;
       merge &ComponentDsn. _NodeListFound(in=inFound);
       by Node;
       if inFound then ComponentIdx=&ComponentIdx.;
       run;

       %let ComponentIdx=%eval(&ComponentIdx+1); /* Next Compoent */

       /* First Node w.o. Component Assigned */
       data _NodeListFound;
       set &ComponentDsn.(where=(missing(ComponentIdx)) obs=1 );
       run;
%end;
%mend;
```