Paper 1065-2017

Using the OPTGRAPH Procedure: Transformation of Transactional Data into Graph for Cluster Analysis

Linh Le, Jennifer Priestley, Kennesaw State University

ABSTRACT

Graph is a mathematic structure capable of representing a network of objects and their relationships. Clustering is an area in graph theory where objects are split into groups based on information of their connections. Depending on the field of the data, object clusters have various meanings, e.g., in the market basket analysis, clusters are families of products that are frequently or likely purchased together, clustering the user log of a website may provide groups of webpages that are often viewed together, etc. This paper provides a SAS Macro that helps transformation of transactional data, or any data with a many-to-many relationship between two entities, into graph data of two types, followed by sample clustering analyses in the graph data using the OPTGRAPH procedure.

INTRODUCTION

Graph structures include a vertex set V that represents the objects of interest, and an edge set E that represents the pairwise relationship among the objects. Clustering techniques in graphs allows the objects to be split into optimal families based on their connection. The resulting clusters may then be used in decision making, e.g. developing a promotion campaign based on a market basket analysis result, or recommender system based on the products each user viewed, etc.

A difficulty to the solution is that often times data sets are not in the best graph format – transformation of data into the desired format may be complicated. As the transactional data format is relatively common in business, this paper provides a SAS macro that supports the transformation process. However, the uses of the macro are not limited to transactional data. Any data set of which each observation represents a pair of $\{entity_1, entity_2\}$ may be transformed into a graph of either entity which then allows graph clustering technique to be applied.

Two types of graphs can be generated from the macro and are introduced in the next section, followed by examples of applying the macro and conducting clustering with the OPTGRAPH procedure.

CORE CONCEPTS AND THE TWO TYPES OF GRAPHS

Mathematically, a graph $G = (V, E)$ (Jungnikel, 2008) consists of a vertex set V and an edge set $E = V^2$. The vertex set represents the objects of interests, and the edge set represents the relationship among them. A graph can be undirected or directed (digraph). Let v_i and v_j be two vertices in G, and $e_{i,j}$ be the edge from v_i to v_j, then in a undirected graph, $e_{i,j} = e_{j,i}$ while in a directed graph the equality may not hold. A graph can be weighted if each edge is assigned a scalar represents any measurement of relationship between the two vertices.

This paper introduces two types of graph: the co-occurrence graph (Raeder, 2011) and the probability graph.

The co-occurrence graph is weighted and undirected. Edges in the co-occurrence graph are based on the frequencies of the object pair occurring in the data. For example, transactional data with schema $\{order, item, customer\}$ may be transformed into an item co-occurrence graph of which each vertex is an item and an edge between two items are the frequency of the item pair being in the same orders. The data can also be transformed into a customer graph where each vertex is a customer and the edges between two customers show how many same items they both purchased. Overall, the weight in the co-occurrence graph is computed as

$$w_{i,j} = Co\_occ(i, j) \quad (1)$$

Figure 1 shows an example of an item co-occurrence graph. The interpretation is that ham and cheese are purchased together 480 times, ham and bread 500 times, and so on.

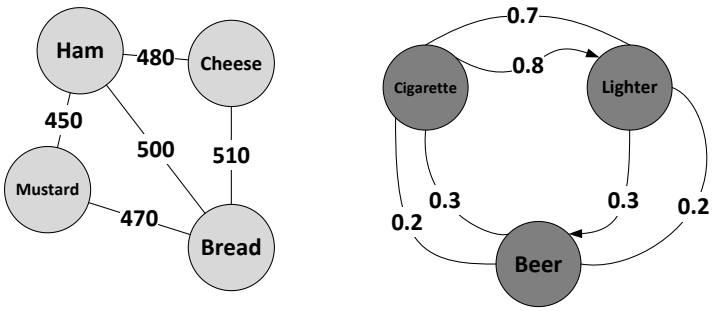


Figure 1. Example of the Co-occurrence Graph and Probability Graph

The second type of graph, the probability graph, is weighted and directed. An edge from object 1 to object 2 shows the observed conditional probability of object 1 leading to object 2. It is derived from the co-occurrence graph and the weight is computed as

$$w_{i,j} = P(j|i) = \frac{Co\_occ(i,j)}{Occ(i)} \quad (2)$$

Return to the previous example, a probability graph derived from the item co-occurrence graph illustrates the cause-effect relationship in each item pair. In details, an edge from item 1 to item 2 shows the observed conditional probability of item 2 being purchased if item 1 is already in the order. The probability graph derived from the customer graph however may not have any cause-effect meaning. An example of the item probability graph is also shown in Figure 1. If cigarette is bought then there is an 80% chance lighter is also purchased, however if lighter is in the order then the chance of cigarette appearing is only 70%. The same interpretation goes with all other edges.

Clustering a graph may be conducted using either connected component algorithms or community detection algorithms. A connected component in a graph is a subset of the graph where there exists a continuous sequence of vertices and edges between any vertex pair. A community is a subset of the graph whose internal vertices are densely connected while connections to other communities are sparser. Figure 2 shows an example of a connected graph, a graph with two connected components and a graph with three communities. The OPTGRAPH procedure in SAS provides both techniques.

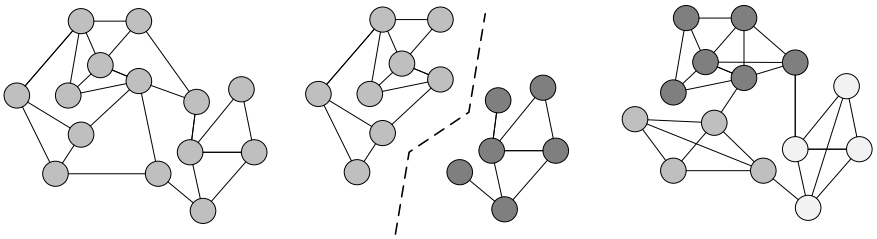


Figure 2. Example of Connected Graph, Connected Components and Graph Communities

THE BUILD_GRAPH MACRO

This paper introduces the BUILD_GRAPH macro:

```
%BUILD_GRAPH(
    data = <data source>,
    link_object = <link object>,
    item = <vertex object>,
    filter = <additional filters on the data>,
    prob_graph = <No/Yes>
    );
```

Parameters used in the macro:

- *data*: the data set to build the graphs

- *link_object*: the link object used to compute the co-occurrence of two vertices. The co-occurrence is the number of times the two vertices have the same values of the link-object in the data. E.g. if an item co-occurrence graph from transactional data is desired, order is the link-object. If a customer graph is desired, link-object is the variable that carries item information.
- *item*: the object of interest to the graph.
- *filter*: any desired filter to apply on the data. Have the same syntax as the SAS WHERE statement
- *prob_graph*: indicates if the probability graph should also be generated. Default value is NO (the yes/no values are not case-sensitive).

Note: the *link_object* and *item* argument can take either character or numeric variables as input.

THE MACRO'S OUTPUT

The macro handles all data transformation and output the co-occurrence graph data format:

$$\{v_1, v_2, weight\}$$

With each observation being an edge in the graph, v_1 and v_2 are the two ends of the edge, and weight is the co-occurrence of the object pair. The output co-occurrence graph data set is named *CO_OCC_GRAPH*.

If the *prob_graph* option is set to *yes*, the probability graph is also generated in the format

$$\{v_1, v_2, co\_occ, v_1\_occ, weight\}$$

Each observation in the data represent a directed edge, and the weight is the conditional probability of v_2 given v_1. The co_occ and v_1_occ attributes are preserved to accommodate further filters if needed. The output probability graph is named *PROB_GRAPH*.

All the temporary data sets generated during transformation are cleared after the macro finishes executing.

SAMPLE ANALYSIS

THE DATA SET

The data set is provided by a major supply chain company which contains approximately 43 million transactional records from September 2014 to January 2015 with about 300 variables from a wholesaler.

Overall, there are over 2.347 million unique orders, 110,000 unique item IDs belonging to 54 categories, and 554 customers from ten different industries such as vendor, retailer, theater, etc. The data normalize the order – item relationship which means each record is a unique pair of order – item. Each transactional record belongs to only one customer, customer class, and time stamp.

A snapshot of the data set with important variables can be seen in Figure 3.

OrderID	ItemID	Customer	Class	Time	Quan.
40614298	75001	J	RETL	25SEP14	1
40608504	42798	S	OCS	24SEP14	1
40614304	34857	C	VEND	25SEP14	7
40614304	37904	C	VEND	25SEP14	4
40614305	97836	C	OCS	25SEP14	15
40607940	79651	D	RETL	24SEP14	1
40607940	96480	D	RETL	24SEP14	1

Figure 3. A Snapshot of the Data

MARKET BASKET ANALYSIS WITH ITEM GRAPH

The co-occurrence graph and probability graph for items can be generated and used in clustering items into families of high frequency or probability of being purchased together. To generate the two graphs simultaneously with the BUILD_GRAPH macro:

```
%BUILD_GRAPH(
    data = sample_transdata,
    link_object= OrderID,
    item = itemID,
    filter = length(attribute2) > 2,
    prob_graph = Yes
    );
```

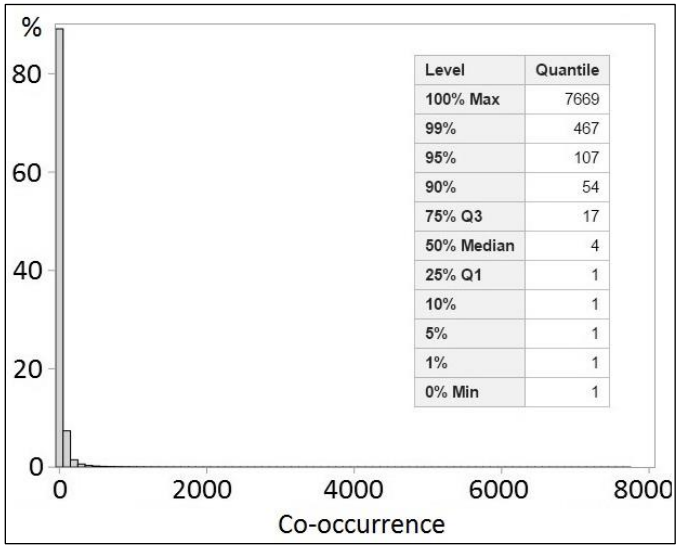
The output co-occurrence graph has 114,848 vertices, 41,989,083 edges, and the probability graph has 84 * 106 edges. A sample of the two data sets can be seen in Figure 4. Since promotion strategy does not often focus on all items available in inventory, a weight threshold can be chosen for the graphs to filter those with low profit.

Obs	v1	v2	weight
1	97	135	3185
2	97	137	2270
3	97	182	3302
4	97	251	1464
5	97	409	2106

Obs	v1	v2	co_occ	v1_occ	weight
1	97	135	3185	5247	0.60701
2	97	137	2270	5247	0.43263
3	97	182	3302	5247	0.62931
4	97	251	1464	5247	0.27902
5	97	409	2106	5247	0.40137

Figure 4. Example of the Outputted Co-occurrence and Probability Graph

The threshold can be chosen by examining the distribution of the co-occurrence which is shown in Figure 5. With the threshold set, the OPTGRAPH procedure can then be applied to cluster the data. Either the CONCOMP statement or COMMUNITY statement can be used, as practical experiments show that as the threshold increases, the number of connected components and communities converges.



Level	Quantile
100% Max	7669
99%	467
95%	107
90%	54
75% Q3	17
50% Median	4
25% Q1	1
10%	1
5%	1
1%	1
0% Min	1

Figure 5. Distribution of Item Pair Co-occurrence

Table 1 shows the filtered co-occurrence graph information by each threshold.

Weight Threshold	Graph Order	Graph Size	Components	Communities
No filter	114,848	$42*10^6$	545	604
467	14560	420512	369	392
1000	6443	155220	259	267
1500	3802	70549	163	168
2000	2299	33733	132	135

Table 1. Filtered Graphs with Different Thresholds

Suppose the co-occurrence threshold is 2000, the OPTGRAPH procedure can be used as in the code snippet below. The output data set for connected components has the clusters labeled as *concomp*, and the data set for communities has the clusters labelled as *community_1*, as shown in Figure 6.

```
proc optgraph
    direction=undirected
    data_links=fullgraph
    out_nodes=comp2000;
    data_links_var
        from = v1
        to = v2
        weight = weight;
    concomp;
    where count >= 2000;

proc optgraph
    direction=undirected
    data_links=fullgraph
    out_nodes=comm;
    data_links_var
        from = v1
        to = v2
        weight = weight;
    community;
    where count >= 2000;
```

Obs	node	concomp
1	97	1
2	135	1
3	137	1
4	182	1

Obs	node	community_1
1	97	0
2	135	0
3	137	0
4	182	0

Figure 6. Sample Output Component and Community Data Set

The probability graph can also be analyzed with the OPTGRAPH procedure, however with a few changes as it is a directed graph. Suppose the probability threshold is 0.5, a co-occurrence threshold should still be set to avoid pairs with high probability but low purchase frequency.

Alternatively, a filter for customer may be applied to focus the analysis on a customer or a customer segment in a specific time period. The code below shows two examples of applying additional filters to the macro. Note that the syntax to the *filter* argument is similar to the *WHERE* statement.

```
%BUILD_GRAPH(
    data = sample_transdata,
    link_object= OrderID,
    item = ItemID,
    filter = Customer="ABC",
    prob_graph = Yes
    );

%BUILD_GRAPH(
    data = sample_transdata,
    link_object= OrderID,
    item = ItemID,
    filter = Class="RETL",
    prob_graph = Yes
    );
```

Finally, since the OPTGRAPH procedure currently does not provide a visualization option, the graph set can be imported and visualized using other packages. Additional information can also be embedded:

- Vertex color represents individual item frequency
- Edge width represents the co-occurrence / probability of the item pair
- Vertex size represents the item's importance

Figure 7 shows an example of the graphs visualized using the mentioned techniques. Other example of visualizing the graphs along with the python code used is provided in the Appendix.

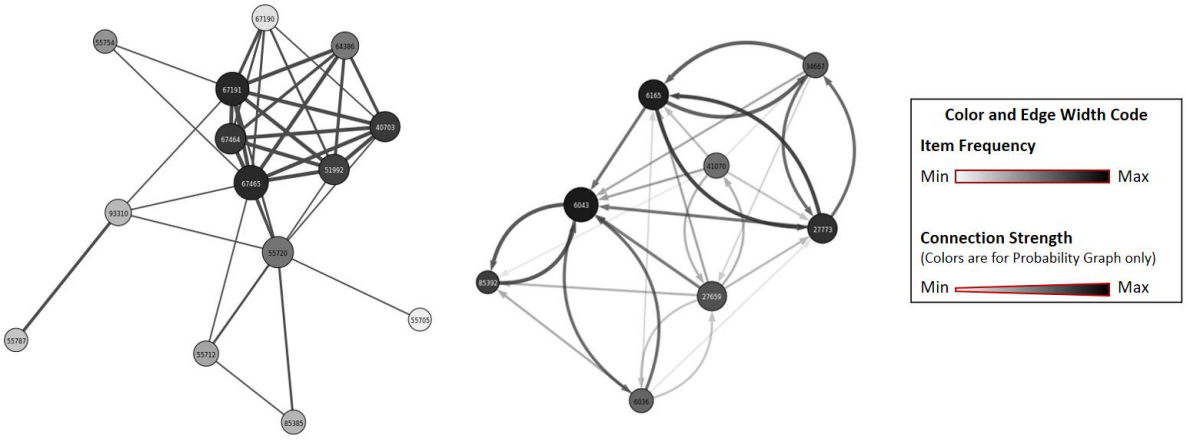


Figure 7. Example of Graph Visualization with Embedded Information

CLUSTERING CUSTOMERS BASED ON PURCHASES OF SIMILAR PRODUCTS

A customer graph can also be generated with their weight being the number of same items each customer pair have purchased. As the probability graph is not meaningful in this case, *prob_graph* is set to NO.

```
%BUILD_GRAPH(
    data = sample_transdata,
    link_object= ItemID,
    item = Customer,
    filter = length(Customer) > 2,
    prob_graph = NO
    );
```

The generated co-occurrence graph has 546 vertices and 55719 edges. Subsequently, a weight threshold can be chosen based on the distribution, and the customer can be clustered. Using the COMMUNITY statement, 546 customers can be clustered into 5 communities as in Figure 8. This information can be used in addition to customer segment to better categorize the customer database.

community_1	Frequency	Percent	Cumulative Frequency	Cumulative Percent
0	77	14.10	77	14.10
1	209	38.28	286	52.38
2	96	17.58	382	69.96
3	149	27.29	531	97.25
4	15	2.75	546	100.00

Figure 8. Customer Communities Generated by the OPTGRAPH Procedure

CONCLUSION

Graph is a powerful tool in modeling and analyzing objects and their pairwise relationships. This paper provides a SAS macro that helps analyst in transforming the popular transactional data format in to graph data for further graph analysis techniques. The input data set is however not limited to transactional format but any data that contains a many-to-many relationship between two entities. The examples provided in this paper show how graphs can be used in the market basket problem, or in categorizing customer. However, the macro and graph analysis can also be used to cluster webpages that are viewed concurrently, items that users browse on an e-commerce website, etc.

REFERENCES

Jungnickel, D. (2008). Graphs, networks and algorithms. Berlin: Springer.

Raeder, T., & Chawla, N. V. (2011). "Market basket analysis with networks." *Social network analysis and mining*, 1(2), 97-113.

CONTACT INFORMATION

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

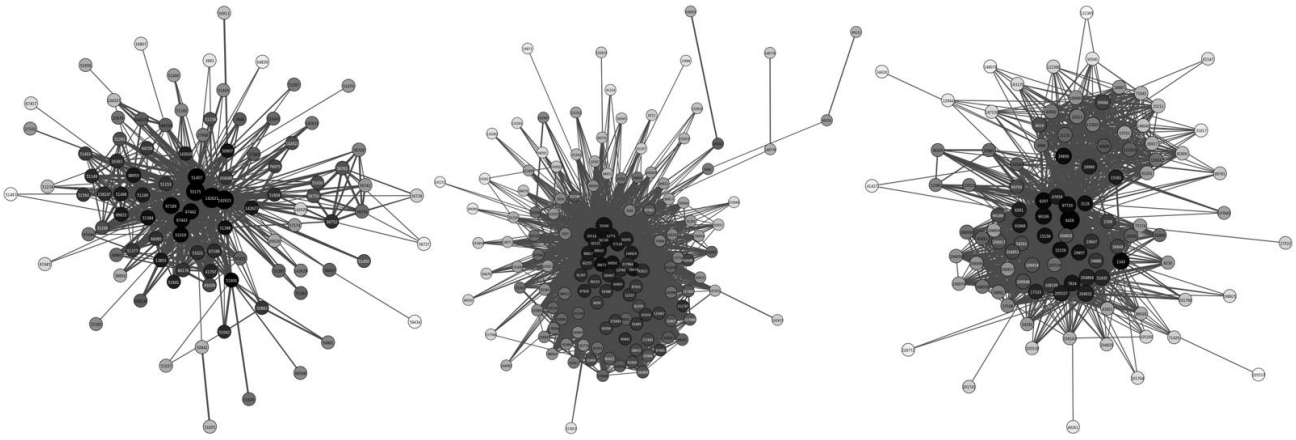
Linh Le
Kennesaw State University
lle12@students.kennesaw.edu

APPENDIX A – BUILD_GRAPH MACRO SAS CODE

```
%MACRO BUILD_GRAPH(data,link_object,item,filter,prob_graph=no);
/*parsing data*/
Data temp (keep = &link_object &item);
      set &data;
      where &filter;

/*making sure all &link_object - &item pairs are unique*/
proc sql;
      create table unique_temp as
      select distinct &link object, &item
      from temp;

/*get individual item count*/
proc sql;
      create table itemcount as
      select &item, count(*) as count
      from &data
      group by &item;
Proc sort data=unique_temp ;
      by &link_object &item;

/*Get wide data*/
Proc Transpose DATA=unique_temp OUT=temp2 prefix=itm;
      var &item;
      by &link_object;
proc sql;
      select nvar - 2
      into :nvar1
      from dictionary.tables
      where libname='WORK' and memname='TEMP2';
quit;

%let nvar = %trim(&nvar1);
%let dsid=%sysfunc(open(TEMP,i));
%if %trim(%sysfunc(vartype(&dsid,2))) = C %then %let sign = 1;
%else %let sign = 0;
%let rc=%sysfunc(close(&dsid));

/*subsetting pairs*/
Data TEMP3 (drop=itm1-itm&&nvar i j);
      set TEMP2;
      %if &sign = 1 %then %do;
            array vars{*} $ itm1-itm&&nvar;
            do i = 1 to &nvar-1;
                  if TRIM(vars[i]) = " " then leave;
                  else do j = i+1 to &nvar;
                        if TRIM(vars[j]) = " " then leave;
                        else do;
                              pair =
CATX("|",PUT(vars[i],8.),PUT(vars[j],8.));
                              item1 = vars[i];
                              item2 = vars[j];
                              output;
                        end;
                  end;
```

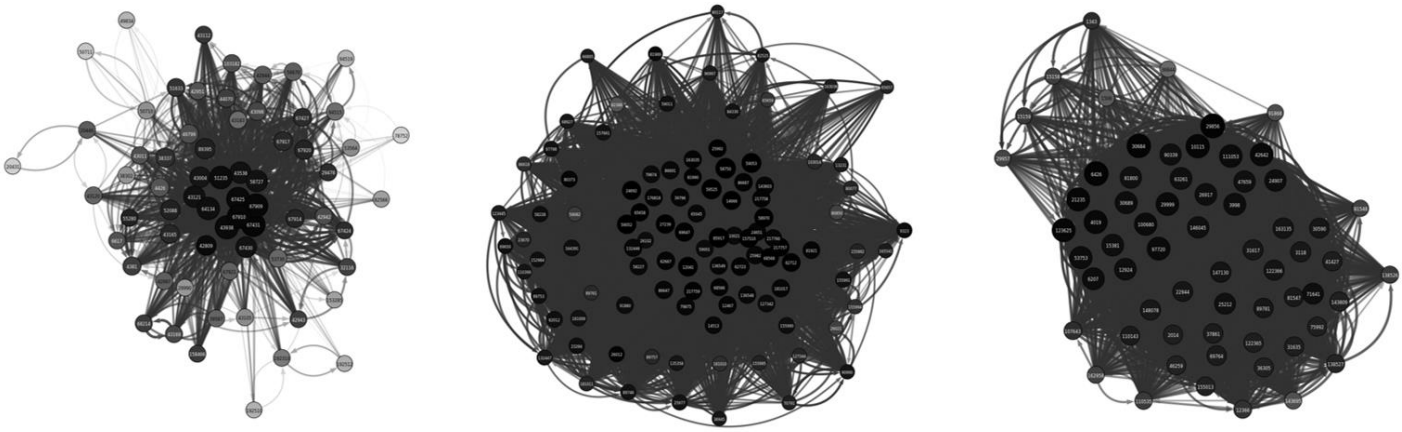
```
            end;
      %end;
      %else %do;
      array vars{*} itm1-itm&&nvar;
            do i = 1 to &nvar-1;
                  if vars[i] = . then leave;
                  else do j = i+1 to &nvar;
                        if vars[j] = . then leave;
                        else do;
                              pair =
CATX("|",PUT(vars[i],8.),PUT(vars[j],8.));
                              item1 = vars[i];
                              item2 = vars[j];
                              output;
                        end;
                  end;
            end;
      %end;
      drop _name_;

/*Create Co-occurrence Graph Set*/
proc sql;
      create table co_occ_graph as
      select item1 as v1, item2 as v2, count(pair) as weight
      from temp3
      group by pair, item1, item2;
quit;

%if %lowcase(&prob_graph) = yes %then %do;
/*Create Probability Graph Set*/
proc sort data=co_occ_graph; by v1 v2;

/*transform to directed graph*/
data _temp_ (drop=temp);
      set co_occ_graph;
      temp = v1;
      v1 = v2;
      v2 = temp;
data di_graph;
      set co_occ_graph _temp_;

proc sort data=di_graph; by v1;
proc sort data=itemcount; by &item;
data prob_graph;
      merge di_graph (in=a rename=(weight=co_occ))
              itemcount (rename =(&item=v1 count=v1_occ));
      by v1;
      weight = co_occ / v1_occ;
      if a;

proc datasets library=work;
      delete di_graph _temp_;
%end;

proc datasets library=work;
      delete temp unique_temp temp2 temp3 itemcount;
%MEND;
```

APPENDIX B – GRAPH VISUALIZATION EXAMPLE



a) Vendor b) Retailer c) Theater

Co-occurrence Components from Three Major Customer Segments in the Data Set



a) Customer B b) Customer D c) Customer A

Probability Components from Three Top Customers in the Data Set

APPENDIX C – PYTHON CODE TO VISUALIZE GRAPHS WITH THE IGRAPH PACKAGE

```
import igraph as g
import numpy as np
from scipy.stats import  percentileofscore as pctl
import math

f = open("/home/graph/input.csv", "r")

f1 = open("/home/graph/input2.csv", "r")

rawV = []
rawE = []
rawW = []
vFreq = []

for line in f:
    if "item1" not in line:
        tmp = line.split(",")
        rawE.append([tmp[0].strip(),tmp[1].strip()])
        rawW.append(float(tmp[2]))

for line in f1:
    if "node" not in line:
        tmp = line.split(",")
        rawV.append(tmp[0].strip())
        vFreq.append(int(tmp[-1].strip()))

itemGraph = g.Graph()

for i in range(len(rawV)):
    itemGraph.add_vertex()

itemGraph.vs["name"] = rawV
itemGraph.vs["Freq"] = vFreq

npFrq = np.array(vFreq)
npW = np.array(rawW)

for v in itemGraph.vs:
    rgbf = 1. - pctl(npFrq, v["Freq"])/100.
    v["color"] = (rgbf, rgbf, rgbf, 1.)
    if rgbf < 0.35:
        v["label_color"] = "white"

itemGraph.add_edges(rawE)
itemGraph.es["weight"] = rawW

for e in itemGraph.es:
    rgbf = pctl(npW,e["weight"])/100.
    e["width"] = 1 + math.ceil(rgbf * 4)
    e["color"] = "rgba(75,75,75,1.)"

components = itemGraph.components().subgraphs()
```

```
add = 0

visual_stype = {}
visual_stype["vertex_label_size"] = 8
if itemGraph.is_directed():
    visual_stype["margin"] = 100
    add = 100

for i in range(len(components)):
    temp = components[i]
    temp.vs["label"] = temp.vs["name"]
    pg = temp.pagerank(directed=False,weights="weight")
    if np.min(pg)==np.max(pg):
        pgs = 2./3.
    else:
        pgs = (pg - np.min(pg))/(np.max(pg)-np.min(pg))
    temp.vs["size"] = 30 + 15*pgs
    visual_stype["layout"] = temp.layout_lgl()
    fname = "/home/graph/G" + str(i) + ".png"
    n = len(temp.vs)
    visual_stype["bbox"] = (500+n*5 + add, 500+n*5 + add)
    if len(temp.vs) > 2:
        g.plot(temp,fname, **visual_stype)
```