Using SAS® to Measure Community Cohesion
Desiree Alderson, MDRC, New York, NY
Lyndsay McDonough, MDRC, New York, NY
Emilio J. Rodriguez, MDRC, New York, NY

ABSTRACT
Can you use SAS® to measure community cohesion? Many social programs aim to create or enhance connections among individuals. For example, the Supporting Healthy Marriage project includes an effort to establish support groups for couples. The available data do not always contain variables that can unambiguously associate individuals (e.g., family name or zip code). However, they sometimes contain information about people’s activities or locations. SAS programs can use people’s frequent presence at the same time and place to identify groups and link individuals. This paper will review PROC SQL and DATA step techniques to detect and quantify non-obvious associations.

INTRODUCTION
Programmers are often presented with data on thousands of individuals and asked to find associations — for example, to identify such groups as families or friends. Many social programs aim to investigate and develop group solidarity, and programmers must measure degrees of program success. Defining these groups proves challenging when the program data available do not contain obvious variables of association.

In the Supporting Healthy Marriage (SHM) project, married couples meet in groups for marriage education sessions over a ten-week period. The hope is that these groups remain stable over time so that couples develop supportive networks with peers. The program’s goal is for groups to advance through the ten-week curriculum together. Couples sometimes need to visit other groups for make-up sessions or to transfer to other groups altogether; as members of each group come and go, the stability of the groups varies.

The first challenge lies in defining a group to which any one couple belongs. Although they begin the curriculum with one group of couples, they will often meet with several groups along the way, the members of which are also fluctuating. Therefore, rather than define a couple’s group by the members with whom they begin the curriculum, a programmer might define a group by the frequency of meetings that a couple has with other couples.

The second challenge is to measure the cohesiveness of the group. The more often couples meet face to face, the higher the weight of their associations with other group members and the higher the sense that they are part of a cohesive group. The degree of cohesiveness of each group is rated by the frequency of meetings attended simultaneously.

This paper has two goals: First, to use people’s presence at the same time and place to identify associations (i.e., probable friends); and second, to measure the cohesiveness of these associations (i.e., friendship rates).

SAMPLE DATA SET
The data set FindFriend is used in this paper:
This data set contains attendance data on 12 people and four marriage education sessions. Each observation represents one person’s attendance at one session. The variable FULLNAME is the name of the individual; the variable SESSION is the date the individual attended a marriage education session. For example, Andy, Alice, Bob, Bee, Cal, Claire, Floyd, and Fay all attended a marriage education session on January 2, 2007.

We assume that the probability that two individuals became friends increases with the number of common sessions they attended. If two people met each other at 10 marriage education sessions, the probability that the two individuals developed a friendship would be higher than if two people met each other at only one marriage education session. We can reasonably assume that two people met in a session because these are small events.

**FINDING FRIENDS WITH PROC SQL**

Step 1: We start by counting the number of marriage education sessions each individual attended. The goal was for every participant to complete four sessions, but not all participants attended all of the sessions. Each observation is one person’s attendance at one session, so if we count the number of times a person’s name appears, we will know how many sessions the person attended.

We can reasonably assume that two people met in a session because these are small events.

The SQL aggregate function COUNT can be used to count the number of times FULLNAME appears in the data set. To calculate the number of sessions each individual attended, we need to count the number of times each unique value of FULLNAME appears in the data set. The GROUP BY clause in PROC SQL specifies that SQL should count for each unique value FULLNAME. Then COUNT is assigned to the variable NUM_OF_SESSIONS:

```
proc sql;
create table numsessions as
select fullname, count(fullname) as num_of_sessions
from findfriend
  group by fullname;
quit;
```
Next, our objective is to count the number of sessions that each participant attended with every other participant. To find participants who attended sessions together we must compare each record with every other record and look for records with common session dates. Participants with the same session date attended class together.

Step 2: With PROC SQL, comparing every record in a table to every other record in the same table requires a reflexive join. This means we join (or merge) the table to itself. By default, PROC SQL reflexive joins create the Cartesian product (i.e., every record in one table is combined with every other record in the table).

The code below creates a table named CommonSessions by joining the FindFriend table to itself. Every record is linked to every other record. A PROC SQL WHERE clause keeps only the records with session dates that match. In the PROC SQL SELECT clause, we identify names from the first copy of the table by the variable MEMBERNAME and the names from the second copy of the table by the variable CONNECTIONNAME. ORDER BY sorts the table by MEMBERNAME and CONNECTIONNAME so that everyone who attended class together is displayed in the first and second columns and the dates that they were in class together is displayed in the third column. It is easy to see each individual’s group of probable friends since the result set displays the name of anyone who ever attended at least one common marriage education session.

```
proc sql;
  create table commonsessions as
  select member.fullname as membername, group.fullname as connectionname,
       group.session as commonsession
  from findfriend as member, findfriend as group
  where group.session=member.session and membername ne connectionname
  order by membername, connectionname, commonsession;
quit;
```

The data set CommonSessions has one observation for each time two people met. Data include who met whom and when they met. Partial output:

<table>
<thead>
<tr>
<th>Obs</th>
<th>FullName</th>
<th>Num_Of_Sessions</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Alice</td>
<td>4</td>
</tr>
<tr>
<td>2</td>
<td>Andy</td>
<td>4</td>
</tr>
<tr>
<td>3</td>
<td>Bee</td>
<td>2</td>
</tr>
<tr>
<td>4</td>
<td>Bob</td>
<td>2</td>
</tr>
<tr>
<td>5</td>
<td>Cal</td>
<td>1</td>
</tr>
<tr>
<td>6</td>
<td>Claire</td>
<td>1</td>
</tr>
<tr>
<td>7</td>
<td>Dave</td>
<td>2</td>
</tr>
<tr>
<td>8</td>
<td>Dawn</td>
<td>2</td>
</tr>
<tr>
<td>9</td>
<td>Emma</td>
<td>1</td>
</tr>
<tr>
<td>10</td>
<td>Eric</td>
<td>2</td>
</tr>
<tr>
<td>11</td>
<td>Fay</td>
<td>3</td>
</tr>
<tr>
<td>12</td>
<td>Floyd</td>
<td>3</td>
</tr>
</tbody>
</table>
Step 3: The more often people meet face to face, the higher the sense that they are part of a cohesive group. Therefore, the next step is to count the number of common sessions each pair of individuals attended. The PROC SQL GROUP BY clause can be used to count each pair of linked individuals:

```sql
proc sql;
  create table numincommon as
  select membername, connectionname,
    count(*) as num_of_common_sessions
  from commonsessions
  group by membername, connectionname
  order by membername, connectionname;
quit;
```

Partial output:

<table>
<thead>
<tr>
<th>Obs</th>
<th>MemberName</th>
<th>ConnectionName</th>
<th>Num_of_Common_Sessions</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Alice Aim</td>
<td>Andy Aim</td>
<td>4</td>
</tr>
<tr>
<td>2</td>
<td>Alice Aim</td>
<td>Bee Bear</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>Alice Aim</td>
<td>Bob Bear</td>
<td>2</td>
</tr>
<tr>
<td>4</td>
<td>Alice Aim</td>
<td>Cal Camp</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>Alice Aim</td>
<td>Claire Camp</td>
<td>1</td>
</tr>
<tr>
<td>6</td>
<td>Alice Aim</td>
<td>Dave Dean</td>
<td>2</td>
</tr>
<tr>
<td>7</td>
<td>Alice Aim</td>
<td>Dawn Dean</td>
<td>2</td>
</tr>
<tr>
<td>8</td>
<td>Alice Aim</td>
<td>Emma East</td>
<td>1</td>
</tr>
<tr>
<td>9</td>
<td>Alice Aim</td>
<td>Eric East</td>
<td>2</td>
</tr>
<tr>
<td>10</td>
<td>Alice Aim</td>
<td>Fay Frost</td>
<td>3</td>
</tr>
<tr>
<td>11</td>
<td>Alice Aim</td>
<td>Floyd Frost</td>
<td>3</td>
</tr>
</tbody>
</table>
The variables NUM_OF_SESSIONS and NUM_OF_COMMON_SESSIONS are the number of sessions attended by each person and the number of common sessions attended by each pair, respectively. These can be used to measure the level of association of the pair’s relationship, as we do in the next step.

**QUANTIFYING FRIENDSHIP WITH PROC SQL**

Step 4: Calculate level of association for each pair of participants. To create an indicator of level of association, it is reasonable to ask what percentage of the time did participant X meet with participant Y. We call this the meet rate. In the next PROC SQL step, we calculate the variable MEET_RATE for each pair of individuals.

PROC SQL’s computational features allow us to divide NUM_OF_COMMON_SESSIONS by NUM_OF_SESSIONS to compute the meet rate. MEET_RATE is the percentage of MEMBERNAME’s sessions that CONNECTIONNAME also attended. For example, for Alice, MEET_RATE tells us that Emma attended 25 percent of the sessions that Alice attended.

```sql
proc sql;
create table meetrates as
    select c.membername, connectionname, num_of_common_sessions, num_of_sessions,
           num_of_common_sessions/num_of_sessions as meet_rate
    format=percent9.2
    from numincommon c, numsessions n
    where c.membername=n.fullname;
quit;
```

Partial output:

<table>
<thead>
<tr>
<th>Obs</th>
<th>MemberName</th>
<th>ConnectionName</th>
<th>Num_Of_Common_Sessions</th>
<th>Num_Of_Sessions</th>
<th>Meet_Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Alice</td>
<td>Aim</td>
<td>4</td>
<td>4</td>
<td>100%</td>
</tr>
<tr>
<td>2</td>
<td>Alice</td>
<td>Bee Bear</td>
<td>2</td>
<td>4</td>
<td>50%</td>
</tr>
<tr>
<td>3</td>
<td>Alice</td>
<td>Bob Bear</td>
<td>2</td>
<td>4</td>
<td>50%</td>
</tr>
<tr>
<td>4</td>
<td>Alice</td>
<td>Cal Camp</td>
<td>1</td>
<td>4</td>
<td>25%</td>
</tr>
<tr>
<td>5</td>
<td>Alice</td>
<td>Claire Camp</td>
<td>1</td>
<td>4</td>
<td>25%</td>
</tr>
<tr>
<td>6</td>
<td>Alice</td>
<td>Dave Dean</td>
<td>2</td>
<td>4</td>
<td>50%</td>
</tr>
<tr>
<td>7</td>
<td>Alice</td>
<td>Dawn Dean</td>
<td>2</td>
<td>4</td>
<td>50%</td>
</tr>
<tr>
<td>8</td>
<td>Alice</td>
<td>East East</td>
<td>1</td>
<td>4</td>
<td>25%</td>
</tr>
<tr>
<td>9</td>
<td>Alice</td>
<td>Eric East</td>
<td>2</td>
<td>4</td>
<td>50%</td>
</tr>
<tr>
<td>10</td>
<td>Alice</td>
<td>Fay Frost</td>
<td>3</td>
<td>4</td>
<td>75%</td>
</tr>
<tr>
<td>11</td>
<td>Alice</td>
<td>Floyd Frost</td>
<td>3</td>
<td>4</td>
<td>75%</td>
</tr>
<tr>
<td>71</td>
<td>Emma</td>
<td>Aim</td>
<td>1</td>
<td>1</td>
<td>100%</td>
</tr>
<tr>
<td>72</td>
<td>Emma</td>
<td>Andy Aim</td>
<td>1</td>
<td>1</td>
<td>100%</td>
</tr>
<tr>
<td>73</td>
<td>Emma</td>
<td>East East</td>
<td>1</td>
<td>1</td>
<td>100%</td>
</tr>
</tbody>
</table>

If we examine the output for both Alice and Emma, we can see that MEET_RATE is not an ideal measure. Emma attended only one session and Alice was also at that session. Alice attended 100% of Emma’s sessions, so Emma’s meet rate with Alice is 100%. This is higher than Alice’s meet rate with Fay. Since Alice met Fay three times, we would like for Alice’s meet rate with Fay to be higher than Emma’s meet rate with Alice. Therefore, calculating percent of time that the pair attended the same session is not a robust enough measure of association in situations where not every one attends the same number of sessions. We need a variable on a different scale.

We weight MEET_RATE by the number of common sessions to create the variable FRIENDSHIP_RATE. FRIENDSHIP_RATE is calculated as the number of common sessions that a pair attended times their MEET_RATE. Again we can use PROC SQL’s computational features to multiply NUM_OF_COMMON_SESSIONS by MEET_RATE.
proc sql;
   create table findfriendrates as
      select c.membername, connectionname, num_of_common_sessions, num_of_sessions,
            num_of_common_sessions/num_of_sessions as meet_rate
      format=percent9.2,
            num_of_common_sessions*(calculated meet_rate) as friendship_rate
      format=4.2
      from numincommon c, numsessions n
      where c.membername=n.fullname;
quit;

Partial output:

<table>
<thead>
<tr>
<th>Obs</th>
<th>MemberName</th>
<th>ConnectionName</th>
<th>Num_Of_Common_Sessions</th>
<th>Num_Of_Sessions</th>
<th>Meet_Rate</th>
<th>Friendship_Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Alice</td>
<td>Andy</td>
<td>4</td>
<td>4</td>
<td>100%</td>
<td>4.00</td>
</tr>
<tr>
<td>2</td>
<td>Alice</td>
<td>Bee Bear</td>
<td>2</td>
<td>4</td>
<td>50%</td>
<td>1.00</td>
</tr>
<tr>
<td>3</td>
<td>Alice</td>
<td>Bob Bear</td>
<td>2</td>
<td>4</td>
<td>50%</td>
<td>1.00</td>
</tr>
<tr>
<td>4</td>
<td>Alice</td>
<td>Cal Camp</td>
<td>1</td>
<td>4</td>
<td>25%</td>
<td>0.25</td>
</tr>
<tr>
<td>5</td>
<td>Alice</td>
<td>Claire Camp</td>
<td>1</td>
<td>4</td>
<td>25%</td>
<td>0.25</td>
</tr>
<tr>
<td>6</td>
<td>Alice</td>
<td>Dave Dean</td>
<td>2</td>
<td>4</td>
<td>50%</td>
<td>1.00</td>
</tr>
<tr>
<td>7</td>
<td>Alice</td>
<td>Dawn Dean</td>
<td>2</td>
<td>4</td>
<td>50%</td>
<td>1.00</td>
</tr>
<tr>
<td>8</td>
<td>Alice</td>
<td>Emma East</td>
<td>1</td>
<td>4</td>
<td>25%</td>
<td>0.25</td>
</tr>
<tr>
<td>9</td>
<td>Alice</td>
<td>Eric East</td>
<td>2</td>
<td>4</td>
<td>50%</td>
<td>1.00</td>
</tr>
<tr>
<td>10</td>
<td>Alice</td>
<td>Fay Frost</td>
<td>3</td>
<td>4</td>
<td>75%</td>
<td>2.25</td>
</tr>
<tr>
<td>11</td>
<td>Alice</td>
<td>Floyd Frost</td>
<td>3</td>
<td>4</td>
<td>75%</td>
<td>2.25</td>
</tr>
</tbody>
</table>

The friendship rate for Alice and Fay is higher than the friendship rate for Emma and Alice, which is exactly what we want. We have a friendship rate for Alice and each person she met; next we can measure cohesion for Alice’s group of friends.

**MEASURING COMMUNITY COHESION WITH THE DATA STEP**

Step 5: Now that we have identified each person’s group of friends and friendship rates, we can use the DATA step to measure group cohesion. We are interested in the number of friends a person has and how often they met. It may also be helpful to know each person’s highest possible friendship rate because one close friend can provide a lot of social support. To calculate these measures, we need the data to be organized as one observation per person. On each member’s record, we save the name of each person they ever attended a session with and the measures of association we created. A DO loop can help us reorganize the data set.

During a DATA step, SAS reads the first record created and manipulates the variables as specified in the DATA step. At the end of the step, the record is automatically output before SAS begins the next record. However, enclosing the set statement within a DO loop changes this default behavior. Enclosing the set statement within a DO loop causes SAS to read multiple records together. SAS continues reading records until the DO loop’s condition is met; then SAS outputs the data from multiple records onto one line.

The DO loop below arranges the data so that each person’s group of friends is one observation. The DO loop iterates until it reads the last record for MEMBERNAME. Then the DATA step will output the variables created by the arrays.
data cohesion;

array friend (12) $20;
array numcomm (12);
array contact (12);
array frrate (12);

do f=1 to 12 until(last.membername);
set findfriendrates ;
by MemberName;
friend(f)=ConnectionName;
umcomm(f)=Num_Of_Common_Sessions;
contact(f)=numcomm(f)>0;
frrate(f)=FriendShip_Rate;
end;

Partial output:

Looking again at the observation for Alice, the variables FRIEND1-FRIEND11 tell us the names of every person Alice met. The variables NUMCOMM1-NUMCOMM11 tell us the number of times Alice met each person; FRIEND1 corresponds with NUMCOMM1. For example, since FRIEND3 equals BOB BEAR and NUMCOMM3 equals 2, we know that Alice met Bob twice. CONTACT1-CONTACT11 are dummy variables indicating whether two people met. Since Alice met everyone in our dataset, CONTACT1-CONTACT11 will all equal one for Alice. FRRATE1-FRRATE11 are the friendship rates for FRIEND1-FRIEND11. For example, Claire is Alice's FRIEND5. We saw above in step 4 that Claire and Alice have a friendship rate of 0.25. Therefore, Alice’s FRRATE5 equals 0.25.

Step 6: Use the three sets of numeric variables just created (NUMCOMM1-NUMCOMM11, CONTACT1-CONTACT11, and FRRATE1-FRRATE11), SUM, MEAN, and MAX to calculate summary measures of cohesion for each person’s group of friends.

Total_Contacts = sum(of numcomm1-numcomm12);
Total_People_Contact = sum(of contact1-contact12);
Avg_Num_Contacts = mean(of numcomm1-numcomm12);
Avg_FrNd_Rate = mean(of frrate1-frrate12);
Max_FrNd_Rate = max(of frrate1-frrate12);

label Total_Contacts = "Total number of contacts"
Total_People_Contact = "Number of people with whom had any contacts"
Avg_Num_Contacts = "Average number of contacts with people met"
Avg_FrNd_Rate = "Average friendship rate"
Max_FrNd_Rate = "Maximum friendship rate";
SUM, MEAN, and MAX created five measures to characterize an individual's experience in the marriage education program.

1. TOTAL_CONTACTS is the number of times an individual met anyone. This means that TOTAL_CONTACTS will be 10 if Andy met one person 10 times or if Andy met 10 people one time.
2. TOTAL_PEOPLE_CONTACT is the number of people an individual met, regardless of how many times they met.
3. AVG_NUM_CONTACTS is the average number of contacts with all the people that an individual met. This is TOTAL_CONTACTS/TOTAL_PEOPLE_CONTACT.
4. AVG_FRND_RATE is an individual's average friendship rate.
5. MAX_FRND_RATE is an individual's maximum friendship rate.

INTERPRETING MEASURES OF COHESION

There are several different types of individual and group behavior that can change these five measures of cohesion. From an individual's point of view, there are four likely scenarios. For example, 1) Andy and Alice attend their assigned 10 sessions and meet the same 10 couples every session; 2) Andy and Alice attend their assigned 10 sessions, meet the same 10 couples every session, and meet others who join their session once; 3) Andy and Alice attend their assigned 10 sessions, and the other assigned couples attend occasionally; 4) Andy and Alice attend some of their assigned sessions and some make-up sessions. Scenarios 1) and 2) correspond with high average friendship rates and scenarios 3) and 4) correspond with low average friendship rates.

The factors relevant to group cohesion include the number of sessions the individual attended, the individual's consistency in attending her assigned sessions, group stability, and group size. TOTAL_CONTACTS and TOTAL_PEOPLE_CONTACT depend on number of sessions attended and the number of attendees. If Alice attends many sessions with the same people, her TOTAL_CONTACTS will be high and her TOTAL_PEOPLE_CONTACT will be low. This would correspond with a high average friendship rate. The average friendship rate, AVG_FRND_RATE, depends on the number of sessions people attended together.

In our data set, Alice and Andy attended the highest number of sessions and have the highest number of contacts. They attended 4 sessions together so they also have the highest maximum friendship rate. But the composition of the group Alice and Andy met changed each time. Alice and Andy met Fay and Floyd twice; they met Bee, Bob, Dave, Dawn, and Eric twice; and they met Cal, Claire, and Emma once. Since their group was not stable, Alice and Andy do not have the highest average friendship rate. Therefore, we can use all five measures to characterize an individual's experience and the cohesion of her community.

CONCLUSION

PROC SQL's ability to create a Cartesian product is a valuable tool. You can use PROC SQL and the WHERE clause to connect individuals who have something in common. We used this method to find associations and measure the cohesiveness of individuals’ connections. Our techniques can be used to identify other connections among people who have something in common. For example, these techniques could be used to identify college students who take the same classes or students who are members of the same club.
REFERENCES

ACKNOWLEDGEMENTS

SAS® and all other SAS Institute, Inc., product or service names are registered trademarks or trademarks of SAS Institute, Inc., in the USA and other countries. ® indicates USA Registration.

The authors appreciate the help of Electra Small, Paulette Staum, and Christopher Bost in writing this paper.

CONTACT INFORMATION
Desiree Alderson
Research Analyst
MDRC
16 East 34th Street, 19th Floor
New York, NY 10016
(212) 340-8863
desiree.alderson@mdrc.org

Lyndsay McDonough
Research Analyst
MDRC
16 East 34th Street, 19th Floor
New York, NY 10016
(212) 340-8852
lyndsay.mcdonough@mdrc.org

Emilio Rodriguez
Research Assistant
MDRC
16 East 34th Street, 19th Floor
New York, NY 10016
(212) 340-8899
emilio.rodriguez@mdrc.org