Paper SAS381-2017

Circular Metadata-Group Membership Can Make You Dizzy!

Greg Lehner and Karen Hinkson, SAS Institute Inc.

ABSTRACT

Today it is vital for an organization to manage, distribute, and secure content for its employees. In most cases, different groups of employees are interested in different content, and some content should not be available to everyone. It is the SAS administrator's job to design a metadata-group structure that makes managing content easier. SAS® enables you to create any metadata-group organizational structure imaginable, and it is common to define a metadata-group structure that mimics the organization's hierarchy. Circular group memberships are frequently the cause of unexpected issues with SAS web applications. A circular group relationship can be as simple as having two groups that are members of one another. You might not be aware that you have defined this type of recursive association between groups. This paper identifies some problems that are caused by recursive group memberships and provides tools to investigate your metadata-group structure that help identify recursive metadata-group relationships. The paper explains the process of extracting group associations from the SAS® Metadata Server and how to organize this data to investigate group relationships. The discussion uses two methods to visualize circular group structures. SAS® Visual Analytics is used to generate a network diagram that provides a graphical representation of an organization's group-relationship structure, and a stored process is used to generate a report.

INTRODUCTION

In today's business world, it is vital for organizations to manage, distribute, and secure content for their employees. SAS offers a powerful package of tools (Base SAS® software along with SAS® Management Console) that you can use to manage and secure content for your users. Typically, you associate multiple users that should be able to access the same content into groups. As stated in the *SAS® 9.4 Intelligence Platform: Security Administration Guide, Third Edition*, "Groups are primarily used in access controls, because it is more efficient to assign permissions to groups than to individual users. You can create a nested group structure by making one group a member of another group." (SAS Institute Inc. 2016)

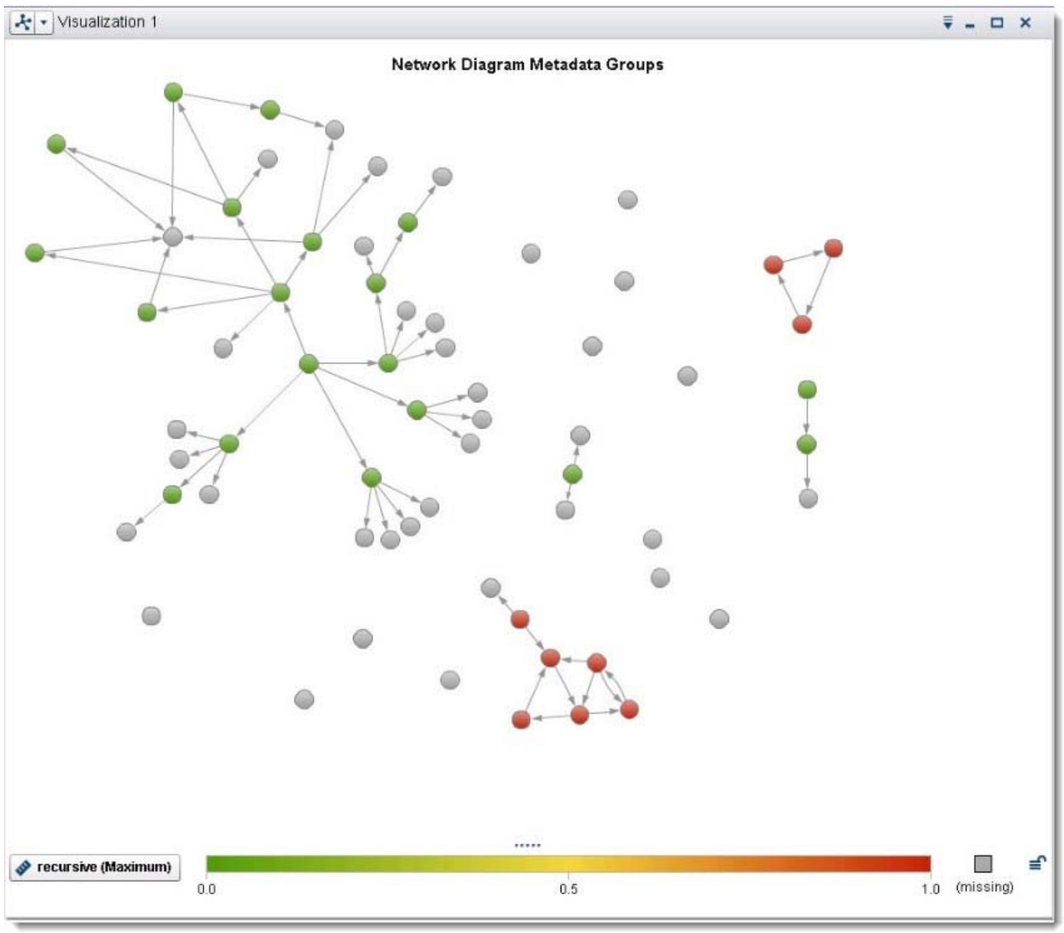
Circular group memberships are commonly the cause of unexpected issues with SAS® web applications. A *circular group membership* can be as simple as two groups that are members of one another. SAS software enables you to create any metadata-group organizational structure that you imagine. However, you might not be aware that you have defined a recursive association between two groups. According to the *SAS® 9.4 Intelligence Platform: Security Administration Guide, Third Edition*, you should "Avoid creating circular or reciprocal group memberships (for example, GroupA is in GroupB, GroupB is in GroupC, and GroupC is in GroupA). Such relationships introduce unnecessary complexity and can negatively impact permissions and performance. When you define nested group memberships (interactively or through importing identity information), the best practice is to use as simple and non-redundant a structure as possible." (SAS Institute Inc. 2016)

When circular groups are defined, specific unusual behavior can occur in SAS® BI Dashboard. In that situation, members of the recursive groups cannot save SAS BI Dashboard content. The cause of this behavior is difficult to identify because recursive groups are not expected, and when you look at the permissions for groups of which users are direct members, the indication is that permissions are correct for the actions that users attempted to perform. Identifying recursive metadata groups can be very difficult because SAS Management Console does not have a facility that can assist with such investigation. SAS Technical Support has also received information about other performance problems when recursive groups are defined. These performance problems occur because the software loops when user permissions are requested. Errors are not generated in this situation, but system performance is affected.

This paper discusses tools that you can use to investigate your metadata-group membership structure. First, the paper explains the process for extracting group associations from the SAS® Metadata Server and how to manipulate this data to make investigating the relationships between groups easier. You can use the resulting SAS table, GROUPLINE, to generate a report in SAS® Enterprise

Guide® and Base SAS® software or through a stored process. Appendix B in this paper illustrates how you can use an OLAP cube to display circular group relationships in a web report.

The paper also shows how to use SAS® Visual Analytics to generate a network diagram (as shown in the following example output) that provides a graphical representation of an organization's group-relationship structure, which enables you to easily identify circular group structures.



Output 1. SAS Visual Analytics Explorer – Network Diagram

PROCESSING GROUP RELATIONSHIPS

This section explains how you can extract group metadata from the SAS Metadata Server using and process group lineage to look for circular relationships. You can use Base SAS software and the %MDUEXTR macro to extract the metadata and process the group relationships.

The following code example takes the output from the %MDUEXTR macro and processes the data to create the GROUPLINE table, which contains group lineage information and identifies groups with circular relationships.

Note: The %MDUEXTR macro is available when you license SAS software that includes the SAS® Metadata Server. The macro is documented in "Appendix 2: User Import Macros" of the *SAS® Intelligence Platform: Security Administration Guide, Third Edition*. (SAS Institute Inc. 2016)

The following code is available for downloading at **`ftp.sas.com/techsup/download/SGF2017/CircularMetadaGroups/SAS381-2017_SAS_Code.zip`**.

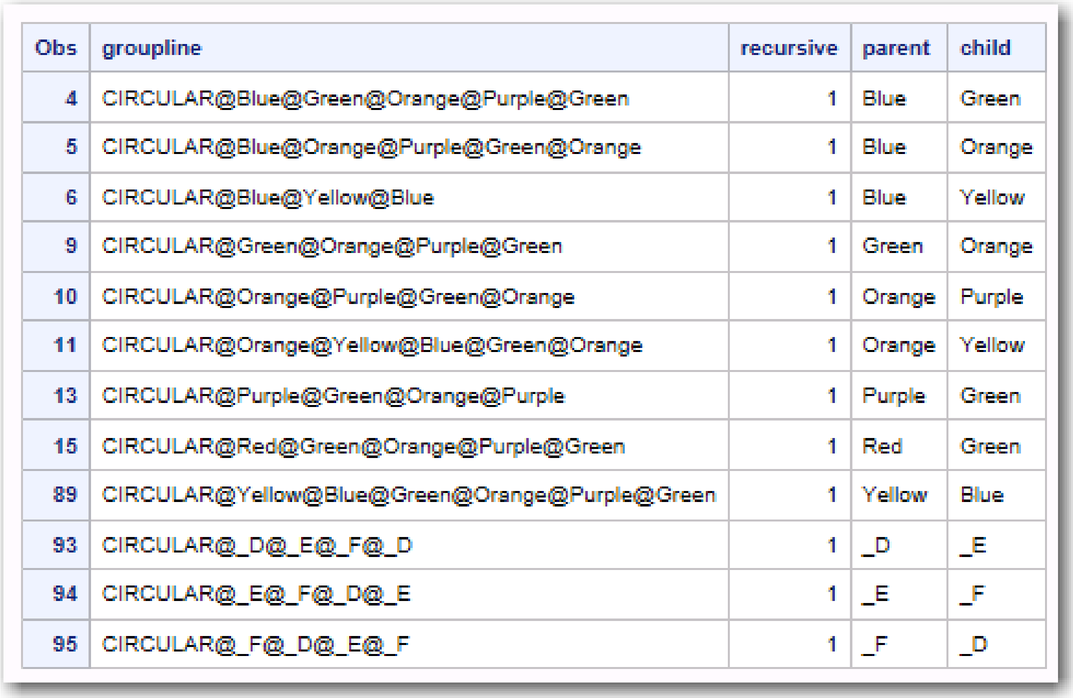
```
%mduextr(libref=work)

   /* Create a format containing IDs for all roles. */
data work.fmtRole (keep=fmtname type start label HLO);
   set work.idgrps(where=(grptype='ROLE')) end=eof;
   fmtname='fmtRole';
   type='C';
   start=keyid;
   label='R';
   output;


   if eof then
   do;
     start='Other';
     HLO='O';
     label='G';
     output;
   end;
run;

proc format cntlin=work.fmtRole;
run;

   /* Keep only groups that are members of groups -- no roles */
data work.memgroups;
   set groupmemgroups_info;
   if put(id,$fmtRole.)='R' then delete;
   if put(memid, $fmtRole.)='R' then delete;
run;

   /* Get unique listing for all groups that are members of */
   /* groups or that have members that are groups.          */
proc sort data=work.memgroups (keep=id) out=work.memgroupids nodupkey;
   by id;
run;

   /* Get groups that are not members of other groups */
data work.justgroups (drop=groupType);
   merge work.group_info (keep=id name groupType where=(groupType=''))
         work.memgroupids (in=mem);
   by id;
   if not mem;
run;

   /* Create full listing of groups. */
data work.allgroups;
   set work.memgroups work.justgroups;
   parent=name;
   child=memname;
run;
```

```
   /* WORK.ALLGROUPS is the table that contains all parent/child        */
   /* group relationship information. This table can be used in SAS     */
   /* Visual Analytics Data Explorer to create a network diagram.       */
   /* However, processing the data further enables you to apply color   */
   /* to the network diagram and then to use this code to generate      */
   /* reports.                                                          */
proc sort data=work.allgroups;
   by parent child;
run;

   /* The transpose processes create a table with one observation that */
   /* has all of the parent/child groups identified as p1-pN and c1-cN.*/
   /* This process enables you to process the data in order to         */
   /* identify circular metadata relationships.                        */
proc transpose data=work.allgroups out=work.parent prefix=p
               name=source label=label;
   var parent;
run;
proc transpose data=work.allgroups out=work.child prefix=c name=source
               label=label;
   var child;
run;

proc sql noprint;
   select count(parent) into :numobs from work.allgroups;
run;
quit;

    /* Table WORK.P_C is a one-observation table that holds one column  */
    /* for each parent (p1-pN) and one column for each child (c1-cN).   */
    /* This is done to enable easier recursive processing of the data   */
    /* to identify circular relationships.                              */
data work.p_c (drop=source);
   merge parent child;
run;

   /* Assign a LIBNAME statement that points to a permanent location so */
   /* that you can easily reference the data in SAS Visual Analytics or */
   /* for your own analysis. For a stored process or SAS Enterprise     */
   /* Guide, you can use the Work library.                             */
libname groups 'c:\temp';

  /* This DATA step creates a variable, GROUPLINE, that shows the      */
  /* group hierarchy of each parent/child group combination. In        */
  /* addition, when a circular relationship is identified, the result  */
  /* of GROUPLINE is identified as CIRCULAR, and it shows the          */
  /* circular relationship.                                            */
data groups.groupline (keep=parent child groupline recursive);
   attrib groupline recursive_groups length=$2000;
   set work.p_c;
   array c(&numobs);
   array p(&numobs);
```

```
do i=1 to dim(p); /* Loop through each column to build GROUPLINE. */
   recursive=0;
   groupline='@'||p(i);
   check_c=c(i);
   if c(i)='' then
   do; /* If child is missing */
      recursive=.;
      parent=p(i);
      child=c(i);
      output;
   end;  /* If child is missing */
   else
   do j=1 to dim(p);
         /* Loop through each column and identify  */
         /* the parent/child relationships.        */
      if check_c=p(j) then
      do; /* If the check field matches the parent column */

            /* If CHECK_C is not missing, then it might be part */
            /* of a circular group structure. Determine whether */
            /* the current CHECK_C is already in GROUPLINE,     */
            /* which indicates a circular relationship.         */
         if check_c ^='' then
         do; /* If the CHECK_C field is not missing */
            x=findw(strip(groupline),strip(check_c),'@','E');
            if x>0 then
               do; /* Circular group */
                  recursive_groups=strip(recursive_groups)||strip(groupl
                  ine);
                  if
                  findw(strip(recursive_groups),strip(check_c),'@','E')>
                  0 then recursive=1;
                  else recursive=0;
                  groupline='CIRCULAR'||strip(groupline)||'@'||strip(che
                  ck_c);
                  parent=p(i);
                  child=c(i);
                  output;
                  j=9999;
               end; /* Circular group */
               else
               do; /* Not a circular group */
                  groupline=strip(groupline)||'@'||strip(check_c);
                  check_c=c(j);
                  if check_c='' then
                  do;
                     parent=p(i);
                     child=c(i);
                     output;
                     j=9999;
                  end;
                  else j=0;
               end; /* Not a circular group */
            end; /* If check field matches parent column */
```

```
              else
              do; /* Check field does not match */
                  /* the parent column         */
                 groupline=strip(groupline)||'@'||strip(check_c);
                 parent=p(i);
                 child=c(i);
                 output;
                 j=9999;
              end; /* Check field does not match */
                   /* the parent column         */
           end; /* If check field matches the parent value */
        end; /* Loop through each column to identify the */
             /*  parent/child relationships.          */
   end; /* Loop through each column to build GROUPLINE.*/
run;

   /* Remove the comment delimiters from the following PROC PRINT       */
   /* code to produce a report showing the circular group relationships.*/
   /* All of the code in this example can be run with SAS Enterprise    */
   /* Guide and Base SAS software, or it can be run as a stored         */
   /* process. You might want to specify a different LIBNAME statement  */
   /* for the GROUPLINE table.                                          */
/*
proc print data=groups.groupline (where=(recursive=1));
run;
*/

/* END OF CODE */
```

The following table shows of a subset of the GROUPLINE table that is created by the SAS code above.



Obs	groupline	recursive	parent	child
4	CIRCULAR@Blue@Green@Orange@Purple@Green	1	Blue	Green
5	CIRCULAR@Blue@Orange@Purple@Green@Orange	1	Blue	Orange
6	CIRCULAR@Blue@Yellow@Blue	1	Blue	Yellow
9	CIRCULAR@Green@Orange@Purple@Green	1	Green	Orange
10	CIRCULAR@Orange@Purple@Green@Orange	1	Orange	Purple
11	CIRCULAR@Orange@Yellow@Blue@Green@Orange	1	Orange	Yellow
13	CIRCULAR@Purple@Green@Orange@Purple	1	Purple	Green
15	CIRCULAR@Red@Green@Orange@Purple@Green	1	Red	Green
89	CIRCULAR@Yellow@Blue@Green@Orange@Purple@Green	1	Yellow	Blue
93	CIRCULAR@_D@_E@_F@_D	1	_D	_E
94	CIRCULAR@_E@_F@_D@_E	1	_E	_F
95	CIRCULAR@_F@_D@_E@_F	1	_F	_D

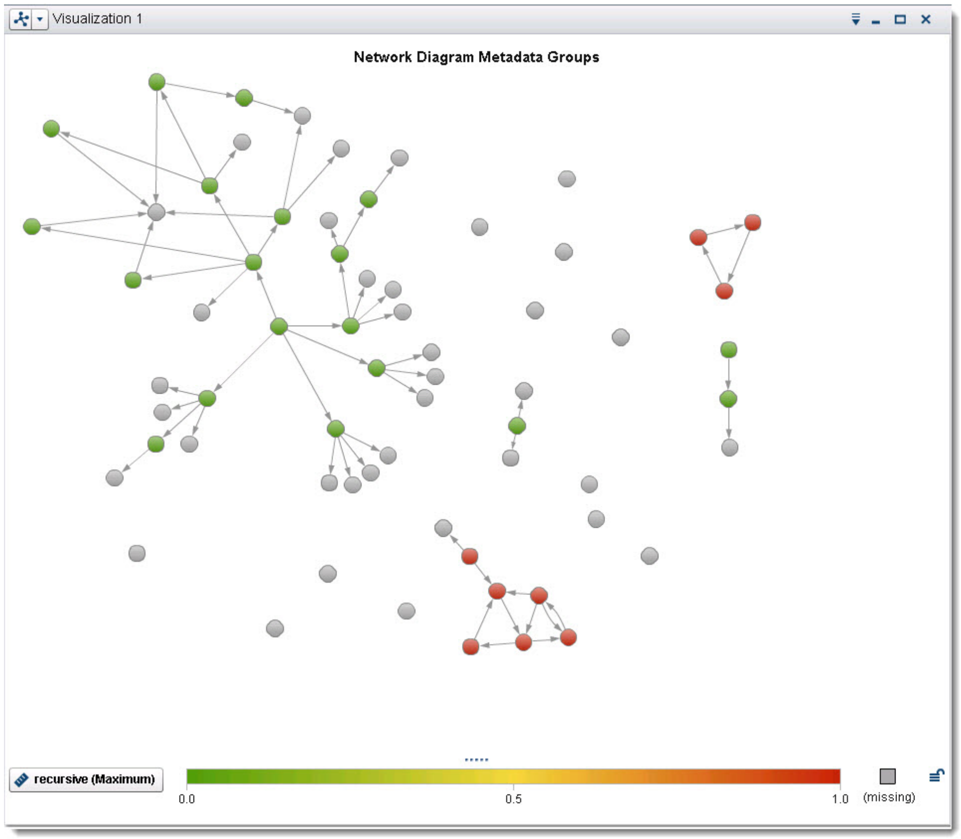
Table 1. Results of the PROC PRINT Listing for the GROUPLINE Table (Filtered for Recursive Group Lines Only)

For this table, the data has been filtered to show only circular group relationships (the value of the **recursive** column is **`1`**). The column **groupline** identifies the hierarchy of group memberships. You can see in the first row that the circular relationship is **`Green`**, **`Orange`**, **`Purple`**, and **`Green`**. **`Blue`** is the parent of **`Green`**, but it is not part of the circular relationship in this case (although it is part of a circular relationship with **`Yellow`** in observation 6).

Note: It is possible for a parent group not to have a child group associated with it. Group information is stored in metadata in such a way that each group has a record in which it is a parent. If the parent has one or more child groups associated with it, then there is one record for each parent/child combination.

USING SAS® VISUAL ANALYTICS TO CREATE NETWORK DIAGRAMS THAT SHOW RELATIONSHIPS

The SAS Visual Analytics Explorer – Network Diagram is a useful tool that graphically displays metadata group relationships from the GROUPLINE table.



Output 2. SAS Visual Analytics Explorer – Network Diagram

This output is an example that shows a network diagram that shows metadata group relationships. Notice the colors, where green indicates non-circular relationships between groups and red indicates a circular relationship.

In looking at the diagram above, you can see that there are two sets of three groups. The first set shows a three-level, linear parent-child relationship. The second set shows a circular group relationship.

Here is a more detailed view of the information from the diagram that shows labels as well as the organization of the data that produces each structure.

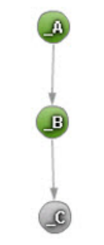
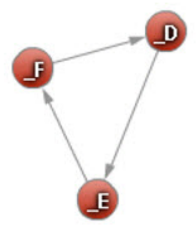
			
Data		Data	
Parent	Child	Parent	Child
_A	_B	_D	_E
_B	_C	_E	_F
_C		_F	_D

Table 2. Metadata Group Structures and Supporting Data

In this view, **Parent** and **Child** groups are identified in each row of data. When you use the code that is provided later in this paper to extract group metadata, every group is identified as a parent at least once. If a parent group has member groups that are associated with it, those members are identified as child groups. It is possible for a parent group not to have a child group associated with it, as illustrated in the table above with group _C.

The single network that is shown below is after a complex organizational structure. Based on the node color, you can see that there are no circular group relationships in this network. If you follow the directions of the arrows, you can see that some relationships that appear to be circular are, at a glance, actually linear.

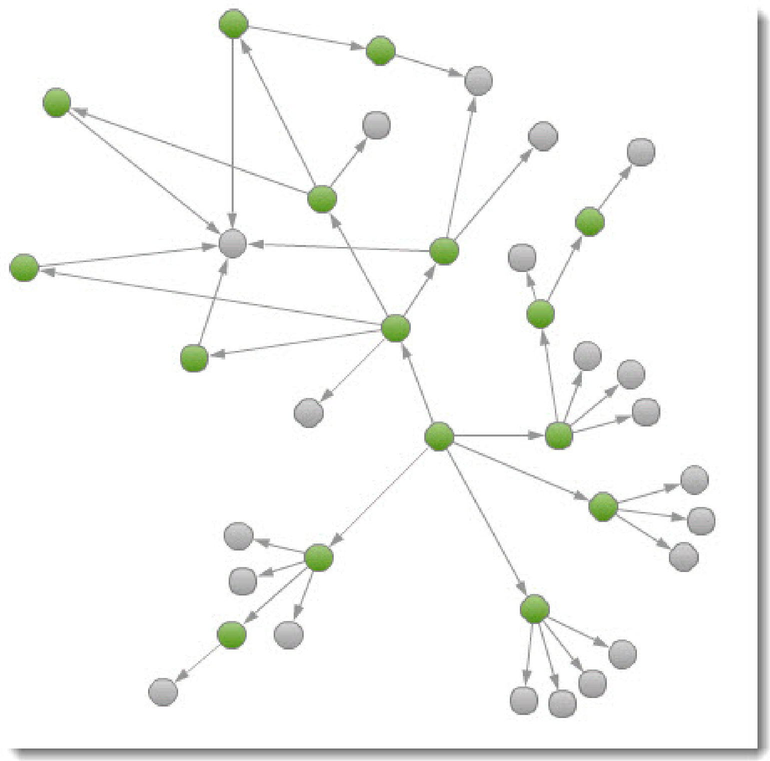


Figure 1. A Network Diagram Based on a Complex Organizational Structure

CREATING A NETWORK DIAGRAM USING SAS® VISUAL ANALYTICS

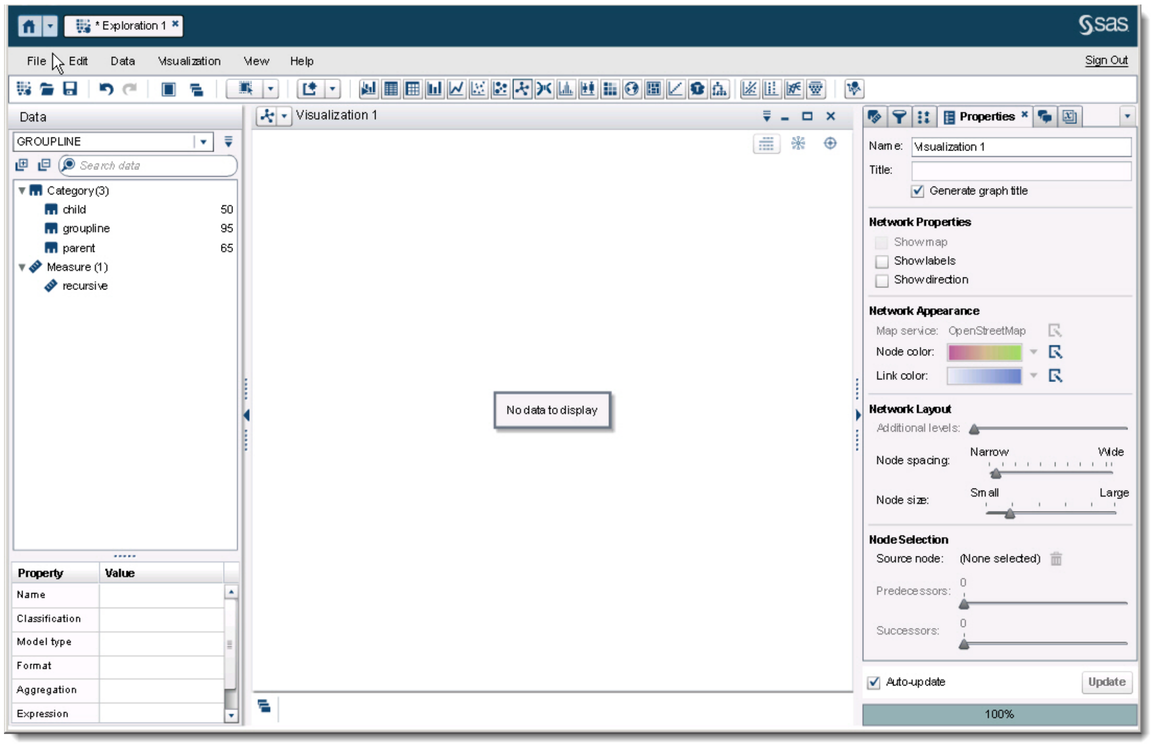
Now that you have seen examples of network diagrams, it is time to look at steps that you can use to create a network diagram from the GROUPLINE table (Table 1) that was created in the previous section.

If the GROUPLINE data is not already loaded into SAS® LASR™ Analytic Server, you can import the GROUPLINE table from SAS Visual Analytics Explorer.

Note: This paper does not detail how to use SAS Visual Analytics Explorer. Instead, it is assumed that you have working knowledge of the SAS Visual Analytics Explorer. For details about the explorer, see the See the *SAS® Visual Analytics: User's Guide* for your release of the software. (**`support.sas.com/documentation/onlinedoc/va/index.html`**)

To create a network diagram in SAS Visual Analytics Explorer:

1. Create a new visualization.
2. Import or reload the GROUPLINE table, if necessary, and add the table to the visualization.
3. Add a network diagram to the visualization.



GROUPLINE

Category (3)

child	50
groupline	95
parent	65

Measure (1)

recursive

Property	Value
Name	
Classification	
Model type	
Format	
Aggregation	
Expression	

Name: Visualization 1

Title:

✓ Generate graph title

Network Properties

Show map

Show labels

Show direction

Network Appearance

Map service: OpenStreetMap

Node color:

Link color:

Network Layout

Additional levels:

Node spacing: Narrow Wide

Node size: Small Large

Node Selection

Source node: (None selected)

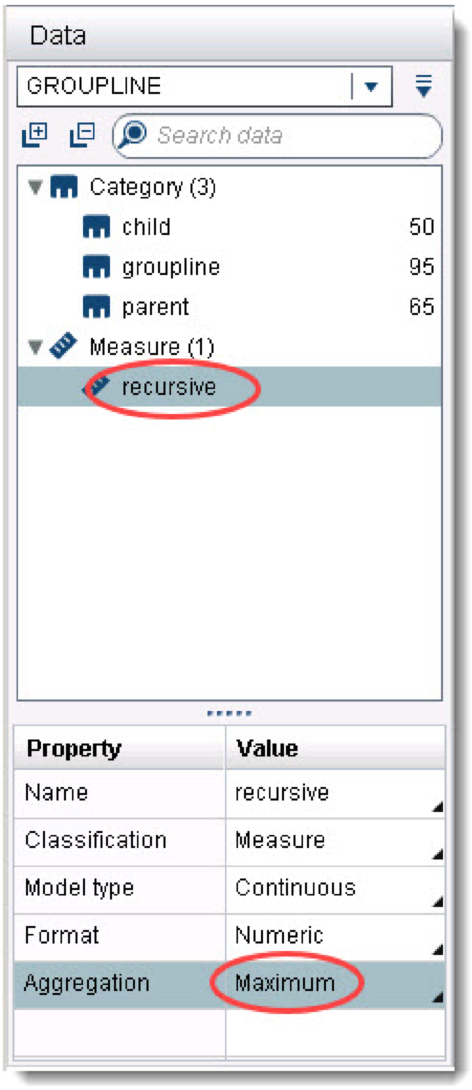
Predecessors: 0

Successors: 0

✓ Auto-update

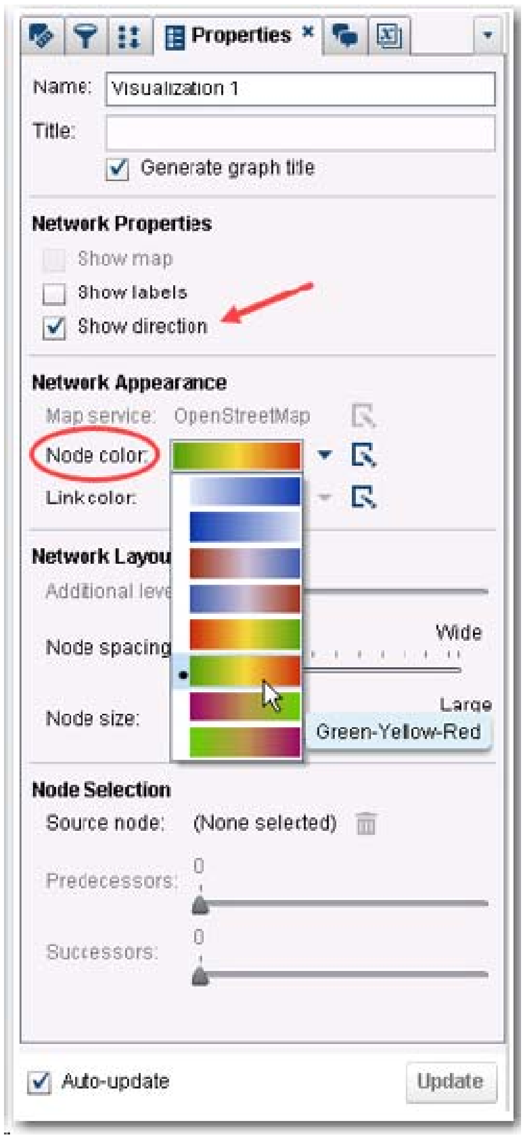
Update

4. In the Data pane, change the aggregation for **recursive** to **`Maximum`**.



Property	Value
Name	recursive
Classification	Measure
Model type	Continuous
Format	Numeric
Aggregation	Maximum

5. On the **Properties** tab:
   - Select the check box **Show direction**.
   - Change the value for **Node color** to a gradient that starts with green and ends with red.



Properties

Name: Visualization 1

Title:

[✓] Generate graph title

Network Properties

[ ] Show map

[ ] Show labels

[✓] Show direction

Network Appearance

Map service: OpenStreetMap

Node color:

Link color:

Network Layou

Additional leve

Node spacing — Wide

Node size: — Large

Green-Yellow-Red

Node Selection

Source node: (None selected)

Predecessors: 0

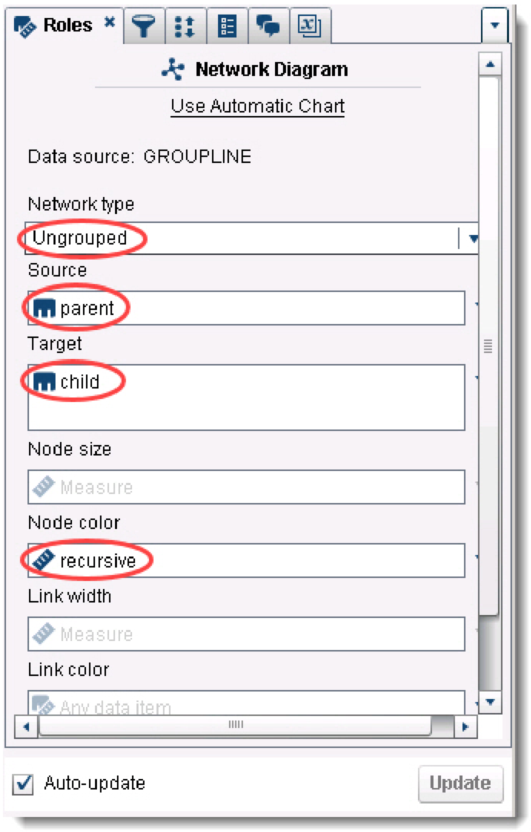
Successors: 0

[✓] Auto-update

Update

6. On the **Roles** tab, assign the following parameter values:

   - **Network type: `Ungrouped`**
   - **Source: `parent`**
   - **Target: `child`**
   - **Node color: `recursive`**

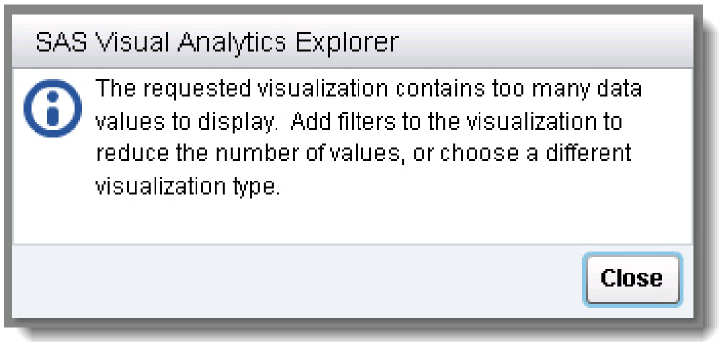


Data source: GROUPLINE
Network type: Ungrouped
Source: parent
Target: child
Node size: Measure
Node color: recursive
Link width: Measure
Link color: Any data item

Note: If you have a very large number of groups or parent/child relationships, it is possible that the amount of data can exceed the data limits that are imposed by the network diagram. The limits are as follows:

- 10000 rows of data
- 1000 nodes on the diagram
- 1000 links on the diagram

The following dialog box is displayed when you exceed a network diagram's data limit:

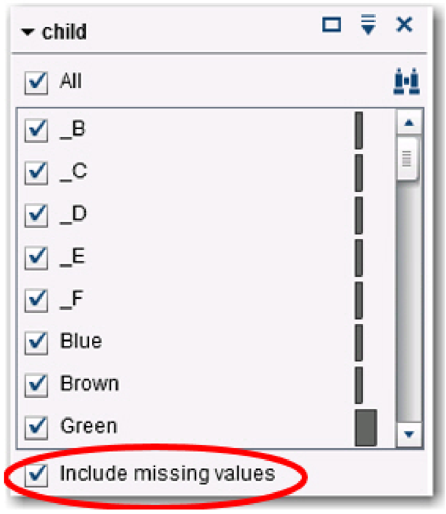


SAS Visual Analytics Explorer

The requested visualization contains too many data values to display. Add filters to the visualization to reduce the number of values, or choose a different visualization type.

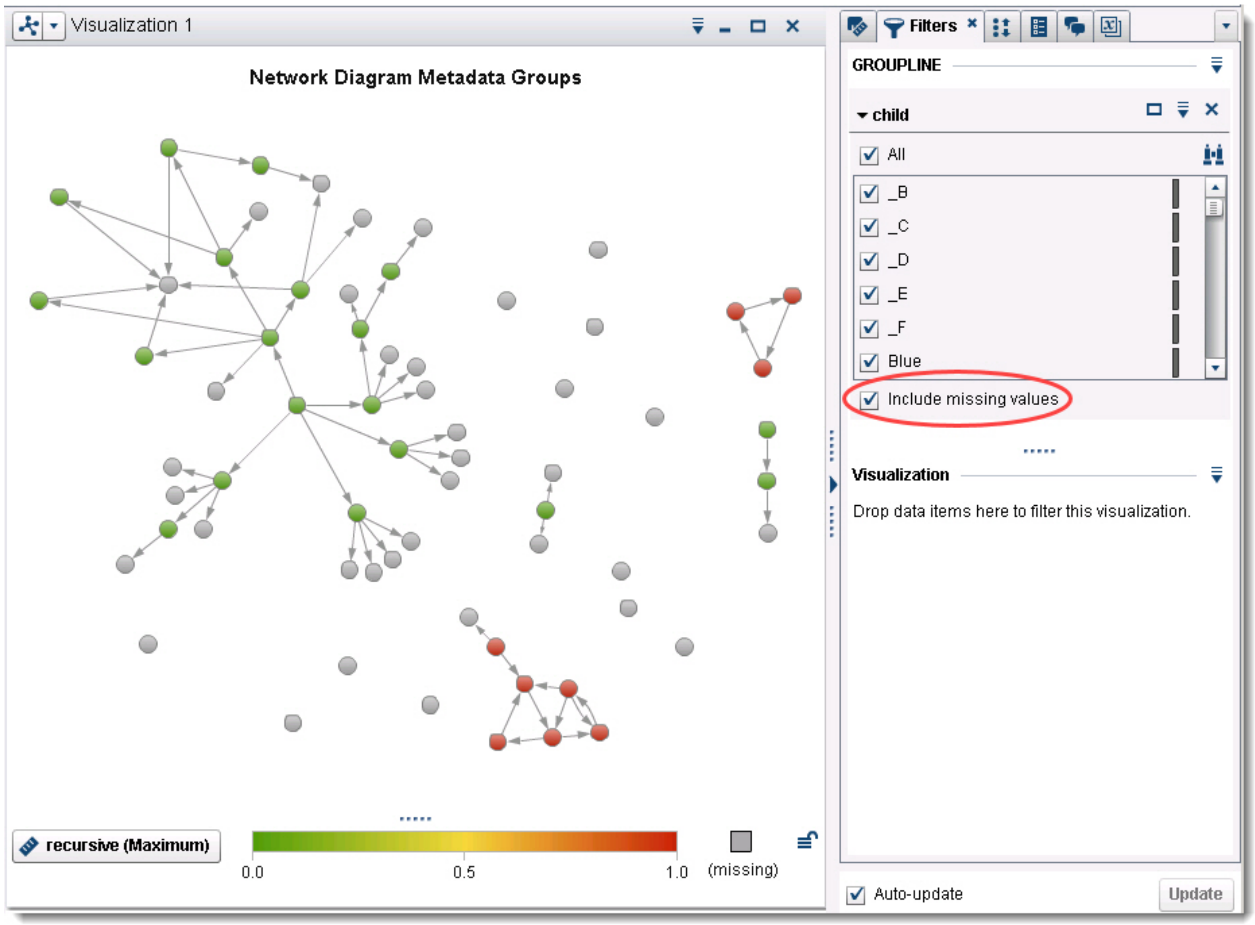
You can filter either the source data or the visualization to reduce the amount of data that the software attempts to display.

One option is to filter on the **child** column. To do that, clear the **Include missing values** check box.



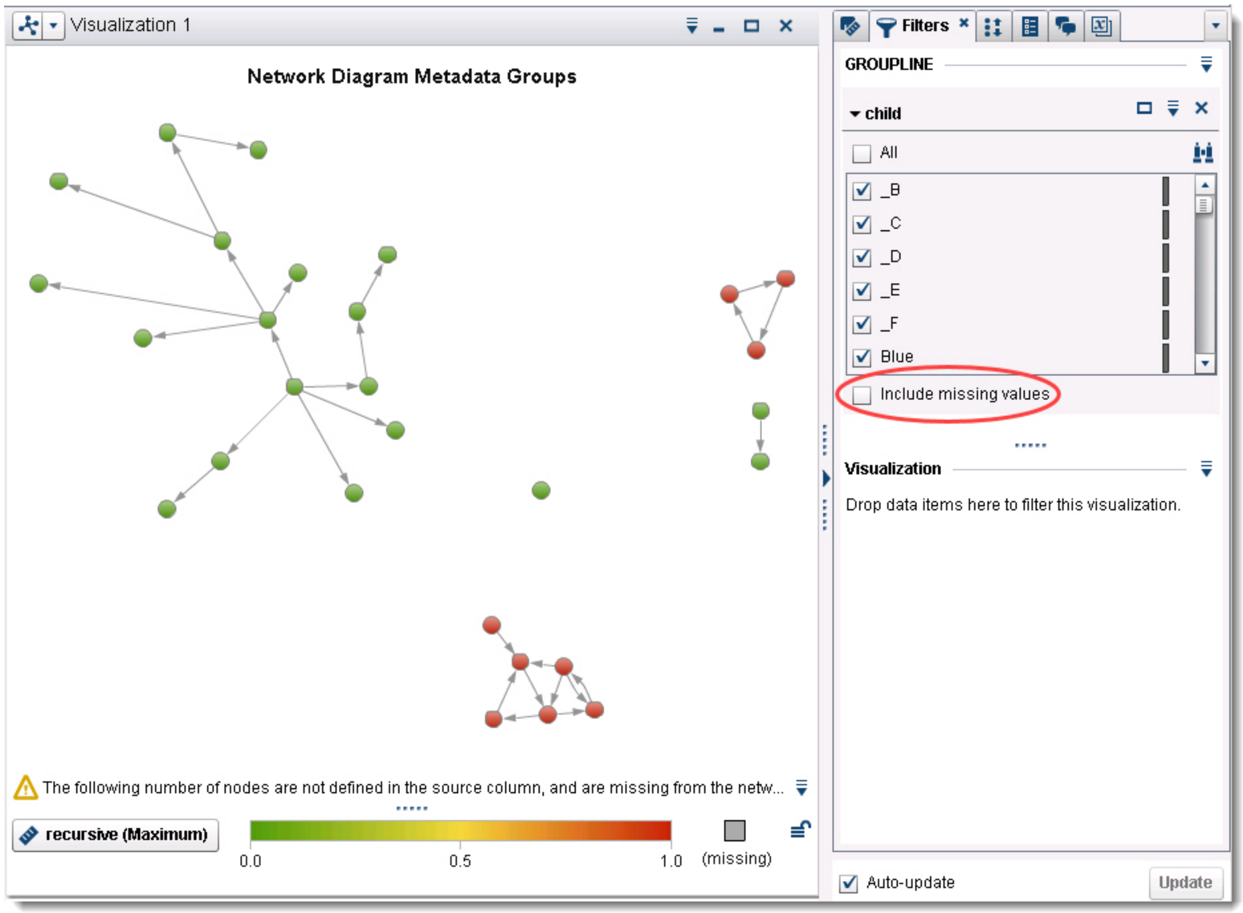
Display 1. Filtering on the Child Column

The unfiltered network diagram below includes missing values for **child**. A missing value for the **child** column effectively means a parent group with no children.



Output 3. Unfiltered Network Diagram That Includes Missing Child Values

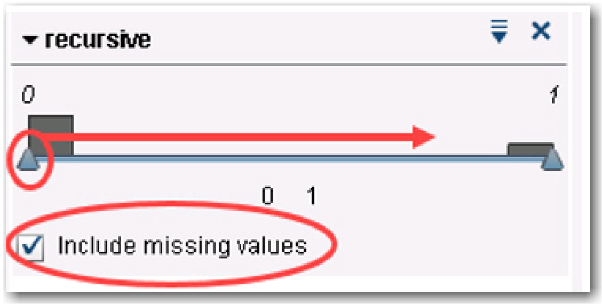
The following display shows the same network diagram, but with missing values for **child** removed:


Visualization 1
Network Diagram Metadata Groups
Filters
GROUPLINE
child
All
_B
_C
_D
_E
_F
Blue
Include missing values
Visualization
Drop data items here to filter this visualization.
The following number of nodes are not defined in the source column, and are missing from the netw...
recursive (Maximum)
0.0
0.5
1.0
(missing)
Auto-update
Update

Output 4. Network Diagram with Missing Child Values Excluded

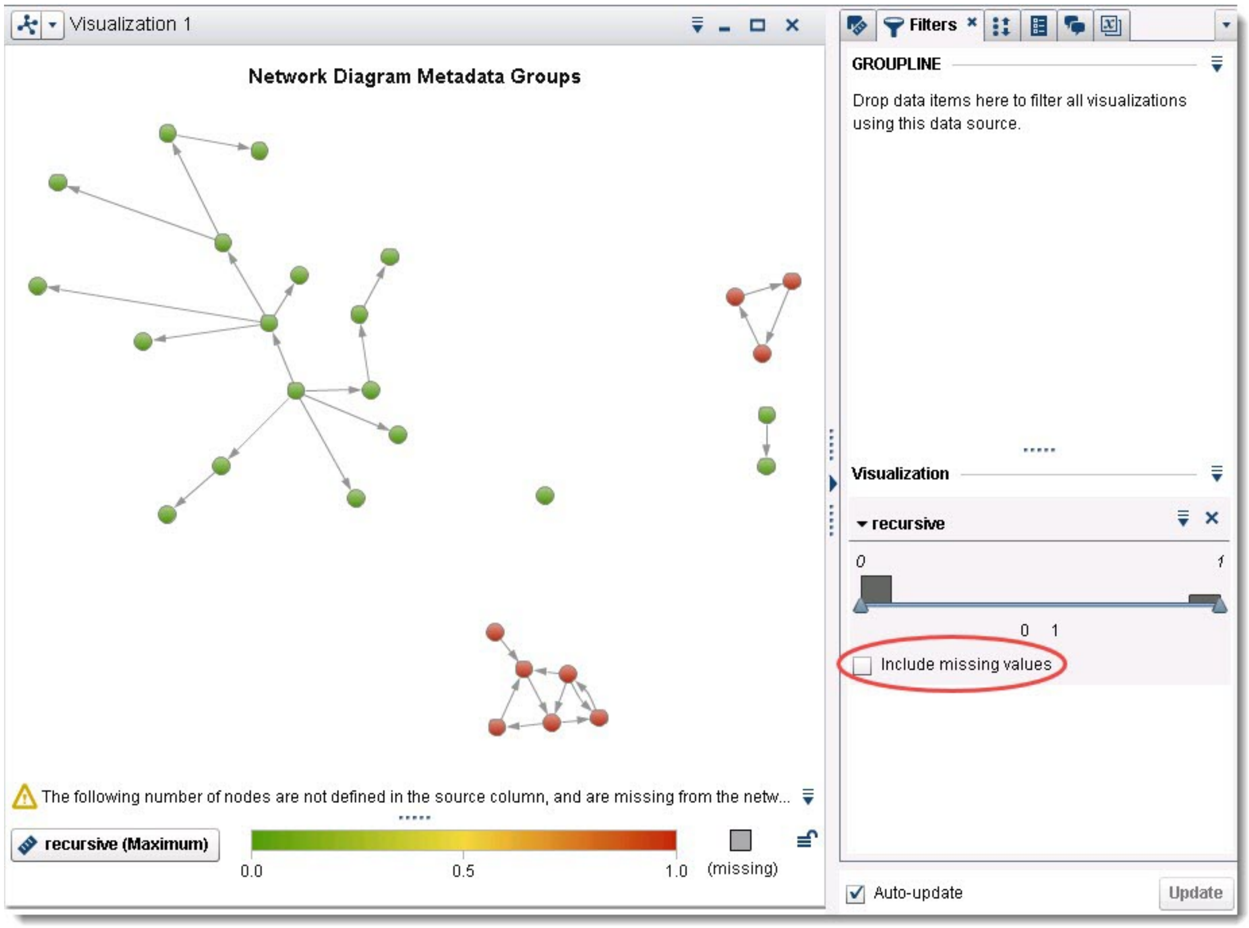
You can also filter on the **recursive** column. To fo that, include only those items that are indicated as being recursive:

- Move the slider for **recursive** to **1**.
- Clear the **Include missing values** check box



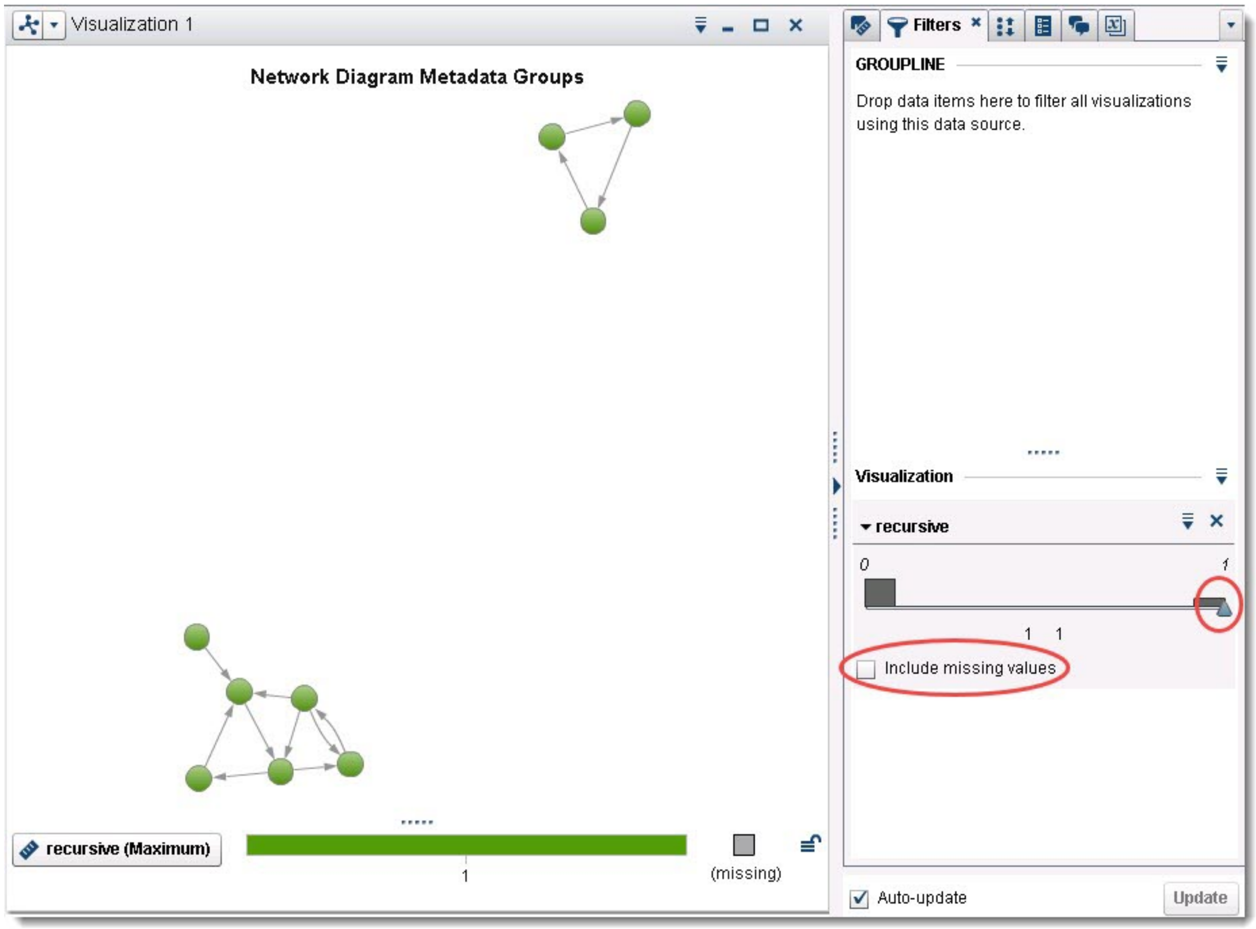
Display 2. Filtering on the Recursive Column

The following display is the same diagram that is shown earlier for filtering on the **child** column. But this diagram uses the recursive filter. Again, when missing values are excluded, only groups containing child groups are displayed.


Visualization 1
Network Diagram Metadata Groups
The following number of nodes are not defined in the source column, and are missing from the netw...
recursive (Maximum)
0.0
0.5
1.0
(missing)
Filters
GROUPLINE
Drop data items here to filter all visualizations using this data source.
Visualization
recursive
0
1
0 1
Include missing values
Auto-update
Update

Output 5. Using the Recursive Filter with Missing Child Values Excluded

You can also use the slider to filter the value of recursive to **1**. This setting only keeps groups that are directly involved in a circular relationship. Notice in the next display, Output 6, that filtering with the **recursive** data column set to **1** affects the node colors. In this case, you might want to adjust the node color palette on the **Properties** tab.

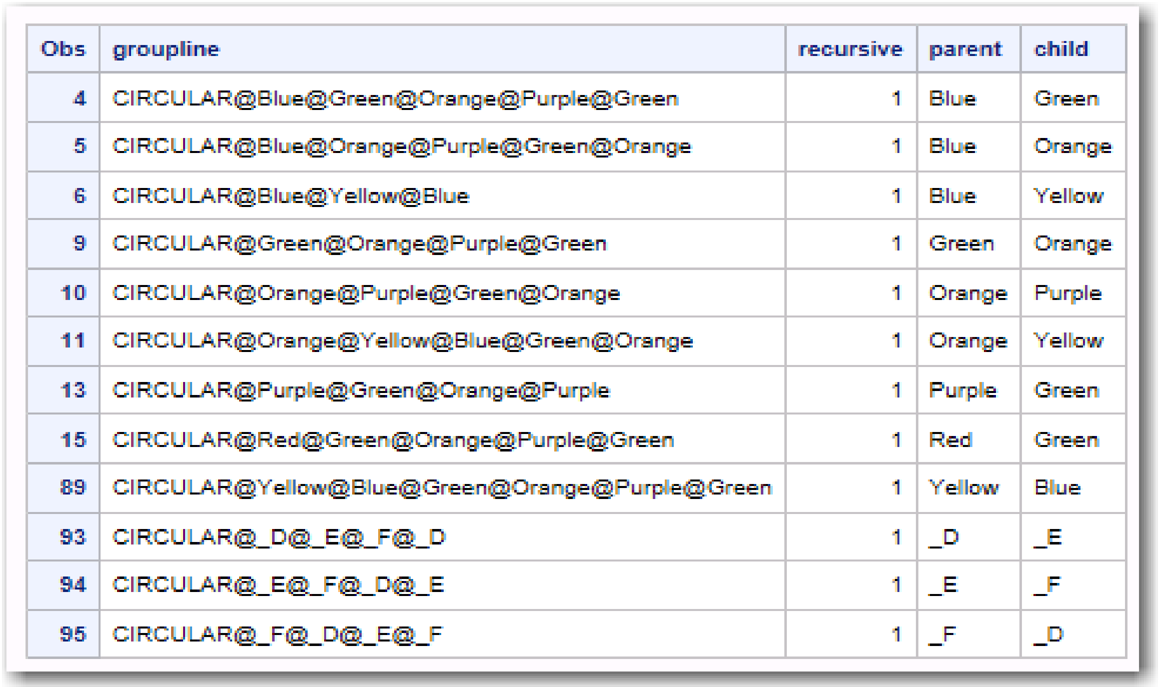


Output 6. Using the Slider to Filter the Value of the Recursive Data Column

By changing these settings, you reduce the amount of data that is processed. While you might not be able to see the complete metadata group structure, you can still identify circular group relationships.

USING SAS® CODE TO IDENTIFY CIRCULAR METADATA GROUP RELATIONSHIPS

You can include the Base SAS® code example from Processing Group Relationships in a stored process. Table 3 (which is the same as Table 1) results when you remove the comment delimiters from around the PROC PRINT step at the end of the code example.



Obs	groupline	recursive	parent	child
4	CIRCULAR@Blue@Green@Orange@Purple@Green	1	Blue	Green
5	CIRCULAR@Blue@Orange@Purple@Green@Orange	1	Blue	Orange
6	CIRCULAR@Blue@Yellow@Blue	1	Blue	Yellow
9	CIRCULAR@Green@Orange@Purple@Green	1	Green	Orange
10	CIRCULAR@Orange@Purple@Green@Orange	1	Orange	Purple
11	CIRCULAR@Orange@Yellow@Blue@Green@Orange	1	Orange	Yellow
13	CIRCULAR@Purple@Green@Orange@Purple	1	Purple	Green
15	CIRCULAR@Red@Green@Orange@Purple@Green	1	Red	Green
89	CIRCULAR@Yellow@Blue@Green@Orange@Purple@Green	1	Yellow	Blue
93	CIRCULAR@_D@_E@_F@_D	1	_D	_E
94	CIRCULAR@_E@_F@_D@_E	1	_E	_F
95	CIRCULAR@_F@_D@_E@_F	1	_F	_D

Table 3. Results of the PROC PRINT Listing for the GROUPLINE Table (Filtered for Recursive Group Lines Only)

You can also create your own reports, sourced by the GROUPLINE table, for investigation or auditing purposes. Note that the PROC PRINT step, as written below, does not produce a report or output table if there are no circular group relationships in your environment:

```
proc print data=a.groupline (where=(recursive=1));
run;
```

The resulting table (Table 3) displays only those records that are related to circular metadata group relationships so that you can investigate those relationships.

CONCLUSION

SAS enables you to create any metadata-group organizational structure that you can imagine. As you have seen, this functionality provides the ability to create circular group relationships. These recursive associations increase complexity in your environment and can negatively impact permissions (access to objects) and performance. The Base SAS code provided in this paper enables you identify circular group relationships in your environment. Tools such as the SAS Visual Analytics Explorer network diagram enable you to easily visualize the structure of your metadata group and highlight groups that are identified as part of a recursive relationship. In addition, Base SAS software's PRINT procedure can produce a table listing of group lines that help you investigate circular group relationships. You can use these tools to create a plan to correct recursive group memberships in SAS Management Console. Resolving circular relationships in metadata groups makes your environment more stable and enhances performance.

APPENDIX A: USING METADATA FUNCTIONS TO PRODUCE THE ALLGROUPS TABLE

The code in this appendix uses metadata functions to extract the parent/child group data from SAS Metadata Server. The function calls to the metadata server are a little complex because there are many nested levels of functions and because of the nested structure of the metadata. This code creates the ALLGROUPS table. The ALLGROUPS table contains the list of all metadata groups and children groups. This table is the input source for the transpose process. You can replace all code before the first TRANSPOSE procedure with the following code:

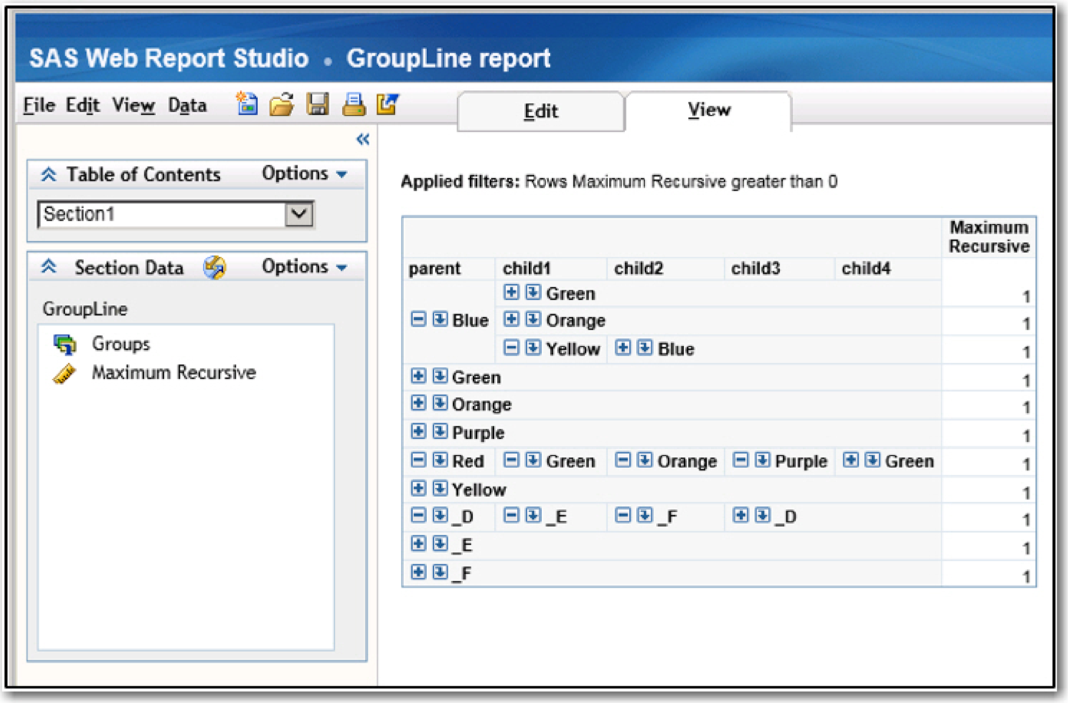
```
data work.allgroups(keep=parent child parent_id child_id flag);
   length GroupUri memberofuri attr_val $256 p_type type gptype attr_name
          $60 id $17 parent child parent_id child_id assoc assoc_uri $60;
   call missing(GroupUri, type, ptype, gptype, id, GroupName, memberofuri,
                memberofGroupName);
   group_obj="omsobj:IdentityGroup?IdentityGroup[@PublicType eq
                                                     'UserGroup']]";
   groups=metadata_resolve(group_obj,type,id);
   if (groups>0) then
      do n=1 to groups;
         nobj=metadata_getnobj(group_obj,n,GroupUri);
         rc=metadata_getattr(GroupUri,'PublicType', p_type);
   if p_type='UserGroup' then
      do;
        rc=metadata_getattr(GroupUri, 'Name', parent);
        rc=metadata_getattr(groupUri, 'Id', parent_id);
        child_id='';
        child='';
        flag=9;
        if metadata_getnasn(groupuri,'MemberIdentities',1,assoc_uri)<1 then
           output;
        else do;
            flag=0;
            do j=1 to
             metadata_getnasn(groupuri,'MemberIdentities',1,assoc_uri);
             rc=metadata_getnasn(groupuri,'MemberIdentities',j,assoc_uri);

                if substr(assoc_uri,1,21)='OMSOBJ:IdentityGroup\' then
                do;
                  flag=1;
                  child_id=substr(assoc_uri,22,17);
                  rc=metadata_getattr(child_id,'Name',child);
                  output;
                end;
             end;
             if flag=0 then output;
          end;
      end;
   end;
run;
```

APPENDIX B: USING AN OLAP CUBE TO DISPLAY CIRCULAR GROUP RELATIONSHIPS

For those who do not have SAS Visual Analytics licensed but do have SAS® Enterprise Business Intelligence, a SAS® Web Report Studio report based on an OLAP cube is an efficient way to display circular group relationships.

To create the source data for the OLAP cube, you need to parse the groups from the **groupline** column in the GROUPLINE table to create a Groups dimension in the OLAP cube. The GROUPLINE table (Table 1) is the same data table that is used to create the SAS Visual Analytics network diagrams that are highlighted throughout this document.

Output 7, shown below, displays a web report that is based on the GROUPLINE OLAP cube. Notice that the crosstab is filtered for rows with maximum recursive greater than `0`. In this way, only lineages with circular relationships are displayed. This filter is helpful for analysis and investigating problematic circular group relationships. Ideally, you need to remove this filter in order to display the GROUPLINE cube because (in the best-case scenario) no lineages meet the filter requirement of maximum recursive greater than `0`!



Applied filters: Rows Maximum Recursive greater than 0

parent	child1	child2	child3	child4	Maximum Recursive
Blue	Green				1
	Orange				1
	Yellow	Blue			1
Green					1
Orange					1
Purple					1
Red	Green	Orange	Purple	Green	1
Yellow					1
_D	_E	_F	_D		1
_E					1
_F					1

Output 7. SAS® Web Report Studio Report Based on the OLAP Cube

You can then view the GROUPLINE OLAP cube in the SAS Enterprise Guide cube viewer or in a SAS Web Report Studio report.

The following code uses the GROUPLINE table as the source data for the GROUPLINE OLAP cube.

```
   /* Specify the library for the groupLine_Lineage_cube data table, */
   /* which is the source data for the cube.                          */
   /* This library is also referenced in the OLAP procedure.          */
libname test 'c:\data\tables';

   /* Specify the library for the GROUPLINE table. */
libname sgf 'c:\temp';

   /* This code loops through the groups in the groupline table and   */
   /* parses the child groups into separate variables. Note that      */
   /* the number of child variables is not determined automatically.  */
   /* You either need to add code to programmatically determine how   */
   /* many child variables are needed or you need to manually adjust  */
   /* the number to suit your data. This particular example,          */
   /* only needs four levels (child1-child4). But one more level was */
   /* added just to show a missing child column at the end of each    */
   /* lineage.                                                        */
data test.groupLine_Lineage_cube (drop=dlm dlmEnd i s);

      /* Remember to set the maximum number of child variables in  */
      /* the ATTRIB and ARRAY statements!                          */
   attrib child1-child5 length=$60;
   set sgf.groupline;
   array c(*) child1-child5;
   if child ^='' then
      do;
             /* Some parent groups in a group line that contains a    */
             /* circular group relationship might not have the        */
             /* recursive flag set. In this case, the parent group is */
             /* not directly involved in the circular relationship.   */
             /* For example, in the group line A-B-C-D, the parent    */
             /* group A will not be flagged as recursive.             */
         if substr(groupline,1,1) NE '@' then s=10;
            else s=2;
            i=1;
               do while (s ^= 0);
                  dlm=find(groupline,'@','i',s);
                  dlmEnd=find(groupline,'@','i',dlm+1);
                  if dlmEnd > 0 then
                     do;
                        c(i)=substr(groupline,dlm+1,dlmEnd-dlm-1);
                        s=dlmEnd;
                  end;
                  else do;
                     c(i)=substr(groupline,dlm+1);
                     s=0;
                  end;
                  i+1;
         end;
      end; /* child ^= 0 */
run;

   /* Register the table metadata so the table can be used in PROC OLAP. */
proc metalib;
   omr (library="Test Library" metarepository="Foundation");
   select (groupLine_Lineage_cube);
run;

   /********  Remove the comment delimiters from this code if you want to
   delete the cube
OPTIONS VALIDVARNAME=ANY;
PROC OLAP
   CUBE="/Shared Data/Cubes/GroupLine"
   MAX_RETRIES=3 MAX_RETRY_WAIT=60 MIN_RETRY_WAIT=30
   DELETE;
   METASVR OLAP_SCHEMA="SASApp - OLAP Schema";
RUN;
```

```
********    End of code comment block    ********/

   /* The OLAP procedure code below uses the GROUPLINE table that was   */
   /* ultimately created from the %MDUEXTR macro. This cube definition  */
   /* includes two dimensions: Groups and Parent/Child. The Groups      */
   /* dimension displays the lineage of the groupline column. The        */
   /* Parent/Child dimension is not necessary for displaying circular   */
   /* group relationships in a web report, but it might be useful if    */
   /* you want to view a flat listing of parent/child relationships.    */
   /*                                                                   */
   /* Note: Along with the obvious updates of cube and METASVR          */
   /* parameters, be sure to update the child columns that are listed   */
   /* the Groups dimension. Ideally, this process should be             */
   /* parameterized and placed in a macro.                              */
OPTIONS VALIDVARNAME=ANY;
PROC OLAP
   CUBE="/Shared Data/Cubes/GroupLine"
   MAX_RETRIES=3  MAX_RETRY_WAIT=60  MIN_RETRY_WAIT=30
   DATA=test.GROUPLINE_LINEAGE_CUBE PATH='C:\data\olap\cubes'
   DESCRIPTION='GroupLine';
   METASVR OLAP_SCHEMA="SASApp - OLAP Schema";

       /* Remember to update this DIMENSION section for the appropriate */
       /* number of child levels!                                       */
      DIMENSION Groups
         CAPTION='Groups' SORT_ORDER = ASCENDING
         HIERARCHIES=( Groups ) /* HIERARCHIES */;

         HIERARCHY Groups
            ALL_MEMBER='All Groups' CAPTION='Groups'
            LEVELS=( parent child1 child2
                     child3 child4 child5 ) /* LEVELS */  DEFAULT;

         LEVEL parent CAPTION='parent' SORT_ORDER=ASCENDING;
         LEVEL child1 CAPTION='child1' SORT_ORDER=ASCENDING;
         LEVEL child2 CAPTION='child2' SORT_ORDER=ASCENDING;
         LEVEL child3 CAPTION='child3' SORT_ORDER=ASCENDING;
         LEVEL child4 CAPTION='child4' SORT_ORDER=ASCENDING;
         LEVEL child5 CAPTION='child5' SORT_ORDER=ASCENDING;

   DIMENSION 'Parent/Child'n
      CAPTION='Parent/Child' SORT_ORDER=ASCENDING
      HIERARCHIES=( 'Parent/Child'n ) /* HIERARCHIES */;

      HIERARCHY 'Parent/Child'n
         ALL_MEMBER='All Parent/Child' CAPTION='Parent/Child'
         LEVELS=( parent_1 child ) /* LEVELS */  DEFAULT;

      LEVEL parent_1 COLUMN=parent CAPTION='parent' SORT_ORDER=ASCENDING;
      LEVEL child CAPTION='child' SORT_ORDER=ASCENDING;

   MEASURE recursiveMAX
      STAT=MAX COLUMN=recursive CAPTION='Maximum recursive' FORMAT=12.
      DEFAULT;
   MEASURE recursiveSUM
      STAT=SUM COLUMN=recursive CAPTION='Sum of recursive' FORMAT=12.;
      MEASURE recursiveN
         STAT=N COLUMN=recursive CAPTION='Number of values for recursive'
         FORMAT=12.;
      MEASURE recursiveAVG
         STAT=AVG COLUMN=recursive CAPTION='Average recursive' FORMAT= 12.;
      MEASURE recursiveMIN
         STAT=MIN COLUMN=recursive CAPTION='Minimum recursive' FORMAT= 12.;

      AGGREGATION
         child child1 child2 child3 child4 child5 parent parent_1 /
         NAME='Default';
   RUN;
```

REFERENCES

- Downloadable file for the code examples that are discussed in this paper: **ftp.sas.com/techsup/download/SGF2017/CircularMetadaGroups/SAS381-2017_SAS_Code.zip**
- SAS Institute Inc. 2016. *SAS® 9.4 Intelligence Platform: Security Administration Guide, Third Edition*. SAS Institute Inc.: Cary, NC. Available at **support.sas.com/documentation/cdl/en/bisecag/69827/PDF/default/bisecag.pdf**.
  - "Metadata Groups" in "Chapter 2: User Administration" (**support.sas.com/documentation/cdl/en/bisecag/69827/PDF/default/bisecag.pdf#page=27**)
  - "User Bulk Load" in "Appendix 2: User Import Macros" (**support.sas.com/documentation/cdl/en/bisecag/69827/PDF/default/bisecag.pdf#page=248**)
  - "Identity Hierarchy" in "Chapter 2: User Administration" (**support.sas.com/documentation/cdl/en/bisecag/69827/PDF/default/bisecag.pdf#page=31**)
  - "%MDUEXTR" in "Appendix 2: User Import Macros" (**support.sas.com/documentation/cdl/en/bisecag/69827/PDF/default/bisecag.pdf#page=261**)

CONTACT INFORMATION

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

Greg Lehner
SAS Institute Inc.
SAS Campus Drive
Cary, NC 28513
Email: **support@sas.com**
Web: **support.sas.com**

Karen Hinkson
SAS institute Inc.
SAS Campus Drive
Cary, NC 27513
Email: **support@sas.com**
Web: **support.sas.com**