ABSTRACT

Have you ever wanted to integrate your databases with your floor plans, to visualize the data spatially? — If so, then read on!

This paper describes how you can create simple SAS® maps of your floor plans, and then represent data from your inventory, employee, and other databases on the maps.

INTRODUCTION

The technique was developed for internal use at SAS Institute to help visualize inventory data. It was implemented in a system called "lococ — Bob's Crazy Locator Program," which proved so useful in visualizing the inventory data, that it was expanded to interfaced with the employee, computer network, and several other databases.

The loco system has been used for many purposes, including: locating computer equipment with hardware problems, assigning computer network addresses, locating equipment missed in inventory scans, drawing maps to show departmental "boundaries," helping visitors find offices, and identifying areas with UPS power failures.

The same technique was even used to develop a system to help locate demos of interest on the SUGI demo floor.

A SIMPLE EXAMPLE

First, you will need to create a simple SAS map data set of your floor plan. In this data set, you can represent each room as a rectangle, specified with four (x,y) coordinates. The basic technique, summarized below, is described in greater detail on page 1008 of the "SAS/Graph Software, Volume 2, Reference Version 6, First Edition."

A map data set is a SAS data set that defines the boundaries of unit areas, such as states or counties (or, in this case rooms). A map data set must contain the following variables:

1) a numeric variable named X containing the horizontal coordinates of the boundary points.

2) a numeric variable named Y containing the vertical coordinates of the boundary points.

3) one or more variables (eg. "room") that uniquely identify the unit areas on the map. These variables are used in the ID statement of PROC GMAP.

Using this strategy, you can overlay an x,y grid on your floor plans, estimate the coordinates of the corners of your rooms, and enter these coordinates into a SAS map data set.
The following **simple example** shows how to generate a SAS map data set representing a floor plan containing three rooms. In the first step (Figure 1), you estimate the points representing the corners of the rooms on an x,y grid.

![Figure 1](image)

The following sample code shows how you can create a SAS map data set containing these x,y coordinates. Notice that there are four x,y pairs for each room.

```sas
data flrplan;
  input room x y @@;
  cards;
  1 0 0 1 0 40
  1 40 40 1 40 0
  2 60 0 2 100 0
  2 100 40 2 60 40
  3 0 50 3 100 50
  3 100 100 3 0 100
; run;
```

Next, you can plot the map using the **GMAP** procedure, as shown in the following code. For simplicity, you can use the "flrplan" data set as both the map and the data.

```sas
proc gmap map=flrplan
  data=flrplan all;
  pattern v=s c=red;
  id room; choro room /
  coutline=black levels=1
cempty=black nolegend;
run;
```

Figure 2 shows the three rectangles which are produced by the code — very simple, very easy, eh!. Of course, this simple example doesn't tell you much information about the rooms or their contents, but that part is taken care of in the next example.

![Figure 2](image)

I'm sure your imagination is shifting into high gear, now. You're probably saying, "Since I can create my own customized maps, I can organize, analyze, and present all of my spatial and attribute data, not just the ones that fit the standard maps!"

**A MORE COMPLEX EXAMPLE**

In true SUGI style, I have chosen the main SUGI demo room floor plan for a detailed example. Of course, the demo room layout will probably change somewhat between...
publication time and the SUGI conference, but it does provide an excellent example of a floor plan with fairly complex layout and labelling requirements, ... and you might even find the map useful in finding your way around the demo room at SUGI!

CREATING THE SAS MAP DATA SET

The first step is to decide what to represent on your SAS map data set. Whatever entities you choose, make sure they all have unique numbers or labels you can use to refer to them. For most floor plans, you will want to represent rooms, but in the case of the SUGI demo room the stations (i.e., tables) are the natural choice — therefore, each time I say “station” and “statno” you will probably want to substitute the words “room” and “roomno” when relating the example to a traditional floor plan.

The most labor-intensive part of this technique is entering four \( x, y \) coordinates for each station. You might have access to a simpler method, such as the digitizing tables used to enter apparel patterns. If you are extremely lucky, and your “rooms” are very uniformly shaped and positioned, you may even be able to generate the SAS map data set programmatically. To keep this example simple, the brute-force method of creating an ASCII file by hand is demonstrated in the example — feel free to enhance any part of this process as your expertise allows.

First, create a “\( flrplan.dat \)” ASCII file to represent the coordinates of each station. Each line of the ASCII file should contain a station number, followed by an \( x, y \) coordinate — there will be four lines of data per station, representing the 4 corners of the station. The following file fragment shows the format of the ASCII file used to represent the SUGI demo room floor plan.

The sample code following the “\( flrplan.dat \)” file can be used to read the \( x, y \) coordinates from the ASCII file, and create the SAS map data set called “\( flrplan \)”. Notice that several of the datasets, are stored in the “datasets” SAS library, allowing them to be used by later SAS sessions, without needing to read the ASCII file again.

```
data datasets.flrplan;
  infile "flrplan.dat"
  input statno $ 1-8 x y;
  density=0;
  statno=trim(left(statno));
run;
```

While you’re working with the station coordinates, go ahead and calculate the center coordinates of each station, ... believe me, it will come in handy later. You can calculate the centers in many ways — the example below uses a simple SQL query.

```
proc sql;
  create table centers as
  select unique statno, 
  (max(x)+min(x))/2 as x, 
  (max(y)+min(y))/2 as y 
  from datasets.flrplan
  group by statno;
quit; run;
```
NUMBERING THE STATIONs

As shown in the first/simple example, a map is of little use without some annotation to help the readers know what they are looking at. Perhaps the most useful, and simple, form of annotation is to print the station (or room) numbers on the map.

Using the "centers" data set, you can easily create a SAS annotate data set containing the station numbers. The following SAS code shows how to create such an annotate data set, called "number" by simply adding some variables to the "centers" data set.

    data datasets.number;
    length function color text $ 40;
    retain function 'label'
          xsys ysys '2' hsys '3'
          when 'a';
    set centers;
    text=trim(left(statno));
    style='swissl';
    color='black';
    size=1.2; position='B';
    output;
run;

The position='B' causes the station numbers to be placed just above the center coordinate of the station. You might also need to adjust the size of the text.

LABELING THE STATIONS

In addition to numbering the stations, you might also want some textual labels. One way to do this is to specify the desired text for each station, as well as the x,y offset from center, position, angle, and size of the label.

For maximum flexibility, the text of the labels are kept in a separate file in this example — the data could also be queried from a database, but that would be difficult to show in this example. The ASCII file fragment below shows some sample "des_text.dat" data, and is followed by some SAS code that could be used to read the data into a SAS data set called "text".

    data text;
    infile "des_text.dat"
          lrecl=80
          pad missover;
    input statno $ 2-4
          demo $ 11-51;
run;

Here is some sample "des_loc.dat" data, with comments added to show which column becomes which variable. Each line in the "des_loc.dat" file shows the x-offset and y-offset from the center coordinate, the position, angle of rotation, size, and then the station number.

The following SAS code reads the "des_loc.dat" ASCII file into a SAS data set called "loc", which later becomes the "descrip" SAS data set.
The next step is to merge the “text” and "loc" data sets, forming a data set called "descrip". You can use whatever method you’re most comfortable with — SQL is used in this example.

```
data loc;
  infile "des_loc.dat"
    lrecl=80 pad missover;
  input x_off 1-4 y_off 6-9
  position $ 11 angle 13-15
  size 17-19 statno $ 21-24;
run;
```

```
data datasets.descrip;
  length function color $ 20;
  length text $ 40;
  retain function ‘label’
xsys ysys ‘2’ hsys ‘3’
  when ‘a’;
  style=’zapf’;
  color=’black’;
  set descrip;
  text=trim(left(demo));
run;
```

**SELECTING STATIONS OF INTEREST**

Now that you have the SAS map data set, and the annotate data set(s), you’re ready to plot maps that can help you analyze your data spatially.

First, we’ll create a data set containing the station numbers of interest (or, in your case, these will likely be room numbers). In this simple example, we’ll use data set “a”. Since this example concentrates on producing the maps, and not building GUI interfaces, the stations of interest are simply hard-coded in the sample code below.

```
data a;
  input statno $ quantity;
cards;
  111 1
  115 1
  161 1
; run;
```

Last, a little cleanup is done, and variables are added so “descrip” can be used as a SAS

```
proc sql;
create table descrip as
select loc.*, text.demo
from loc left join text
on loc.statno =
  trim(left(text.statno));
```

```
create table descrip as
select descrip.*,
centers.x + x_off as x,
centers.y + y_off as y
from descrip
  left join centers
on descrip.statno= centers.statno;
quit; run;
```

Once the map is plotted, these stations of interest show up as colors or shades on the floor plan (See Figure 3).
Figure 3

SUGI 21 Demo Room

Analysis Theater 131

Presentation Theater 116

Entrance

Information Visualization
PLOTTING THE FLOOR PLAN MAPS

You're probably saying ... "Wow, that was pretty neat, Rob! But, I think you left out the main piece of code — how do I actually produce the plot from the SAS data sets?"

Well, if you've read any of my other papers, you know that I would never leave you hanging in suspense, like that — the final details are described below!

During the final stages of plotting the map, it is often useful to add some additional text to the map. A title is almost always desirable, and if your floor plan is oriented conveniently, you can use a footnote to indicate that the "front entrance" is at the bottom of the map. Also, a pattern statement is used to select the color for the stations of interest — bright colors show up well when you use color screens and printers, but you should use light colors (such as yellow, or a light gray) if you only have a black-n-white printer, and want to be able to read the station numbers on a printout.

```
title "SUGI 21 Demo Room";
footnote "Entrance";
pattern v=s c=red;
```

The final step is to use the GMAP procedure to plot the data, and overlay the annotate data sets containing the station numbers and the textual descriptions. The final plot is shown on the preceding page (Figure 3).

```
proc gmap data=a
  map=datasets.flrplan
  anno=datasets.number all;
id statno;
choro quantity /
  anno=datasets.descrip
  levels=1 coutline=black
  cempty=black nolegend;
run;
```

POSSIBLE “GOTCHAS!”

A little experience can save a lot of time, so I'm going to pass on a few tips that I learned the hard way :-)!

The order of the 4 coordinates for each room is important, because the GMAP procedure will connect the coordinates in the order specified. You can order your points so the rectangle is drawn clockwise, or counter-clockwise, whichever is more convenient. But do not sort the SAS map data set by the x,y coordinates, or you will get bow ties instead of rectangles (see Figure 4).

Do not try to include too much detail in your maps! Rectangles are fairly easy to code, and will suffice for most purposes. If you try to model every subtle corner and bend, well,... I hope your pain threshold is higher than mine.

Also, choose a coordinate system that will make calculating the coordinates of the corners easy. For example, if many of your rooms are the same size, you could say that the length of those walls is 100. Remember the purpose of these plots is to help visualize your data spatially — not to serve as blueprints that will be used for construction purposes.

Also, if you are mapping several floors, it is convenient to name your SAS map data sets such that you can programmatically determine which floor plan to use, based on the room#. For example, room 321 would intuitively be contained in the "floor3" data set.
FUTURE ENHANCEMENTS

Of course, the example described in this paper is just the “tip of the iceberg” — the basic technique presented here could be taken much further.

For example, you will likely want to build some sort of GUI interface to help the users easily select stations (or rooms) of interest. The GUI interface could prompt the user for the rooms of interest (in some user friendly way), create a data set containing those room numbers, and then plot the data onto the maps, as shown above.

There is also great potential for applying this technique to other areas of visualization. Basically, any picture that you can draw using simple graphical elements, and then acquire data values for specific elements in the picture, can be visualized using this technique.

For example, yarn properties have been mapped onto “fabric maps” to predict the appearance of the woven fabric that could be made from the yarn (See Figure 5). In this visualization, each visible segment of yarn is analogous to a statnno in the previous example. Also, the x,y coordinates for the yarn segments were calculated programmatically, based on the measured yarn values.

Another area with great potential is using your SAS map data set floor plans with SAS/GIS®. This opens a whole new level of functionality to the users, allowing them to interact with the maps — for example, they could click on a room to see a list of all the inventory in that room.

Hopefully this paper will inspire you to do something totally new, and previously thought impossible, in your own areas of visualization. Remember, the power of SAS® software is limited only by your imagination!

REFERENCES


SAS, SAS/GRAPH, and SAS/GIS are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration.

The author welcomes questions, suggestions, and comments:

Robert E. Allison, Jr.
realliso@unix.sas.com
SAS Campus Drive
Cary, NC 27513 USA