Paper 612-2017

Using Big Data to Visualize People Movement Using SAS® Basics

Stephanie R. Thompson, Datamum

ABSTRACT

Visualizing the movement of people over time in an animation can provide insights that tables and static graphs cannot. There are many options, but what if you want to base the visualization on large amounts of data from several sources? SAS® is a great tool for this type of project. This paper summarizes how visualizing movement is accomplished using several data sets, large and small, and using various SAS procedures to pull it together. The use of a custom shape file is also highlighted. The end result is a GIF, which can be shared, that provides insights not available with other methods.

INTRODUCTION

Student success is an important metric at many colleges and universities. One project wanted to evaluate student movement patterns over time. This information could inform public safety, snow removal priorities, dining scheduling, and determining student engagement. Looking at static reports could answer these questions but seeing the data on an actual campus map over time would prove more enlightening.

The project merged data from several sources and rolled data up to the building level. Snapshots were taken or generated at 10 minute intervals over several days within a semester. This information would be plotted to a map of the campus. A gif animation would be created to visualize the aggregated data over time.

All of this work, including the animation, was done with relatively basic SAS procedures. Accessing, cleaning, and compiling data are strong points of SAS. Plus, the ability to generate and plot to a map made SAS the perfect tool for the job.

Please note that all visualizations and data shown are for illustrative purposes only.

GETTING THE DATA

The data for the project were in several systems. The student schedule data was in an Oracle database. This database was generated from an extract of the student information system into a data warehouse. SAS/ACCESS to Oracle was used to make this connection and PROC SQL pulled the necessary data.

Corse specific information was also extracted from the data warehouse. This information was in addition to the student schedule data. It provided class schedule details such as classroom, building, meeting days, and class times. The course information was joined to the schedule data to provide one dataset that would show where students should be for classes.

Student locations at times other than during class was determined by the use of wireless access data. The campus has a large number of wireless access points and connections were used to determine student location based on information from this data. Wireless data does not reside in a structured database and an extract was developed to capture it. The main system only keeps this data for approximately 30 days and then it is overwritten. Therefore, extracts were needed to be able to use this data for analytics. The extracts were transformed and then loaded into an Oracle database for project use.

The last piece of data was a dataset that contained the time of the day in 10 minute intervals. Ten minutes would capture most of the class start and end times. This interval was also granular enough to show movement over time in an animation without being too choppy or too long. The time intervals were generated in Excel then imported into SAS. This time dataset was used to merge to schedule and wireless data.

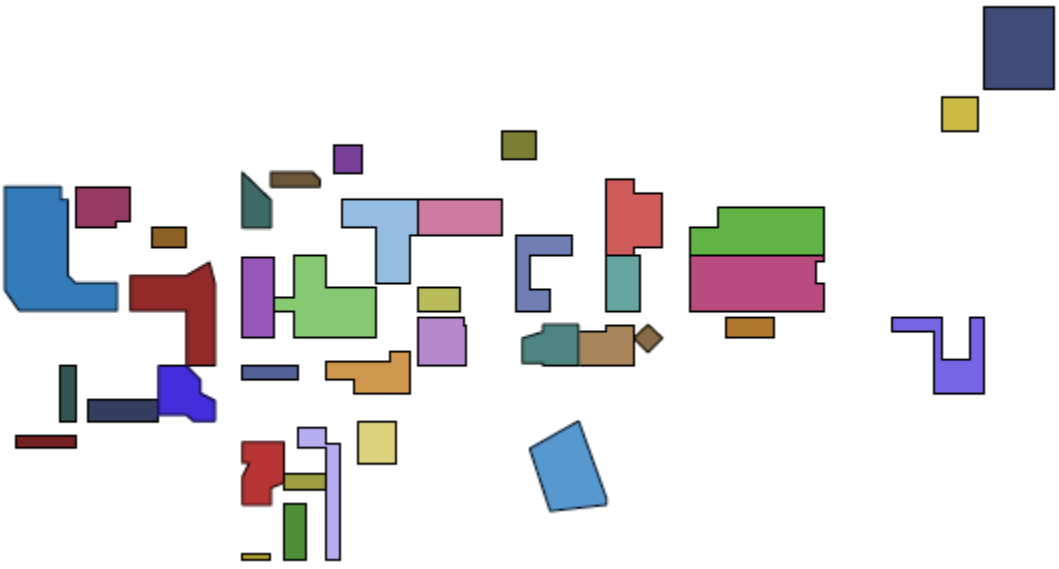
YOU NEED A MAP?

A map of the campus was needed to visualize the data. The team wanted to use the buildings themselves as the basis for plotting occupancy. Therefore, a heat map superimposed on an existing map would not suffice. This is definitely one option to create this type of visualization but it tends to be less exact than plotting to the actual buildings. The goal was to generate a visualization similar to one where each state in the United States is colored based on population or some other characteristic.

Existing maps seemed to abound. There were several variations on the institute's web site and the facilities department had several types of CAD drawings of the campus and all of the buildings. The SAS procedure to be used for the visualization needed either latitude and longitude or x and y coordinates along with a structure identifier (a shape file). Unfortunately, the online maps could not be saved into the proper format and an existing underlying shape file was not available.

The next candidate were the CAD drawings from the facilities group. CAD could not be read directly by SAS so the files needed to be exported to the proper format. It was easy to generate a shape file from the CAD files, but some metadata was lost in the conversion. The individual building identifiers were lost. Only the coordinates were retained. When this file was used to render the map, it looked like a bunch of spaghetti as everything was connected by a continuous line. An attempt was made to assign sequential coordinates to individual buildings. This approach did generate individual buildings but the shape file had over 2,500 rows of coordinates. This process would take a very long time and not be as exact as it should be.

The next attempt was a manual approach. A plain version of the campus map with only building outlines was printed in black and white on 8.5" by 11" paper. This was taped to a window and a piece of quadrille paper was taped over it. The buildings were traced and coordinates generated from an axis added to the quadrille paper. These coordinates, along with an added building identifier, were entered into am Excel workbook and imported into SAS. The dataset rendered a map that was recognizable as the campus. It lacked a lot of detail but proportions and layout were consistent with the map it was drawn from. While in no way was this an ideal solution, it provided a way to move forward whereby the team could illustrate what this type of visualization could do. The "old school" map is shown below.



After presenting the old school generated campus map in a preliminary animation, funding was eventually secured to have a student worker generate a shape file from a more detailed campus map. This new map contained detailed outlines of the campus buildings, including residence halls and apartments, as well as parking

lots for additional perspective. The student worker created a true shape file (with a .shp extension) for the project. PROC MAPIMPORT with no options was used to create a dataset to be used in future visualizations.

```
PROC MAPIMPORT OUT= m.rit_map1 DATAFILE='I:\bldings_v01\footprints_v01.shp';
run;
```

ROLLING-UP THE DATA

The time dataset was merged to both the schedule and wireless data. The merge was purposefully a Cartesian join. This would ensure that all time combinations could be evaluated. This may not have been the most eloquent approach but it worked for the project.

```
proc sql;
create table alltime as
select *
from tb10, schedcollege;
quit;
```

Each class time was compared to the time variable from the 10-minute interval dataset and if it overlapped the observation was flagged. The same occurred for the wireless sessions. The comparison here was based on when the session was initiated and its duration.

The flagged observations were used to create a new dataset. This new dataset was aggregated by counting records for each building, day of the week, and time interval combination. This would generate the occupancy values to be displayed on the map. The interest was not in individuals, but rather how many people were in a given building at a given time.

CREATING THE MAP

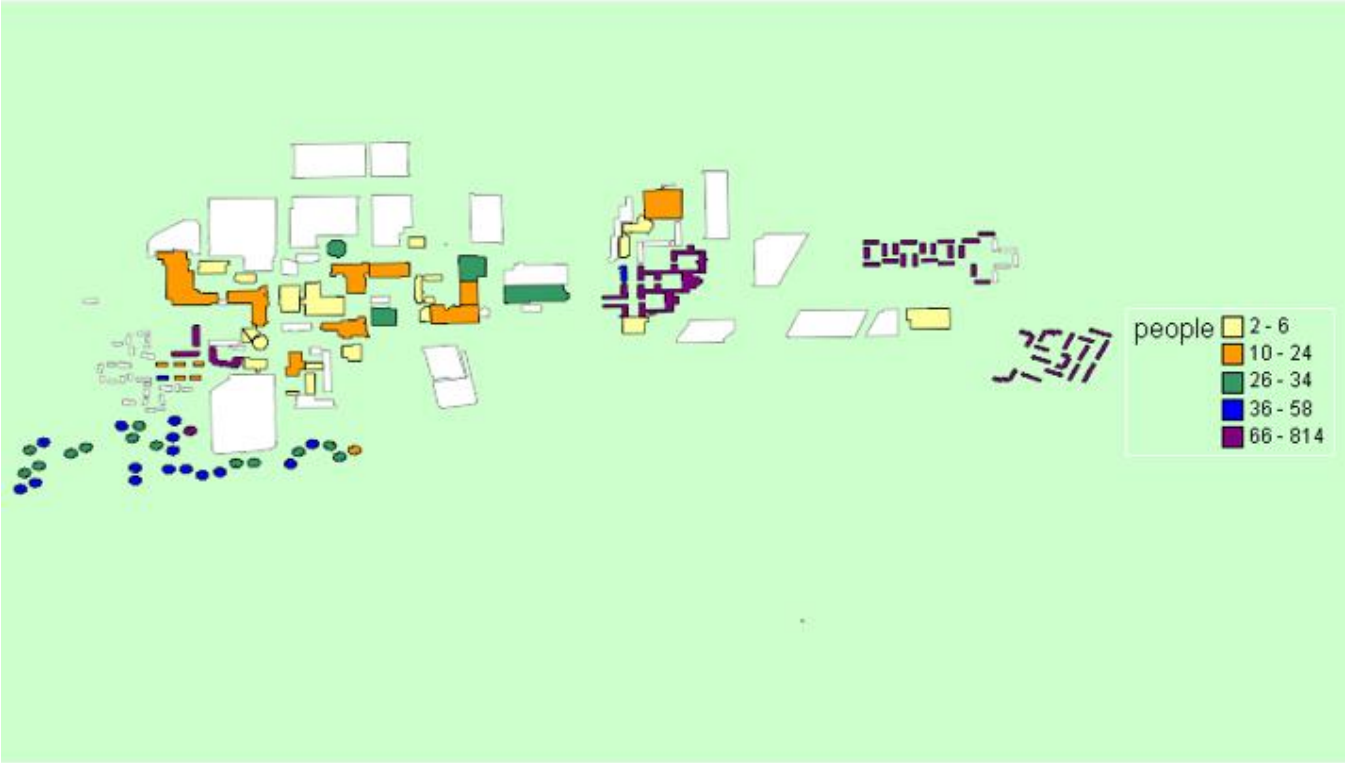
The first map visualization was generated using PROC GMAP.

```
proc gmap data = schedday10college map = m.rit_map1;
id bldg;
choro amt;
run;
quit;
```

ID specifies the variable that delineates the individual shapes. The ID variable can be either character or numeric but must be the same in the data and map datasets. CHORO in combination with the variable amt states which variable contains the values used to fill the map shapes.

The QUIT statement is necessary after the run since PROC GMAP is one of the procedures that supports RUN-group processing. This means is that the procedure will execute but continue to run until either a QUIT statement or step boundary like a datastep is encountered.

The map generated with the new shapefile is shown below. Categories of the count of students were generated in order to have a consistent color scheme across the different time frames. Without this, each map generated for the animation would create its own scale and make it difficult to compare across time. The scales would change since the overall count of occupancy differed widely based on the time of day. There are fewer people active at 3:00 AM compared to 10:00 AM. The new map is a marked improvement over the old school version!



MAKING AN ANIMATION

The process for creating the animation was adapted from a paper presented at SUGI 29 which can be found in the references. Generally, the approach centers around clearing then setting the graphics options, designating a device driver, developing a loop over which to generate maps, and finally setting up the GMAP procedure to generate the map you want.

The options used in the program are below.

```
goptions
reset=all
device=gifanim
gsfname=animout
gsfmode=replace
gunit=pct
ftext='Tahoma'
ctext=blue
border
iteration=0
delay=150
;
```

Some options are specific to the animation: device, gsfname, gsfmode, iteration, delay. Other options are for the appearance of the map: ftext, ctext, and border. Additional options relating to the appearance of the map can be added if desired.

The loop is necessary to generate a map for each time interval you would like. In this case, every 10 minutes over a 3-day period. Each map is rendered and stored to the gif to create the animation. The delay option sets how long each map is displayed. The example code below generates macro variables for each time interval as well as for each college. The first section of code is from outside of the macro loop and the other is within the macro loop.

```
proc sql noprint;
select count(distinct college) into :coll
from schedday10college;
quit;
```

The macro variable &coll is created in the code above to set how many times the outer loop iterates – once for each college.

```
data  null_;
set colls;
if  _n_ = &j;
call symput('colluse', college);
run;

proc sql noprint;
select count(distinct dow||put(time, time.)) into :daytime
from schedday10college
where college = "&colluse";
quit;
```

Above, the macro variable &colluse is created based on the row of the dataset colls which contains a list of each college. The macro variable &daytime is generated as the count of each day time combination used in the inner lop to generate that many distinct maps.

Within the loops is the procedure to generate the individual maps. The inner loop is shown below in its entirety.

```
%macro manymaps;

%do i=1 %to &daytime;
%if &i eq 2 %then goptions gsfmode=append;;
%if &i eq &daytime %then goptions gepilog='3B'x;;


data  null ;
set testit;
if  _n_ = &i;
call symput('dow', dow);
call symput('stime', time);
call symput('stimef', put(time, tod8.));
run;


Title "Students in Class &dow Starting at &stimef with a Major in &colluse";

proc gmap data = schedday10college map = campusmap all;
id bldg;
where dow = "&dow" and time = &stime and college ="&colluse";
choro enroll / cempty=red statistic=sum;
run;
quit;

%end;
%mend;
```

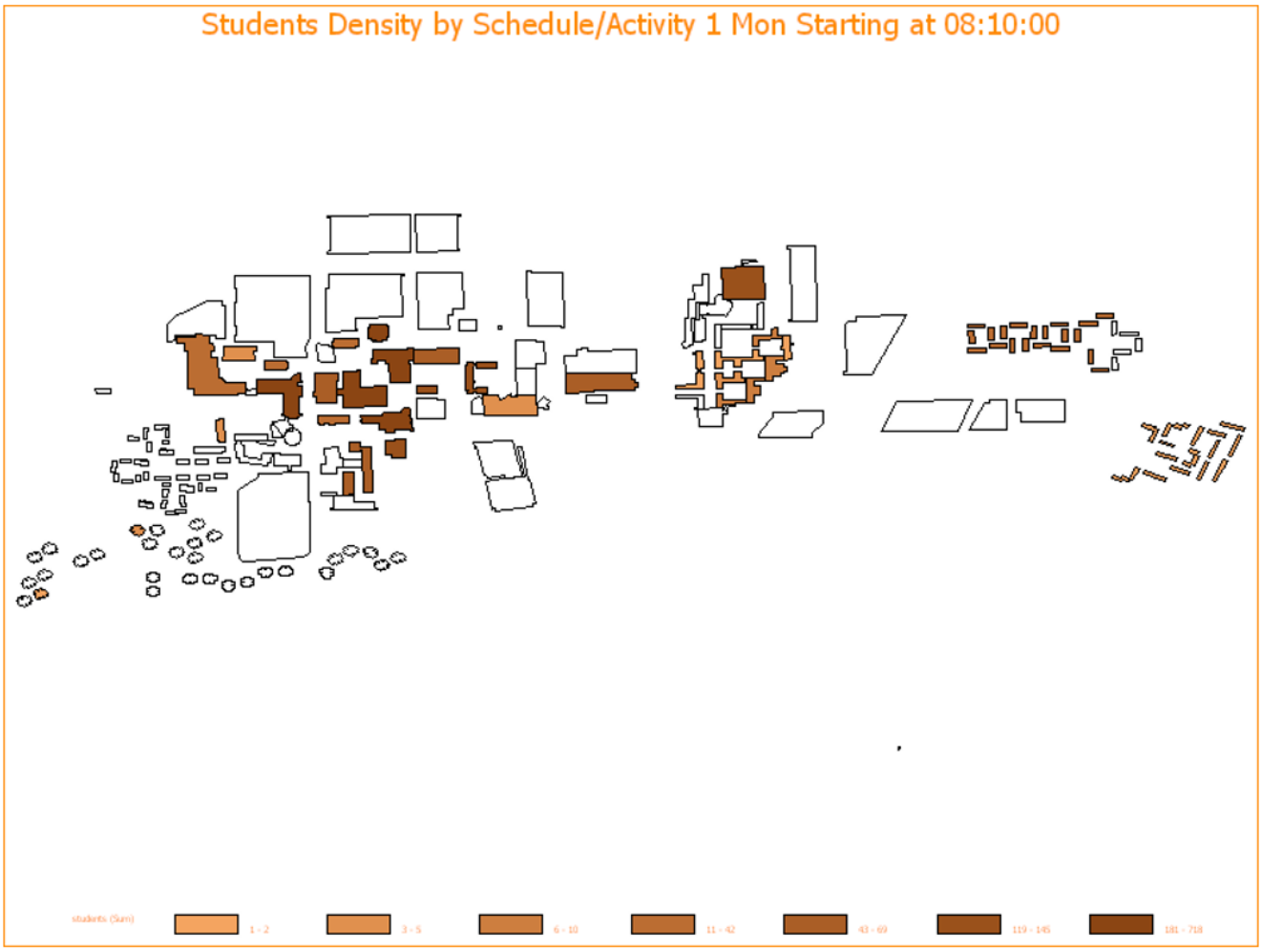
If the loop is not on the first iteration, the goption gsfmode needs to be changed to replace. Otherwise each map will just replace the other one and there will be no animation.

Additional macro variables are generated and used to create titles for each map in the animation. This will provide context to the viewer. They are also used in the sub setting of the data used in the map.

A few notes regarding the options used in PROC GMAP. The ALL option on the PROC GAMAP statement will generate the outline of each building regardless of whether or not there is an occupancy count. Without this option, only buildings with an occupancy will be rendered. This can make the map sparse and hard to interpret.

CEMPTY= on the CHORO statement defines the outline color for building without occupancy. This was set to show specifically that the building was unoccupied.

One of the maps from the animation is below. Note, this particular version of the animation did not use the CEMPTY= option and the color scheme was set differently.



Other examples of animations generated by SAS are on the website cited in the references. This site was used to evaluate other possibilities and to look at additional sample code. There is a wide range of possibilities for this type of visualization.

ADDITIONAL PROC MAPIMPORT OPTIONS

As work continued on this project, additional options for PROC MAPIMPORT were learned. The issue with the converted CAD files was that the shape identifier was lost as mentioned earlier. This made the entire map look like a giant dot-to-dot. There visualization generated using PROC GMAP was one continuous line. These other options were not available when earlier proceedings papers and resources were referenced.

These options solve the spaghetti map problem when added to the PROC. Adding the CONTENTS option puts more information in the SAS log to assist in evaluating the imported datafile. The CREATE_ID_ option solved the spaghetti issue and rendered individual buildings. These options go on the PROC MAPIMPORT statement.

```
proc mapimport datafile='C:\FMS\bldgs_polygon.shp' out = fmsbldgpoly  contents
create_id_;
run;
```

The log contains additional information from the CONTENTS option which is highlighted below.

```
86
87  proc mapimport datafile='C:\FMS\bldgs_polygon.shp' out = fmsbldgpoly contents create_id_;

      Contents of the Input file

   Input filename: C:\FMS\bldgs_polygon.shp

   List of Fields and Attributes

   # Field      Type Width Decimals

   1 BLDGS_p_ID  NUM    5    0
88  run;

Number of Fields:    1
Number of Records:   315
NOTE: The data set WORK.FMSBLDGPOLY has 6232 observations and 5 variables.
NOTE: PROCEDURE MAPIMPORT used (Total process time):
    real time       10.68 seconds
    cpu time        0.06 seconds
```

The code below will generate the map using the newly created ID as a value to set the color gradient to. This allows you to see how the visualization looks right after import to make sure it is as expected.

```
proc gmap map=fmsbldgpoly data=fmsbldgpoly all;
id  id ;
choro _id_ / levels=10;
run;
quit;
```

The one challenge with this approach is that the ID created by the CREATE_ID_ option does not match the existing building numbers that were used in the earlier PROC GMAP on the ID statement (id bldg; in the code under “Creating the Map” above). Therefore, the variable bldg containing the proper building number would need to be added to the fmsbldgpoly dataset in order to match the schedday10college dataset. Then the amt variable could be used to color in the shapes based on the date.

CONCLUSION

SAS is the perfect tool to create this type of visualization. It handles everything from data access and preparation right through creating the animation. Large and small data can be combined easily and from multiple sources. Any type of map that you might need can be used in your program or you can import any .shp file. You can even create your own map using x and y coordinates. Animation is another benefit with this approach. Instead of

viewing a series of individual maps, they can be displayed for you in series. This can help to spot trends and allow you to see things over time more easily. This is just another method do build knowledge from data.

APPENDIX

A loner section of sample code which includes the summarization of the data and generation of the animation is in the Appendix.

REFERENCES

Allison, Robert. "Robert Allison's GIF Animations (#27)." Link accessed 08/31/2016. Available at http://robslink.com/SAS/democd27/aaaindex.htm.

Hadden, Louise S., "Red Rover, Red Rover, Send Data Right Over: Exploring External Geographic Data Sources with SAS®." MWSUG 2016 Proceedings, Cincinnati, OH. Available at http://www.mwsug.org/proceedings/2016/DV/MWSUG-2016-DV05.pdf.

Zdeb, Mike. "Creating Maps with SAS/GRAPH® - Drill Downs, Pop-Ups, and Animation." SUGI 29 Proceedings, Montreal, Canada. Available at http://www2.sas.com/proceedings/sugi29/120-29.pdf.

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:
Stephanie R. Thompson
Datamum
Email: stephanie @datamum.com
Web: http://www.datamum.com

APPENDIX

```
PROC IMPORT OUT= tb10
            DATAFILE= "I:\time by 10.xlsx"
            DBMS=EXCEL REPLACE;
     SHEET="Sheet2$";
     GETNAMES=YES;
     MIXED=NO;
     SCANTEXT=YES;
     USEDATE=YES;
     SCANTIME=YES;
RUN;

data tb10;
set tb10;
format time timeampm11. ;
run;

proc sql;
create table schedcollege as
select college, bldg, dow, start, end, count(*) as enroll
from schedday college
group by college, bldg, dow, start, end;
quit;

proc sql;
create table alltime as
select *
from tb10, schedcollege;
quit;

data alltime;
set alltime;
if time ge start and time lt end then flag = 'Y';
else flag = 'N';
run;

data schedtime;
set alltime;
if flag = 'Y' then output;
run;

proc sql;
create table schedday10college as
select college, bldg as charbldg, dow, time, sum(enroll) as enroll
from schedtime
group by college, bldg, dow, time
order by college, bldg, dow, time;
quit;

data schedday10college;
set schedday10college;
if charbldg = '007A' then bldg = 7.1;
else if charbldg = '007B' then bldg = 7.2;
else bldg = put(charbldg, 8.);
run;
```

```
*** reset all graphics options to default values, select the GIFANIM device driver;
*** specify other parameters for the GIF file including ITERATION and DELAY
(animation controls);
goptions
reset=all
device=gifanim
gsfname=animout
gsfmode=replace
gunit=pct
ftext='Tahoma'
ctext=blue
border
iteration=0
delay=150
;
*** ;

proc sql noprint;
select count(distinct college) into :coll
from schedday10college;
quit;

proc sql;
create table colls as
select distinct college
from schedday10college;
quit;

%macro manycols;

%do j=1 %to &coll;

goptions
reset=all
device=gifanim
gsfname=animout
gsfmode=replace
gunit=pct
ftext='Tahoma'
ctext=blue
border
iteration=0
delay=150
;


data  null_;
set colls;
if   n  = &j;
call symput('colluse', college);
run;

proc sql noprint;
select count(distinct dow||put(time, time.)) into :daytime
from schedday10college
where college = "&colluse";
```

```
quit;

proc sql;
create table testit as
select distinct dow, time
from schedday10college
where college = "&colluse"
order by dow, time;
quit;


%put Day and Time Combinations --- &daytime --- for &colluse;

pattern v=e c=blue;

%macro manymaps;

%do i=1 %to &daytime;
%if &i eq 2 %then goptions gsfmode=append;;
%if &i eq &daytime %then goptions gepilog='3B'x;;


data  null ;
set testit;
if   n  = &i;
call symput('dow', dow);
call symput('stime', time);
call symput('stimef', put(time, tod8.));
run;


Title "Students in Class &dow Starting at &stimef with a Major in &colluse";

proc gmap data = schedday10college map = campusmap all;
id bldg;
where dow = "&dow" and time = &stime and college ="&colluse";
choro enroll / cempty=red statistic=sum;
run;
quit;

%end;
%mend;

%let colluset = %trim(&colluse);

filename animout "I:\class_time10_&colluset..gif";
%manymaps;

%end;
%mend;

%manycols;
```