Paper 8801-2016

Do It Yourself (DIY) Data: Creating a Searchable Data Set of Available Classrooms using SAS® Enterprise BI Server

Nicole E. Jagusztyn, Hillsborough Community College

ABSTRACT

At a community college, there was a need for college employees to quickly and easily find available classroom time slots for the purposes of course scheduling. The existing method was time-consuming and inefficient, and there were no available IT resources to implement a solution. The Office of Institutional Research, which had already been delivering reports using SAS® Enterprise BI Server, created a report called “Find an Open Room” to fill the need. By combining SAS® programming techniques, a scheduled SAS® Enterprise Guide® project, and a SAS® Web Report Studio report delivered within the SAS® Information Delivery Portal, a report was created that allowed college users to search for available time slots.

INTRODUCTION

Public postsecondary institutions, especially community colleges, can often find themselves understaffed and underfunded, particularly in the areas on Information Technology. Implementing new technology at a pace to meet the expectations of faculty, staff and students is often difficult given available resources, and can take years to become fully available. However, Institutional Research offices who use SAS® may be able to quickly fill a need using their available tools. This was the case at an urban community college in Tampa, FL. This paper will describe how to use SAS® software to create an interactive report that allows users to search for available classroom time slots for the purposes of course scheduling.

THE PROBLEM

While waiting for the Information Technology office to implement a robust software solution to help with course scheduling, deans and program managers at a community college were left with no efficient way to find available rooms in which to schedule courses. The only available tool was a report delivered by the college’s Enterprise Resource Planning (ERP) software which forced employees to check the schedule of each classroom one-by-one in search of an available time slot. A dean or program manager may have to search through the schedules of dozens of individual classrooms before finding an available time slot. Upon learning of this, the Institutional Research office stepped in to develop a report using SAS® that would provide deans and program managers with a list of available classrooms that met their search criteria.

The first problem to address was the creation of the data. Although data exists within the ERP on when and where current classes are scheduled, there is no existing data on the available time slots *between* scheduled classes. This data needed to be created. Further, it would need to be updated each morning on an automated schedule. The second problem was the flexibility of the programming that would be needed to accommodate the complexity in the college’s course scheduling. There was no uniformity in the length of individual class meetings or in the start or end dates of a course sections, and the college did not have block scheduling. The programming would need to account for all of this irregularity. Finally, the report would need an easily understandable visualization and a platform on which to deliver it. There were few resources available to train individual users, so a fairly straight-forward visualization was needed.

THE PLAN

The Institutional Research office developed a plan to create a new report using SAS®, which would be called “Find an Open Room”:

1. Create a SAS® Enterprise Guide® project that includes a SAS® program designed to extract a

simple dataset out of the college's ERP, including data regarding the currently scheduled classes.

2. Schedule this project to run every morning so that the data is updated each day.
3. Create another SAS® program designed to create a dataset of the available classrooms and time slots.
4. Create a report using SAS® Web Report Studio, within SAS® Enterprise BI Server, delivered using SAS® Information Delivery Portal to allow college users to easily find available classrooms for the purposes of course scheduling.

ASSEMBLING THE DATASET

The dataset needed to start is fairly simple. Information about the current schedule of classes is needed, including the building and room where the class is held, the start and end date of the class, the days of the week when the class is held, and the start and end time of the class. If a particular class has an irregular schedule, for example, if the course meets from 1:00 pm to 2:00pm on Monday but 3:00pm to 4:00pm on Wednesday, this class will be duplicated in the data with one row for each instance of a course pattern. It is important to capture the complete course schedule. Additionally, an academic term variable is needed if the intent is to show availability for multiple upcoming terms. All following code will be run iteratively for each term, with data from multiple terms combined into a final data table. Display 1 contains the format of the data that was used in this example, although other data formats could be used.

	term	section_name	bldg	room	start_date	end_date	days	start_time	end_time
1	15/FA	SCY-0051-68696	YPST	332	09/14/15	09/21/15	MTWTH	18:00	22:00
2	15/FA	SCY-0052-68701	YPST	314	11/10/15	11/13/15	TWTHF	18:00	22:00
3	15/FA	SCY-0052-68702	YPST	332	09/22/15	09/25/15	TWTHF	18:00	22:00
4	15/FA	CJD-8800-68707	YPST	332	10/26/15	10/30/15	MTWTH	8:00	17:00
5	15/FA	CJD-8800-68709	YPST	332	11/30/15	12/04/15	MTWTH	8:00	17:00
6	15/FA	CJD-8800-68710	YPST	332	11/16/15	11/20/15	MTWTH	8:00	17:00
7	15/FA	CJD-8800-68712	YPST	332	10/12/15	10/16/15	MTWTH	8:00	17:00
8	15/FA	CJD-8800-68713	YPST	332	09/28/15	10/02/15	MTWTH	8:00	17:00
9	15/FA	CJD-8800-68715	YPST	332	08/31/15	09/04/15	MTWTH	8:00	17:00
10	15/FA	CJD-8800-68716	YPST	332	08/17/15	08/21/15	MTWTH	8:00	17:00
11	15/FA	CJD-8801-68732	YPST	332	09/17/15	09/18/15	THF	8:00	17:00
12	15/FA	CJD-8801-68733	YPST	332	09/03/15	09/04/15	THF	8:00	17:00
13	15/FA	CJD-8801-68734	YPST	332	08/20/15	08/21/15	THF	8:00	17:00

Display 1. Example of Input Dataset

Next, manipulate the dataset so that there is a row of data for every day of the week in which a class meets, and assign a numerical value to each day of the week. In the current example, this was accomplished by searching for a day within each class's schedule, writing out datasets for each day of the week, then setting the datasets back together. The final dataset is shown in Display 2. Notice that classes that meet on multiple days of the week now have a row of data for every meeting day.

	section_name	start_date	end_date	start_time	end_time	days	day	weekdaynum	bldgroom
1	ACG-2021-83346	05/16/16	06/22/16	18:00	21:30	MW	Monday	2	YPST-216
2	ACG-2021-83346	05/16/16	06/22/16	18:00	21:30	MW	Wednesday	4	YPST-216
3	ACG-2021-83347	06/28/16	08/04/16	18:00	21:20	TTH	Thursday	5	YPST-216
4	ACG-2021-83347	06/28/16	08/04/16	18:00	21:20	TTH	Tuesday	3	YPST-216
5	ACG-2021-83763	05/17/16	06/23/16	11:30	14:50	TTH	Thursday	5	BADM-232
6	ACG-2021-83763	05/17/16	06/23/16	11:30	14:50	TTH	Tuesday	3	BADM-232
7	ACG-2021-83764	05/16/16	06/24/16	11:30	14:50	MW	Monday	2	DSSC-204
8	ACG-2021-83764	05/16/16	06/24/16	11:30	14:50	MW	Wednesday	4	DSSC-204
9	ACG-2021-83765	05/16/16	06/24/16	18:30	21:50	MW	Monday	2	DTEC-425
10	ACG-2021-83765	05/16/16	06/24/16	18:30	21:50	MW	Wednesday	4	DTEC-425
11	ACG-2021-83766	06/27/16	08/08/16	11:30	14:50	MW	Monday	2	DTEC-409
12	ACG-2021-83766	06/27/16	08/08/16	11:30	14:50	MW	Wednesday	4	DTEC-409
13	ACG-2071-83351	06/28/16	08/04/16	11:30	14:50	TTH	Thursday	5	YPST-216
14	ACG-2071-83351	06/28/16	08/04/16	11:30	14:50	TTH	Tuesday	3	YPST-216

Display 2. Example of Formatted Dataset

Next, run an ARRAY that will fill in a value of 1 for every occupied day during the term. The ARRAY will need to span from the first possible day of the term to the last possible day of the term, although the example code below is only showing four days as an illustration. It is easy to cut and paste the values for the ARRAY from a spreadsheet that has been formatted with the appropriate date values, and concatenated with the extra characters needed for the date formatting. The ARRAY shown is determining if the date value from daysx[i] falls within the start and end date of the class and if the numerical day value of the date matches the numerical day value of the class meeting day. If both conditions are matched, the value of vars[i] becomes 1. For example, if a class meets on Mondays, then every Monday between the start date and the end date of the class will have a value of 1. Notice that the variables in vars[i] look like dates, and match corresponding date values in daysx[i]. Later, these data elements will be turned into date values. Once the DATA step containing the ARRAY completes, it is sorted and transposed:

```
data daytest;
   set week;
   array daysx[4] ('11JAN2016'D '12JAN2016'D '13JAN2016'D '14JAN2016');
   array vars[4]  D011116 D011216 D011316 D011416;
   do i = 1 to dim(daysx);
   if start_date <= daysx[i] <= end_date and weekday(daysx[i]) =
   weekdaynum then vars[i] = 1;
   end;
run;

proc sort data = daytest; by bldgroom day start_time end_time; run;

proc transpose data = daytest out = daytest2;
   by bldgroom day start_time end_time;
   var D011116 D011216 D011316 D011416;
run;
```

Next, run a second ARRAY to create five minute increments from the earliest possible start time to the latest possible end time. The day spans from 7:00am to 11:00pm at the current college, although the sample code is only showing the first five increments as an illustration. Again, values could be cut and paste from a formatted spreadsheet to populate the array. Similar to above, the DATA step is determining if the time value falls within the start and end time of the class, and if this condition is met, codes[i] is given a value of 1. Also, similar to above, the variables in codes[i] correspond to the time vales in times[i], and will later be converted into time values. Once the DATA step containing the ARRAY completes, a PROC MEANS is run on a sorted dataset to combine all of the instances in which a room is occupied during a particular day-time. Note that in the following example code, the data element ‘datevar’ is a re-

named version of the data element '_NAME_' that was produced as a result of the PROC TRANSPOSE above:

```
data timespan;
   set daytest2;
   array times[5] ('07:00't '07:05't '07:10't '07:15't '07:20't);
   array codes[5]   T0700 T0705 T0710 T0715 T0720;
   do i = 1 to dim(times);
   if start_time <= times[i] <= end_time then codes[i] = 1;
   end;
run;

proc sort data = timespan; by bldgroom datevar day; run;

proc means data = timespan noprint;
   var  T0700 T0705 T0710 T0715 T0720 ;
   by bldgroom datevar day;
   output out = timespansum SUM(T0700 T0705 T0710 T0715 T0720) = T0700
   T0705 T0710 T0715 T0720;
run;
```

Next, a sorted dataset is reverse transposed. A subsequent DATA step determines the times in which each room is busy and open:

```
proc sort data = timespansum; by bldgroom datevar day; run;

proc transpose data = timespansum out = timespan2;
   by bldgroom datevar day;
   var T0700 T0705 T0710 T0715 T0720;
run;

data timespan3;
   set timespan2;
   if COL1 ge 1 then Status = 'BUSY';
   if COL1 = . then Status = 'OPEN';
run;
```

The contents of the dataset timespan3 will be very large. This dataset will contain, for each classroom, a label of "BUSY" or "OPEN" for every five minute increment of every day from the start of the term to the end of the term. The next steps will identify the exact start and end of every "BUSY" or "OPEN" block of time by classroom. Because the previous DATA steps created array variables that matched the time slots (e.g. 7:00am = T0700), we can identify the first and last occurrence of every busy and open block of time, then convert the variable name back to a time variable using functions. In the DATA step openrooms, the first and last instance of every block of time is identified, the between times are deleted, and a lag function gets the start time on the same row of data as the end time. The DATA step converttimes uses character functions to create a formatted time data element from the former ARRAY variables:

```
data openrooms;
   set timespan3;
   by notsorted bldgroom datevar day Status;
   if first.Status = 1 then st_time2 =
   CATS((SUBSTR(_NAME_,2,2)),':',(SUBSTR(_NAME_,4,2)));
   if last.Status = 1 then ed_time =
   CATS((SUBSTR(_NAME_,2,2)),':',(SUBSTR(_NAME_,4,2)));
   if first.Status = 0 and last.Status = 0 then delete;
   st_time = lag(st_time2);
run;
```

```
data converttimes;
   FORMAT start_time end_time TIME5. date MMDDYY8.;
   set openrooms;
   start_time = input(st_time,TIME5.);
   end_time = input(ed_time,TIME5.);
    date =
    input(CATS((SUBSTR(datevar,2,2)),'/',(SUBSTR(datevar,4,2)),'/',(SUBSTR(
    datevar,6,2))),MMDDYY8.);
    if st_time2 ne '' then delete;
    drop _NAME_ COL1 st_time2 ed_time st_time;
run;
```

Next, a similar set of DATA steps are performed to determine the start and end dates for room availability. This step is important in the current example because start and end dates for class sections do not follow a defined schedule at the current college. Similar to the time array, data elements were created in the ARRAY that mimic dates (e.g. January 11, 2016 = D011116), and can be converted to dates using functions and formatting:

```
proc sort data = converttimes; by bldgroom day start_time status; run;

data altdays;
   set converttimes;
   by notsorted bldgroom day start_time end_time status;
   if first.Status = 1 then st_date2 =
   CATS((SUBSTR(datevar,2,2)),'/',(SUBSTR(datevar,4,2)),'/',(SUBSTR(dateva
   r,6,2)));
   if last.Status = 1 then end_date2 =
   CATS((SUBSTR(datevar,2,2)),'/',(SUBSTR(datevar,4,2)),'/',(SUBSTR(dateva
   r,6,2)));
   if first.Status = 0 and last.Status = 0 then delete;
   st_date = lag(st_date2);
   if st_date2 ne '' then delete;
run;

data convertdates;
   FORMAT start_date end_date MMDDYY8.;
   set altdays;
   start_date = input(st_date,MMDDYY8.);
   end_date = input(end_date2,MMDDYY8.);
   drop st_date2 end_date2 st_date datevar;
run;
```

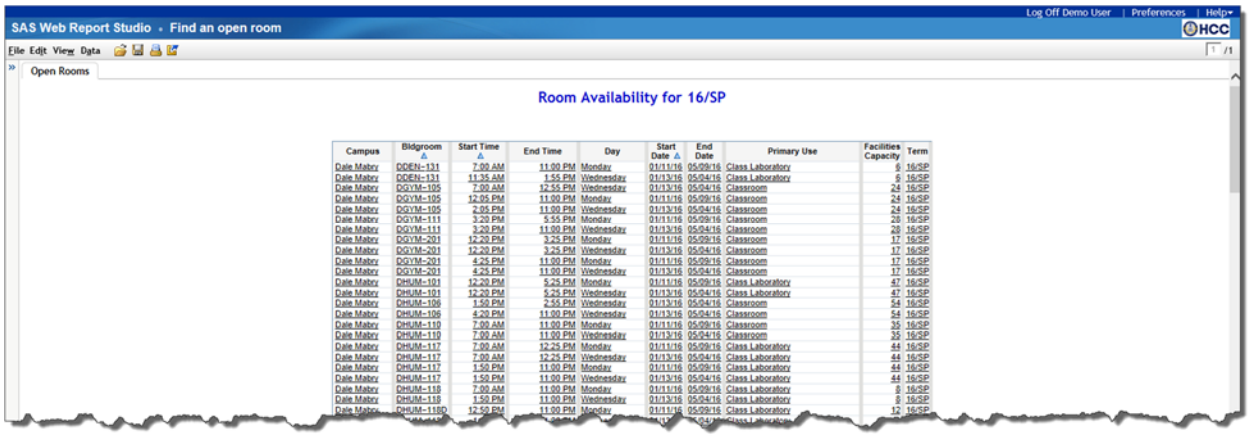
As a final step, some cleaning of the dataset may be needed before using it to create a visualization. For example, your users may expect dates or times to be presented in a particular format. Additionally, you may want to eliminate blocks of time that are too short in which to schedule a class. In the current example, any block of time shorter than 45 minutes was eliminated from the final dataset. As a final step, add characteristics of the rooms and time slots by adding data elements to identify the type of time slot (e.g. morning, afternoon), the type of room (e.g. lecture hall, lab), or any other desired characteristics (e.g. campus, room capacity). These will be later used in creating filters for the report and adding columns to the output table.

ASSEMBLING THE VISUALIZATION

Once the final dataset is complete, save it in Metadata by adding it to a library in SAS® Management Console. Then, create an information map based on the data in SAS® Information Map Studio. Several filters will need to be created including all of the search criteria that a user would need to search for an

open room. The current report contains filters for upcoming term, campus, building, day(s) of the week, time of day (i.e. morning, afternoon, evening), room capacity, and classroom type (e.g. lecture hall, lab).

Using the information map, make a new report using SAS® Web Report Studio. While the design of the report can vary, the main visualization should be a list table with the relevant data elements in view. Add the relevant filters to the report so a user can choose their parameters before viewing the output. For the current community college, all users view reports by first entering SAS® Information Delivery Portal, although this is not necessary in all instances. Display 3 shows a screenshot of the report output from the current report, displaying all of the available rooms based on the selected filters.



Display 3. Example Report Results

CONCLUSION

When IT resources are in short supply, individuals in Institutional Research can quickly fill a need using the tools provided by SAS®. By combining a scheduled SAS® Enterprise Guide® report, SAS® code, and SAS® Enterprise BI Server, a useful report was created allowing college deans and program managers to quickly find available classrooms for the purposes of course scheduling.

ACKNOWLEDGMENTS

The author extends her heartfelt thanks and gratitude to Paul Nagy, Special Assistant to the President in Strategic Planning and Analysis, for his vision regarding the use of SAS® in supporting the strategic plan of the college, to Elizabeth Steinhardt Stewart, Director of Institutional Research and Grants, for her continued support of my work, and to Newton Beardsley, Institutional Research Officer, for his enthusiastic collaboration and depth of knowledge that have enabled us to create amazing products!

CONTACT INFORMATION

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

Nicole E. Jagusztyn
Hillsborough Community College
813-253-7090
njagusztyn@hccfl.edu