

Automating Manual Processes with SAS®: A Practical Example from Higher Education

Smith, Kelly D., & Davis, Elizabeth S., University of North Carolina School of the Arts

ABSTRACT

Doing more with less; improving efficiency; standardizing a process. Automation offers an approach to help Institutional Research offices meet all these goals. In this session, learn about the process one IR office used to automate a process previously done manually through Excel. The automation process utilized SAS® 9.4 and Enterprise Guide. Takeaways include the process framework, useful PROC procedures, and lessons learned that you can apply to your own automation project.

INTRODUCTION

Institutional Research, Institutional Effectiveness, Planning and Evaluation. These are three of the many different names for the department tasked with meeting state and federal reporting guidelines, responding to internal and external data requests, and generally filling a multitude of institutional needs, often with limited staffing. Automating a manual process permits a limited staff to increase productivity and to focus on more time-sensitive requests.

Enrollment projection is a high-priority data request at a small, arts-oriented campus. As an ongoing request and extensive manual process, enrollment projection is a good candidate for automation. The SAS Educational Analytical Suite includes the Base SAS and Enterprise Guide platforms. This article describes the early steps in an ongoing project to automate enrollment projection at the campus.

COMPILING INFORMATION

Key information was collected from a variety of sources for the automation project.

Student admissions, enrollment, retention, and graduation data was obtained from a state level database. Key variables were defined and identified by table location. When equivalent data was present in separate tables, differences in variable names was noted and highlighted.

A detailed process map of the current process was constructed. Decision rationales for key steps and decision points were included; a general visual of the process map is displayed in Figure 1. The process map guided the automation process.

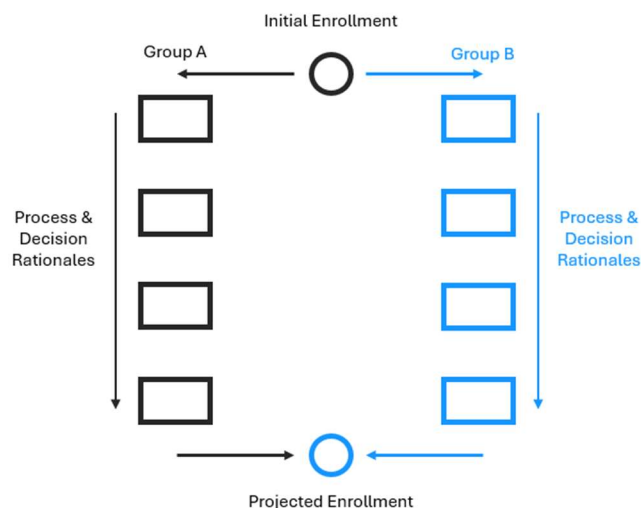


Figure 1. Visual of process map style used for project.

Final reports from previous enrollment projections were collected for use as test data through the automation process.

PROJECT FRAMEWORK

Automating a multi-step project is a large undertaking; therefore, it was decided to break the project down into more manageable steps.

1. Enrollment calculations.
 - a. Projected enrollment.
 - b. Adjustments based on actual enrollment.
2. Retention calculations.
 - a. Projected retention based on weighted, three-year average.
 - b. Adjustments based on actual retention.
3. Data from state database.
 - a. Static datasets.
 - b. Dynamic data extraction from Amazon Redshift using PROC SQL.
4. Export final data.
 - a. Statis reporting by PROC PRINT or PROC EXPORT to Excel.
 - b. Dynamic export to Power BI.

Splitting the project into distinct sections made the coding (and debugging) simpler. Starting with the final calculations helped identify macrovariables that needed to be developed in earlier sections of the code. The enrollment projection process is applied to three groups of students; data for the largest group of students was used in code development. By starting with the largest group of students, potential complications in calculations due to small numbers were minimized.

USEFUL SAS TECHNIQUES

The depth and breadth of SAS provides multiple coding options for a project. A dynamic coding approach was chosen to minimize the amount of initial input and updating required from an analyst. A summary of useful SAS functions and processes follows.

WORKING WITH VARIABLES

The %EVAL function was used to build new macrovariables based on the value of an initial input macrovariable:

```
%LET yr=2020;  
%LET yr11=%EVAL(&yr + 1); /* yr11 = 2021 */  
%LET yr01=%EVAL(&yr - 1); /* yr01 = 2019 */
```

The CATX function was used to build new variables from current variables. CATX was the function of choice because leading and trailing blanks are removed prior to concatenation; additionally, CATX applies a delimiter of choice between concatenated values:

```
Group=CATX(delimiter, item 1, item 2, item 3);  
Group=CATX("_", Institution, Residency, Major);  
/* Institution_Residency_Major */
```

The CALL SYMPUTX routine was used to build macrovariables from the values of a data set variable. CALL SYMPUTX was chosen because it uses left-justification and removes trailing blanks from both arguments:

```
CALL SYMPUTX(macrovariable, value, symbol-table);
CALL SYMPUTX(Category, Retention, G);
/* &category = retention value */
```

The CEIL function was used to control rounding in projected retention and enrollment calculations. CEIL returns the value as the smallest integer equal or greater to the original value:

```
Returning=CEIL(Enrollment * RetentionRate);
/* Returning=CEIL(125*0.95) → Returning=CEIL(118.75) → Returning=119 */
```

To avoid a later many-to-one MERGE, student records across multiple terms were collapsed into one record using an ARRAY statement:

```
DATA Records_array;
RETAIN Student Tm01-Tm04;
  ARRAY Terms (4) $50. Tm01-Tm04; /* describes array */
  SET Records;
  BY Student;
  IF FIRST.Student THEN DO;
    i = 1;
    DO j = 1 to 4;
      Terms(j) = " "; /* creates empty array */
    END;
    END;
    Terms(i) = term_name; /* fills empty array */
  IF LAST.Student THEN OUTPUT;
  i + 1;
RUN;
```

IF THEN statements were used to control a MERGE and also to determine the appropriate values for new variables:

```
IF A=1 AND tm01="Fall 2020" THEN Fall_One = 1; /* Yes = 1 */
ELSE Fall_One = 0; /* No = 0 */

IF B=1 AND completion="Fall 2020" THEN F1_Compl = 1; /* Yes = 1 */
ELSE F1_Compl = 0; /* No = 0 */

IF Fall_One = 1 AND F1_Compl = 0 THEN Poss_Spr_Stu = 1;
ELSE Poss_Spr_Stu = 0; /* No = 0 */
```

MACROS

MACRO routines with positional parameters were used to minimize code and standardize data processing / calculations across multiple academic years:

```
%MACRO Retention(Fall, Spring, Summer, Fall2);

%Retention(Fall2020, Spring2021, Summer2021, Fall2021);
```

PROCEDURES

PROC SUMMARY was used to calculate the number of students by analytical group. The AUTONAME function was used to standardize and simplify the names of outcome variables:

```
PROC SUMMARY DATA=Students;
BY Group;
VAR Fall Spring Summer Fall2;
OUTPUT OUT=StudentCounts SUM= / AUTONAME;
RUN;
```

PROC TRANSPOSE was used to rearrange multiple years of data and permit calculation of the weighted three-year retention average:

```
PROC TRANSPOSE DATA=StudentCounts3yr OUT=ThreeYrCounts_flip;
BY Group;
ID StartTerm;
VAR Fall2Fall;
RUN;
```

PROC PRINT and PROC TABULATE were used to build internal check point reports throughout the code. PROC PRINT and PROC TABULATE were also used for the first set of final report outputs:

```
PROC PRINT Data=Final_2020;
VAR Group Ret_Fall3 Ret_Fall2 Ret_Fall1 Avg_Ret;
RUN;

PROC TABULATE DATA=Final_2020;
CLASS Group Residency Major;
TABLE Major=" " ALL="Total", /* table row set-up */
       (Group=" " * (Residency=" " ALL="SubTotal")) ALL="Total" * (n)
       / BOX="Start Term = &StartFall" MISSTEXT="0"; /* table column set-up */
KEYLABEL n=" ";
RUN;
```

KEY TAKEAWAYS & NEXT STEPS

Three key takeaways have been identified so far in the project. First, there can never be too much communication between project team members. Frequent communication is key to avoiding mis-steps and accidentally “going down a rabbit hole.” A corollary of the first takeaway is to avoid working meetings of extreme length. While it may seem practical to schedule a full-day meeting, mental fatigue can lead to coding mistakes or data mis-interpretation.

A second takeaway is to frequently and repeatedly test preliminary calculations against known output. Frequent cross-checking catches errors and unexpected outcomes early in the process which provides for easier code de-bugging and less frustration later on.

The final takeaway is to be thoughtful and deliberate in writing code for mathematical calculations. For example, the AVG function was used for the three-year retention calculation because the AVG function handles missing data without causing an error. Since some of the analytical sub-groups were extremely small, the probability of missing data could not be ignored.

As the project moves into the next phase, the individual pieces will be assembled into a work routine. The ultimate goal is to build a process that requires minimal updating and input prior to use. An additional goal is to build the process in such a way that errors are easy to backtrace and fix without getting lost in an overwhelming sea of code.

CONCLUSION

Higher education has high data needs but typically has minimal staffing available to meet those needs. Automating routine data requests frees up staff to respond to acute data needs. Automation projects require careful planning, constant communication, frequent testing, and thoughtful selection of coding options to handle the potential for missing data.

RECOMMENDED READING

- Cook, J. & Truett, J. (PharmaSUG, 2023). *Making Publication Metric Tracking Easy: Using an Automated, Integrated System of R, SAS, and Microsoft Power BI to Ease the Pain of Assessing Publication Metrics*. Available at lexjansen.com
- Smith, K. D. (SESUG, 2019). *Which report is appropriate? Let SAS® figure it out!* Available at lexjansen.com
- Smith, K. D., & Earls, P. G. (SESUG, 2019). *From CSV to SAS®: Dealing with empty fields and repeated observations*. Available at lexjansen.com
- Straney, R. (SESUG, 2015). *PROC TRANSPOSE: Flip your Data 90o and Save Time*. Available at lexjansen.com
- Straney, R. (SESUG, 2018). *Going from Zero to Report Ready with PROC TABULATE*. Available at lexjansen.com

CONTACT INFORMATION

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

Kelly D. Smith, Ed.D.
smithke@uncsa.edu

Elizabeth S Davis
Director of Academic Planning & Institutional Research
davisel@uncsa.edu