

SAS® and the Voluntary Framework of Accountability:

A Prime Example of the Use of SAS in Education

Smith, Kelly D.; Frye, Bobbie E.; Earls, Paul G., Central Piedmont Community College

ABSTRACT

Community colleges in the United States provide accessible learning opportunities for more than 10 million students each year (Community College FAQs, n.d.). The student population served by community colleges often differs from the student population served by four year colleges. Consequently, accountability metrics developed for four year colleges are not always appropriate for two year community colleges (American Association of Community Colleges, 2012).

In 2012, metrics designed to better evaluate the effectiveness of community colleges were established as part of the Voluntary Framework of Accountability (VFA). The metrics used in the VFA were developed by the American Association of Community Colleges (AACC), the Association of Community College Trustees and the College Board Advocacy and Policy Center.

VFA metrics utilize data that community colleges are already collecting for internal and external reporting (state and federal). College data is provided to VFA through the production and submission of five comma separated values files (CSV files). The creation of these data files requires SAS users to utilize a wide array of techniques in extracting, recoding, and merging data from multiple data sources at the college.

This paper summarizes the development process and testing of SAS code to create the VFA datasets at Central Piedmont Community College (CPCC) in North Carolina. Particular attention is paid to the sub-setting, merge, and export procedures that were repeatedly utilized in the dataset creation process. The process of creating SAS code for the VFA datasets provides an excellent example of the use of SAS in the field of education.

INTRODUCTION

As with K12 education, institutions of higher education are being called on to produce evidence of their effectiveness. Traditional metrics, such as graduation rate, do not provide a comprehensive picture of effectiveness for community colleges (AACC, 2012). VFA metrics are based on data already collected for mandatory reporting; however, the data must be “reframed” to fit VFA guidelines.

At CPCC, the relevant data is maintained in a variety of datasets within the Office of Institutional Research. VFA reports are produced on an annual basis, so our goal was to develop SAS programs that could automate report production with a minimal amount of user input. Specific sections of the programs have been chosen to highlight some of the techniques: use of macro variables, extraction of data from multiple sources, identification of students eligible for specific cohorts, “reframing” of CPCC data to fit VFA specifications, importing and exporting data in non-SAS formats. Some of the code has been simplified to highlight the desired aspect and avoid confusion.

MACRO VARIABLES

Macro variables are used throughout the VFA programs to generalize the code and simplify the use of the programs for annual reporting. Macro variables are identified at the top of each program, follow a standard pattern, and can be easily updated each year. The macro variable itself is then used in the body of the program for generalization of the code. Macro variables that are found in the code presented in this paper include:

```
%LET lb01=ALL;  
%LET ds01=all_  
%LET yt01=201403;  
%LET rp01=_et;
```

The macro variables in this example can be combined to create the name and location of a specific SAS dataset:

```
SET    &lb01 .&ds01&yt01&rp01;
```

The macro variables refer, respectively, to a SAS library (lb), a dataset name (ds), an academic year and term (yt), and a reporting data point (rp). In this case, dataset all_201403_et is located in the library ALL.

MULTIPLE DATA SOURCES

Every VFA program pulls data from at least one information source in addition to the cohort dataset. For example, the VFA dataset “Student Developmental Need File” provides information on the developmental needs of cohort students. To determine the developmental level of the students in the cohort, we pulled information from three sources: 1) testing file, 2) transfer credits, and 3) override permissions. We used the following process to determine the final developmental placement for the cohort:

1. Testing File
 - a. Create tracking variables (English, Math) from testing file dataset
2. Transfer Credits
 - a. Create Math / English subsets from transfer credit dataset
 - b. Create tracking variables
 - c. MERGE Math / English subsets
 - d. MERGE with Testing File
3. Override Permissions
 - a. Create Math / English subsets from override dataset
 - b. Create tracking variables
 - c. MERGE Math / English subsets
 - d. MERGE with combined Testing/Transfer File

SELECTING COHORT STUDENTS

The two and six year cohorts (Fall 2014, Fall 2010) are used to track student progress and outcomes (SPO). These cohorts consist of students at CPCC for the first time in the summer or fall term and enrolled in a credit course for the fall term. Students enrolled exclusively in English as a Second Language (ESL) courses and dually-enrolled students (high school students in college classes) are not included. Students who were previously dually-enrolled but who have graduated and are at the college for the first time as a college student are included in the cohort (VFA Metrics Manual 2017).

DUALLY-ENROLLED STUDENTS

At CPCC, FirstCUTerm is used to identify a student's first term as a curriculum student. Dually-enrolled students have a value of FirstCUTerm that matches their first curriculum term as a dually-enrolled high school student. If these students return to CPCC after high school graduation, the value of FirstCUTerm will not change. Therefore, these students must be identified based on their status as dual-enrollees (ADCStuCurrentType of 'CCPP') in a previous semester.

We first identified a preliminary set of regular students in the fall cohort term. We also identified dually-enrolled students in the fall cohort term. We then used the MERGE procedure to make sure current dually-enrolled students were eliminated from the dataset of regular students. Next, we examined the previous fall and spring semesters for dually-enrolled students and identified them with a new variable, CCPP. Then, we used the MERGE procedure to identify regular students in the fall cohort who had been dually-enrolled in the previous year. Student who were neither previously dually-enrolled nor first term students in the summer or fall were removed to create the preliminary cohort.

```
/* non CCPP in cohort term */  
DATA Two_year00;  
SET    &lb01 .&ds01&yt02&rp01;  
WHERE  ADCCLsAcadLevel EQ "CU"
```

```

        AND ADCStuCurrentType NOT IN ('CCPP');
PROC SORT DATA=Two_year00; BY ADCClsAcadLevel;
RUN;

/* CCPP in cohort term */
DATA CCPP&yt02 (KEEP=ADCClsAcadLevel);
SET &lb01 .&ds01&yt02&rp02;
WHERE ADCClsAcadLevel EQ "CU"
        AND ADCStuCurrentType IN ('CCPP');
PROC SORT NODUPKEY DATA=CCPP&yt01; BY ADCClsAcadLevel;
RUN;

/* remove CCPP from cohort term */
DATA Two_year00a
        Two_year_CCPP;
MERGE Two_year00 (IN=A)
        CCPP&yt01 (IN=B);
BY ADCClsAcadLevel;
IF A=1 AND B=0 THEN OUTPUT Two_year00a;
ELSE IF A*B=1 THEN OUTPUT Two_year_CCPP;
RUN;

/* CCPP in prior year */
DATA Prior&yt01;
SET &lb01 .&ds01&yt00&rp02
        &lb01 .&ds01&yt01&rp02;
WHERE ADCClsAcadLevel EQ "CU"
        AND ADCStuCurrentType IN ('CCPP');
CCPP=1;
PROC SORT NODUPKEY DATA=Prior&yt02; BY ADCClsAcadLevel;
RUN;

/* identify current students who were CCPP in prior year */
DATA Two_year00b;
MERGE Two_year00a (IN=A)
        Prior&yt02 (IN=B);
BY ADCClsAcadLevel;
IF A;
RUN;

/* remove students whose first college term is not cohort term */
DATA Two_year01;
SET Two_year00b;
IF FirstCUTerm in ("&sttm01", "&sttm02") OR CCPP=1;
RUN;

```

ESL ONLY STUDENTS

To establish the final cohort, we eliminated students who were enrolled solely in ESL classes. We used a flag variable (ESL_Flag) and an accumulating variable (Flag_Sum) with selective output for students whose Flag_Sum variable was not equal to zero:

```

DATA Two_year02
        Two_allESL;
SET Two_year01;
BY ADCClsAcadLevel;
IF ADCClsPrefix IN ('EFL', 'ESL') THEN ESL_Flag=0;
ELSE ESL_Flag=1;

```

```

IF First.ADCPersonID THEN Flag_Sum=0;
Flag_Sum+ESL_Flag;
IF Last.ADCPersonID and Flag_Sum NE 0 THEN OUTPUT Two_year02;
IF Last.ADCPersonID and Flag_Sum EQ 0 THEN OUTPUT Two_allESL;
PROC SORT DATA=Two_allESL; BY ADCPersonID;
PROC SORT NODUPKEY DATA=Two_year02 OUT=Two_year03; BY ADCPersonID;
RUN;

```

CREATING VFA VARIABLES

Three examples of VFA variable creation are presented. In the first example, a CPCC race/ethnicity variable (IPEDSRace) is recoded into the VFA variable Race. In the second example, a series of CPCC variables are used to create the VFA variable Dev_Level_Math. The third example presents the method we used to create new student identification numbers.

VFA VARIABLE RACE

The values of the CPCC variable IPEDSRace provide a verbal description of the group (“Two or More Races”). The SUBSTR procedure was used to identify the group based on the first three characters of the value, and the IF THEN procedure was used to code the Race variable:

```

IF SUBSTR(IPEDSRace,1,3) EQ 'Ame' THEN Race='I';
ELSE IF SUBSTR(IPEDSRace,1,3) EQ 'Asi' THEN Race='A';
ELSE IF SUBSTR(IPEDSRace,1,3) EQ 'Bla' THEN Race='B';
ELSE IF SUBSTR(IPEDSRace,1,3) EQ 'Haw' THEN Race='P';
ELSE IF SUBSTR(IPEDSRace,1,3) EQ 'His' THEN Race='H';
ELSE IF SUBSTR(IPEDSRace,1,3) EQ 'Whi' THEN Race='W';
ELSE IF SUBSTR(IPEDSRace,1,3) EQ 'Two' THEN Race='2';
ELSE IF SUBSTR(IPEDSRace,1,3) EQ 'Non' THEN Race='N';
ELSE Race='U';

```

VFA VARIABLE DEV_LEVEL_MATH

Currently, CPCC offers eight developmental math modules. Students are assigned to specific modules based on their placement. As a first step, the ARRAY procedure was used to combine the eight math modules into an array (matmod). Then, we created a second array (Mat_Flag) by assigning numeric values to each math module in the first array. Next, we created the variable Total, whose value was obtained by applying the SUM procedure to the Mat_Flag array:

```

ARRAY matmod{8} MATModule01-MATModule08;
ARRAY Mat_Flag{8};
DO i=1 to 8;
    IF matmod{i}=' ' THEN Mat_Flag{i}=0;
    ELSE Mat_Flag{i}=1;
END;
Total=SUM(of Mat_Flag{*});

```

The VFA developmental need framework has three levels: “one level below in math”, “two levels below”, “three or more levels below” (VFA Raw Upload Guide, 2017, p. 9). The IF THEN procedure was used to assign the value of the VFA variable Dev_Level_Math based on the value of Total:

```

IF Total IN (1,2,3) THEN Dev_Level_Math='1';
ELSE IF Total IN (4,5,6) THEN Dev_Level_Math='2';
ELSE IF Total GT 6 THEN Dev_Level_Math='3';

```

VFA VARIABLE STUDENT_ID

We used the TRANWRD procedure to protect our students’ privacy by encrypting their student identification numbers (ADCPersonID). A variation of the actual transcription code follows:

```

Student_ID=ADCPersonID;
  Student_ID=TRANWRD(Student_ID, "0", "Q"); /* ADCPersonID encryption */
  Student_ID=TRANWRD(Student_ID, "1", "W");
  Student_ID=TRANWRD(Student_ID, "2", "E");
  Student_ID=TRANWRD(Student_ID, "3", "R");
  Student_ID=TRANWRD(Student_ID, "4", "T");
  Student_ID=TRANWRD(Student_ID, "5", "Y");
  Student_ID=TRANWRD(Student_ID, "6", "U");
  Student_ID=TRANWRD(Student_ID, "7", "I");
  Student_ID=TRANWRD(Student_ID, "8", "O");
  Student_ID=TRANWRD(Student_ID, "9", "P");

  Student_ID=TRANWRD(Student_ID, "Q", "3");
  Student_ID=TRANWRD(Student_ID, "W", "8");
  Student_ID=TRANWRD(Student_ID, "E", "1");
  Student_ID=TRANWRD(Student_ID, "R", "6");
  Student_ID=TRANWRD(Student_ID, "T", "7");
  Student_ID=TRANWRD(Student_ID, "Y", "4");
  Student_ID=TRANWRD(Student_ID, "U", "9");
  Student_ID=TRANWRD(Student_ID, "I", "2");
  Student_ID=TRANWRD(Student_ID, "O", "5");
  Student_ID=TRANWRD(Student_ID, "P", "0");

```

The TRANWRD procedure selectively replaces specific characters within a variable. For example, in the first TRANWRD procedure, all zeroes are replaced with "Q." The second set of TRANWRD procedures recodifies Student_ID with numeric characters. The first TRANWRD procedure in the second round replaces all "Q" values with threes.

Because ADCPersonID has been encrypted, we have the ability to reverse the process if needed.

IMPORTING / EXPORTING NON-SAS DATA

On occasion, the information needed to complete the VFA reports was not available as a SAS dataset. In addition, the final reports needed to upload information to the VFA website were either CSV files or Excel files.

CSV FILE IMPORT

Student financial aid information was most easily accessible through CSV files. The data was brought into SAS with this code:

```

DATA FA_&FaYr;
  INFILE "&PathRoot\FinAidAwards\FA_&FaYr..csv"
  DELIMITER = ',' MISSOVER DSD LRECL=32767 FIRSTOBS=2;
  INPUT
    StudentID :$10.
    FirstName :$25.
    LastName :$25.
    Year :$10.
    Term :$10.
    AwardedAmount :8.
    TermTransmittedAmount :8.
    Award :$10.
;

  FORMAT
    StudentID $10.
    FirstName $25.
    LastName $25.
    Year $10.

```

```

Term $10.
AwardedAmount 8.
TermTransmittedAmount 8.
Award $10.;
IF _ERROR_ THEN CALL SYMPUTX( '_EFIERR_',1);
RUN;

```

The INFILE statement identifies the specific location and filename of the data to bring into SAS. DELIMITER="," identifies a comma as the delimiter (used to separate variables). MISSOVER tells SAS to treat each line as a separate data record. DSD instructs SAS to treat two consecutive adjacent commas as a missing value, and LRECL=32767 identifies the length of the longest record in the file. Since the CSV files included variable headings in the first line of data, we used FIRSTOBS=2 so SAS would start reading data with the second line instead of the first.

INPUT provides SAS with the names and variable type for each piece of data being read from the file, while FORMAT provides information on the type of format to be used for each variable in the new SAS dataset. The use of a colon modifier within the INPUT statement enabled us to avoid including an INFORMAT section in the code. Errors that may occur during the procedure are transformed into the macro variable _EFIERR_ by the CALL SYMPUTX procedure.

CSV FILE EXPORT

Exporting data is a fairly straightforward process. The original SAS dataset is identified with the DATA= statement, and a specific location for the new data file is provided by OUTFILE. DBMS= is used to identify the type of file that is being created, and the REPLACE option indicates that SAS should overwrite any existing file with the same name.

```

PROC EXPORT DATA=Final_TwoDemo
OUTFILE="&PathRoot\TwoYearCohort\Two_Year_Student_Demographic_File.csv"
DBMS=csv
REPLACE;
RUN;

```

The export process is similar for both CSV and Excel files. The primary difference is the use of "xls" instead of "csv" as the type of data to export.

CONCLUSION

As we have shown, we used a variety of SAS techniques to create VFA reports. Since these reports will be generated on an annual basis, we wanted to make the programs not only easy to use, but also easy to update from year to year. As a consequence, we wrote a series of ten SAS programs that will generate the information in VFA specified format each year.

REFERENCES

- American Association of Community Colleges (2012, February). *The Voluntary Framework of Accountability: Developing measures of community college effectiveness and outcomes*. Available from <http://www.aacc.nche.edu/vfa>
- American Association of Community Colleges (2017?). *VFA raw data upload guide*. Available from <https://vfa.aacc.nche.edu/Documents/RawUploadGuide.pdf>
- American Association of Community Colleges (2017, January). *Voluntary Framework of Accountability Metrics Manual Version 5*. Available from <http://vfa.aacc.nche.edu/Documents/VFAMetricsManual.pdf>
- Community College Research Center (n.d.). "Community College FAQs". Available at <http://ccrc.tc.columbia.edu/Community-College-FAQs.html>

CONTACT INFORMATION

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

Kelly D. Smith
Central Piedmont Community College
704-330-2722 x3810
kelly.smith@cpcc.edu

Bobbie J. Frye
Central Piedmont Community College
704-330-6459
bobbie.frye@cpcc.edu

Paul G. Earls
Central Piedmont Community College
704-330-6399
paul.earls@cpcc.edu

SAS and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration.

Other brand and product names are trademarks of their respective companies.