Paper 346-2013

# Using SAS® To Assess Individuals' Best Performance In Multiple Dimensions

Aude Pujula, Louisiana State University; David Maradiaga, Louisiana State University

## ABSTRACT

There are many cases where we need to look at the best performance of an individual in several disciplines over multiple time events. For instance, we might want to know a triathlete's best position in the three disciplines over all the races of the season, or the highest test scores of a student in several sub-scores. Looking at the latter example, this paper compares four different methods implementable in Base SAS® to create a data set that contains one record per student corresponding to the highest test scores. Of particular interest is the use of PROC SQL combined with the SELECT DISTINCT clause and the MAX function that allows the creation of the desired data set in one step.

## INTRODUCTION

As an illustration, we will look at the case of college admissions. Admission offices develop a set of criteria to evaluate students' applications to college. ACT scores are one of those criteria that officers look at. Test scores are broken down into four sub-scores (English, Mathematics, Reading and Science). There is also the composite score that is an average of the four sub-scores. Students often take the test more than once while admission officers may base their decision on the maximum of each score (e.g., Mathematics, English and Composite) regardless if those scores correspond to the same test date.

## THE DATASET

When a student takes the ACT test, she/he can request the scores to be sent to one or several colleges to which she/he wishes to apply. Periodically, admission offices receive ACT scores files from students. Table 1 is a partial print of an ACT file. It contains the id, last name, first name of each prospective student as well as three test sub-scores (English: ACT_E, Maths: ACT_M and Composite: ACT_C). Records appear in chronological order (test date). Note that we can have multiple records for one student (if she/he took the test several times).

| id | Last_Name | First_Name | Test_date | ACT_E | ACT_M | ACT_C |
|---|---|---|---|---|---|---|
| 4 | Clark | Jules | 09FEB2010 | 26 | 24 | 25 |
| 3 | Dupont | John | 10MAR2010 | 25 | 22 | 26 |
| 2 | Smith | Lea | 15APR2010 | 28 | 23 | 25 |
| 4 | Clark | Jules | 20APR2010 | 30 | 24 | 25 |
| 2 | Smith | Lea | 02JUN2010 | 24 | 28 | 28 |
| 1 | Berger | Caroline | 08JUN2010 | 24 | 23 | 24 |
| 4 | Clark | Jules | 22JUN2010 | 29 | 25 | 27 |
| 4 | Clark | Jules | 15AUG2010 | 29 | 24 | 28 |
| 1 | Berger | Caroline | 10SEP2010 | 19 | 18 | 21 |
| 2 | Smith | Lea | 20OCT2010 | 28 | 26 | 27 |

**Table 1. Original Dataset Snapshot**

Table 2 displays the dataset that we ought to create. There is only one record per student with the maximum of each sub-score.

| id | Last_Name | First_Name | ACT_high_E | ACT_high_M | ACT_high_C |
|---|---|---|---|---|---|
| 1 | Berger | Caroline | 24 | 23 | 24 |
| 2 | Smith | Lea | 28 | 28 | 28 |
| 3 | Dupont | John | 25 | 22 | 26 |
| 4 | Clark | Jules | 30 | 25 | 28 |

**Table 2. Desired Dataset Snapshot**

## ALTERNATIVES

We now present four alternatives to create the dataset as in Table 2.

### ALTERNATIVE 1

The first option is to create three sub-datasets, one for each sub-score. Those datasets are sorted by id as well as the corresponding sub-score by descending order to have the maximum score first. Starting with the English sub-score:

```
PROC SORT DATA=ACT_test_2
    OUT=ACT_E (DROP=ACT_C ACT_M);
    BY id DESCENDING ACT_E;
RUN;
```

Then, in a single data step we create the variable first to identify the maximum score for each id.

```
DATA ACT_E;
    SET ACT_E;
    BY id;
    first=FIRST.id;
RUN;
```

We repeat these two steps for the second (Mathematics) and third (Composite) sub-scores.

```
PROC SORT data=ACT_test_2
    OUT=ACT_M (DROP=ACT_E ACT_C);
    BY id DESCENDING ACT_M;
RUN;
DATA ACT_M;
    SET ACT_M;
    BY id;
    first=FIRST.id;
RUN;

PROC SORT data=ACT_test_2
    OUT=ACT_C (DROP=ACT_E ACT_M);
    BY id DESCENDING ACT_C;
RUN;
DATA ACT_C;
    SET ACT_C;
    BY id;
    first=FIRST.id;
RUN;
```

The three sub-datasets are then merged by id. The WHERE statement allows keeping only the maximum scores for each student.

```
DATA ACT_alt_1
(RENAME=(ACT_E=ACT_high_E ACT_E=ACT_high_M ACT_E=ACT_high_C)
DROP=test_date first);
    MERGE ACT_E ACT_M ACT_C;
    BY id;
    WHERE first=1;
RUN;
```

### ALTERNATIVE 2

A straightforward way of reducing the previous code is to create a MACRO. Macros are very useful when we have to write a similar code several times.

First, we define the MACRO which corresponds to the two steps carried out for each sub-score in alternative 1. A macro starts with %MACRO and ends with %MEND. We name the MACRO sortandfirst and specify in parentheses the parameters that will change each time we invoke the MACRO. In order to reduce the code, we use the same name (name=) for the output dataset and the sort variable in the PROC SORT statement.

```
%MACRO sortandfirst (name=, var1=, var2=);
PROC SORT DATA=ACT_test_2
    OUT=&name (DROP=&var1 &var2);
    BY id DESCENDING &name;
DATA &name;
    SET &name;
    BY id;
    first=FIRST.id;
RUN;
%MEND sortandfirst;
```

Once the MACRO is defined, we invoke it three times, specifying the values of the parameters. The three sub-datasets are here created.

```
%sortandfirst (name=ACT_E, var1=ACT_M, var2=ACT_C)
%sortandfirst (name=ACT_M, var1=ACT_E, var2=ACT_C)
%sortandfirst (name=ACT_C, var1=ACT_E, var2=ACT_M)
;
```

Last, we merge the three sub-datasets as in alternative 1.

```
DATA ACT_alt_2(RENAME=(ACT_E=ACT_high_E ACT_M=ACT_high_M ACT_C=ACT_high_C)
DROP=test_date first);
    MERGE ACT_E ACT_M ACT_C;
    BY id;
    WHERE first=1;
RUN;
```

### ALTERNATIVE 3

Alternative 3 uses PROC MEANS to compute the maximum of each sub-score. We use the CLASS statement instead of IF to avoid sorting the data beforehand. The MAX= option outputs the maximum of ACT_E, ACT_M and ACT_C specified in the VAR statement. The WHERE option in parentheses, avoids the computation of the overall maximum.

```
PROC MEANS DATA=ACT_test_2 NOPRINT;
    VAR ACT_E ACT_M ACT_C;
    CLASS id;
    OUTPUT OUT=ACT_MAX(WHERE=(_TYPE_ = 1)) MAX=ACT_high_E ACT_high_M ACT_high_C;
RUN;
```

The output dataset ACT_MAX contains the variables ID, ACT_high_E, ACT_high_M and ACT_high_C. However, the desired dataset (Table 2) also contains the variables First_name and Last_name. Thus, we need to merge ACT_MAX with the original dataset ACT_test_2 which contains these two variables. This implies to sort ACT_test_2 by id (PROC SORT), select the variables of interest (Last_name and First_name) as well as the id variable which is the common variable in ACT_test_2 and ACT_max. The NODUPKEY allows keeping one record by id.

```
PROC SORT DATA=ACT_test_2 NODUPKEY
    OUT=ACT_id (KEEP=id Last_name First_name);
    BY id;
RUN;
```

Finally, we merge ACT_test_2 and ACT_max to create the desired dataset.

```
DATA ACT_alt_3;
    MERGE ACT_id ACT_max (DROP=_FREQ_ _TYPE_);
    BY id;
RUN;
```

In this special case, there is actually a less cumbersome way of creating the desired dataset. In order to have the variables Last_name and First_name in the output data set created in the PROC MEANS procedure, it suffices to add those variables in the CLASS statement. After all, each id corresponds to one unique Last_name and First_name.

```
PROC MEANS DATA=ACT_test_2 NOPRINT;
    VAR ACT_E ACT_M ACT_C;
    CLASS id last_name first_name;
    OUTPUT OUT= ACT_alt_3(WHERE=(_TYPE_ = 1)) MAX=ACT_high_E ACT_high_M ACT_high_C;
RUN;
```

## ALTERNATIVE 4

Finally, alternative 4 uses PROC SQL to create the desired dataset in one single step.

In the SELECT statement, the option MAX() allows to simultaneously return the maximum of each sub-score for each student (id). Three new variables are created (ACT_high_E, ACT_high_M and ACT_high_C).

The DISTINCT clause deletes duplicates. The statement GROUP BY tells SAS to do whatever is specified in SELECT by id.

Note also that only the variables specified in the SELECT statement will appear in the output dataset. In fact, they will appear in the same order as in the SELECT statement.

```
PROC SQL NOPRINT;
CREATE table ACT_alt_4 AS
SELECT DISTINCT id,Last_name,First_name, MAX(ACT_E) AS ACT_high_E, MAX(ACT_M) AS
ACT_high_M, MAX(ACT_C) AS ACT_high_C
FROM ACT_test_2
    GROUP BY id;
QUIT;
```

## CONCLUSION

Table 3 summarizes the four methodological alternatives, giving the SAS procedures, statements, functions and/or options that were used to create the desired output dataset. Alternatives 3 and 4 appear to be the most efficient in terms of the number of steps to perform. Note also that this example can be extended to other statistics such as

minimum (all alternatives). Alternatives 3 and 4 accommodate most of the important descriptive statistics (average, variance, sum and so forth).

| | ALTERNATIVE 1 | ALTERNATIVE 2 | ALTERNATIVE 3 | ALTERNATIVE 4 |
|---|---|---|---|---|
| **SELECT VARIABLES** | DROP/KEEP in data step | DROP/KEEP in data step | CLASS and MAX= in PROC MEANS | Implicit with SELECT in PROC SQL |
| **RETURN THE MAXIMUM OF EACH SUB-SCORE** | PROC SORT with DESCENDING option | PROC SORT with DESCENDING option | MAX= in PROC MEANS | MAX() option in SELECT |
| **DELETE DUPLICATES** | Use FIRST. in data step | Use FIRST. in data step | Implicit with CLASS id | DISTINCT clause in SELECT |
| **SORT BY id** | PROC SORT by id | PROC SORT by id | Implicit with CLASS id | Implicit with GROUP BY |
| **NUMBER OF STEPS** | 7 | 3 (includes invocation of MACRO | 1 | 1 |

**Table 3. Alternatives Comparison**

## ACKNOWLEDGEMENTS

As SAS Student Ambassadors-2013 Award recipients, we express our gratitude to Dr. Julie Petlick for her support as SAS Student Programs Manager. As SCSUG 2012 Scholarship recipients, we would like to thank Dr. Lisa Mendez for her immensurable support as Student Scholarship Coordinator. We would also like to express our deepest appreciation to Mr. Kirk Paul Lafler who has been very inspiring, motivating, and an exceptional SAS mentor. Finally, we would like to thank Dr. Hector O. Zapata for his mentorship throughout the PhD Program.

## CONTACT INFORMATION

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

Name: Aude Pujula
Enterprise: Louisiana State University
Address: 30 Martin D. Woodin Hall
City, State ZIP: Baton Rouge, LA, 70803
Work Phone: 225-578-7268
E-mail: apujul1@tigers.lsu.edu

Name: David Maradiaga
Enterprise: Louisiana State University
Address: 28 Martin D. Woodin Hall
City, State ZIP: Baton Rouge, LA, 70803
Work Phone: 305-905-8566
E-mail: dmarad1@tigers.lsu.edu

## TRADE MARK