Paper 245-2008

Evaluating Sample Code for an Interview

Stephanie R. Thompson, Rochester Institute of Technology, Rochester, NY

ABSTRACT

Requesting sample SAS® code from job candidates before an interview is a good way to gauge their level of experience, ability, and style. Once you have reviewed the code, prepare your questions for the candidate. You want to make sure that they 1) wrote the code themselves, 2) understand what they wrote, and 3) are familiar enough with it to explain why things were done the way they were. This paper will provide you with some guidelines on how to get answers to these questions in an interview. Code samples that I have received over the last few years and the questions that arose will be used as examples. This is also a good time to ask if the candidate can think of another way to accomplish the same task using a different approach. Evaluating work samples can be a way to gain invaluable insight into your candidate's skills and this paper will help you get the most out of it that you can.

INTRODUCTION

Finding a good fit can be a challenge during the interview process especially if you are looking for certain technical skills. SAS programming is a skill that fits this category. A person can put SAS as a skill on their résumé but how can you be sure that they really know SAS? Asking for some sample code that they have written is a good start. Request that it be sent in prior to the interview so you have some time to look it over and prepare questions ahead of time. Can you be sure they wrote the code? No, but you may be able to tell once you start asking questions.

Once you establish that the candidate has written the code, you can begin to dig deeper into how well they understand the code. Do they understand what the PROCS do? Are they aware of run group processing? Are they comfortable with the concept of a SAS library? Knowing how comfortable they are with SAS is a good indication of skill level. Hopefully they are not just copying code from others and using it themselves. This technique can help you spot people who are doing just that. You are looking for someone who can write code, not just change a date and run code.

Lastly, looking at existing code is a great springboard for asking about alternatives to the PROC or DATA step they used. This is also an opportunity to see if they are familiar with functions and how they work. For example, if someone uses the following assignment statement, total1 = v1 + v2 + v3 + v4;, you can ask how else they could have written the statement (total1 = sum(v1, v2, v3, v4). If they mention the sum function you can further inquire about the differences between simple addition and the function (e.g., how missing values are handled). There are multiple ways to do tasks in SAS and the alternatives provided can help in judging their skill level.

The next sections will use actual examples submitted prior to interviews to illustrate the points above. Hopefully these will give you some ideas on how to get more information out of an interview or how to select the code you choose to submit for an interview.

DID YOU WRITE THIS?

Whether or not the candidate wrote the code themselves may be the hardest thing to determine. No one who is thinking clearly will say they submitted someone else's code or got it from the internet. A simple Google™ search on "SAS code" returns the web page, "SAS Online Samples" (http://support.sas.com/documentation/onlinedoc/code.samples.html), at the top of the list. This site has many pieces of code that can be borrowed. Wikipedia (http://en.wikipedia.org/wiki/SAS_System) even has sample code under the subject label "SAS System." The SAS Community Wiki (http://www.sascommunity.org) also contains examples if you search using "sample code." Looking at some of this code may provide clues if any internet borrowing occurred.

Another way to check for understanding is to ask the candidate to briefly explain the purpose of the code. If the code seems to generate weekly reports for sales data (based on your reading of it and any included comments) but you are told the purpose is to manage sales representative bonuses there may be a problem. It might be in your understanding of the code so you could ask for an overview of the flow of the program. This will also help define the candidate's understanding of the code.

You can also ask direct questions such as, “Why did you use PROC SORT here after PROC SQL?” If someone can explain the logic behind the choice it is a good indication that they made the choice. In the past, I have been given code that was written by someone else and I have often wondered why she made a particular choice of procedure. In that case, I could not explain the logic.

A PROMISING CANDIDATE

This section will highlight some of the very well-written code that was submitted for consideration. Each example will be followed by its strengths. The following code examples were all submitted by the same person.

```
/*      Project: PSAS (Persistent Sexual Arousal Syndrome)    */
/*      Purpose: This program is to analyze the demographic   */
/*    data with the total sample and subsamples, compare    */
/*      three distress levels and determine the predictors of */
/*    distress, and do univariate analyses and multiple     */
/*      regression analyses                                     */
/*      Programmer: xxxxxxxxxxxxxxxx                                 */
/*      Date: Nov. xx, xxxx                                 */
/*      Update: Jan. x, xxxx                                        */


options pageno=1 nodate formdlim='-';
```

The use of the comment block at the beginning of the code made it clear what the purpose of the program was. The dates showed that the programmer was familiar with change control and the options set at the beginning indicate that they knew how they wanted things set up for the execution of the program.

```
data a2;
  set a1(drop=subject_no);

  * Set all data with value 99 to missing data;
  array qq _all_;
  do over qq;
          if qq=99 then qq=.;
  end;

  * Define variable TimeOnSet;
  TimeOnset=12*(2004-onsetyr)+(7-onsetmo);

  * Create group variables DistGroup & SexGroup;
  if howdist in (1,2,3) then DistGroup=1;
  else if 4<=howdist<=9 then DistGroup=2;
  else if howdist=10 then DistGroup=3;
  if sexor in (1,2) then SexGroup=1;
  else if sexor in (3,4,5) then SexGroup=2;
  format relstat rel. sexor orient. genhlth level. [others omitted];
  attrib _all_ lable='';
run;
```

This sample of code shows good use of comments and a layout that makes it easy to read and follow. The conditional logic is well written and uses the ELSE IF logic that will save processing time. The formats also appear to be custom formats and demonstrate that the candidate at least knows how to apply formats to the data. You could inquire as to whether or not the person created the formats as well if you do not see them elsewhere in the code. Creating and invoking using user defined formats shows an expanded knowledge of SAS.

```
%macro ttest(iv);
proc ttest data=a2;
       class &iv;
```

```
        var domdes--domfsc; run;
%mend;

%ttest(sexgroup);      %ttest(sxpers);            %ttest(sxorg);
%ttest(sxdesir);       %ttest(sxident);           %ttest(sxintru);
```

Use of the SAS Macro language shows a potentially higher skill level. If you see macros in the code you can expand your questions to macro variables and the macro language itself. Use of macros for repetitive tasks shows some forethought in programming.

IN THE RUNNING

Some code is not well written but is functional. These examples may be from a budding programmer in training or a seasoned programmer that needs additional training. I wouldn't necessarily eliminate these candidates if they have other skills that I am seeking. I would, however, work during an interview to understand whether training would address any issues.

```
/*SURVEY ITERM 7 TO 16 FOR QUALITY SCALE*/
data exit0304;
set exit.edit0304;
if v17 = 1 then v17 = .;
else if v17 = 2 then v17 = 1;
else if v17 = 3 then v17 = 2;
else if v17 = 4 then v17 = 3;
else if v17 = 5 then v17 = 4;
if v18 = 1 then v18 = .;
else if v18 = 2 then v18 = 1;
else if v18 = 3 then v18 = 2;
else if v18 = 4 then v18 = 3;
else if v18 = 5 then v18 = 4;
if v19 = 1 then v19 = .;
else if v19 = 2 then v19 = 1;
else if v19 = 3 then v19 = 2;
else if v19 = 4 then v19 = 3;
else if v19 = 5 then v19 = 4;
**** repeats through 26;
Item=17; ans=v17; ansb=ans; output;
Item=18; ans=v18; ansb=ans; output;
Item=19; ans=v19; ansb=ans; output;
**** repeats through 26;
proc tabulate f=6.1 noseps;
format Item questc.;
format ans rate.;
*** etc.;
```

The code above will recode all of the survey variables but it is long and repetitive. This is a great application for a do loop. Even an array or use of macros could be appropriate. The person who wrote this code may be able to be trained in how to use these additional programming steps and streamline their programs.

```
Data NewSalary;
set my.salaryInfo;
If jobcode=1 then Newsalary= salary*1.03;
if jobcode=2 then Newsalary= salary*1.05;
if jobcode=3 then Newsalary= salary*1.08;
run;
```

This code is completely functional but may be a drain on processing resources if the my.salaryInfo data set is large. Here, each if statement will be evaluated for every observation in the data set when only one case can be true for each. A better alternative is illustrated below:

```
Data NewSalary;
    set my.salaryInfo;
    If jobcode=3 then Newsalary= salary*1.08;
    else if jobcode=2 then Newsalary= salary*1.05;
    else if jobcode=1 then Newsalary= salary*1.03;
run;
```

Here, use of the else if will cut down on processing time since once a true condition is found, the subsequent if statements are bypassed. Also, if there are more observations with a jobcode of 3 in the data set followed by 2 and 1, processing time would be improved even more. I would ask the candidate why the jobcodes were ordered in the manner above if this sample was received for an interview. Knowing how their choice impacts processing is a valuable skill as a programmer.

```
proc sql;
CREATE TABLE match AS
SELECT * FROM sumner.PRETEST a, sumner.POSTTEST b WHERE
a.ParticipantID=b.ParticipantID2

                                                    AND a.BDate=b.BDate2;
QUIT;
```

This code is technically correct but is difficult to read. How the code looks in the editor has an impact on how easily it is read and understood by others.

MAYBE NOT

The code samples that I request are supposed to be representative of the candidate's best work. Sometimes code is provided that may move a candidate down the list in terms of who will receive an interview. It is one thing to list SAS as a skill on a résumé but another to demonstrate the skill. Some knowledge of SAS exists but it may not be at the level of the position I am attempting to fill. The code shown in this section is indicative of a need for more extensive training than I may be willing to take on.

```
DATA F02 (KEEP = SSNBR ACT_COMP HSGPA NEWSTAT SCH_CODE SAT_TOTL FOREIGN
   ORG_ST STATE);
  SET SIS.STCEN02F; /* SELECT FIRST-TIME FRESHMEN, FALL 2002 */
  IF NEWSTAT = 'F';
  SCH_CODE = HSINST;
  QUALIFY = 'NO '; /* INITIALIZE QUALIFY TO NO. */
  PROC SORT DATA=F02;
     BY SCH_CODE;
  /* MERGE TO GET HIGH SCHOOL STATE INFORMATION */
  DATA F02MERGE (KEEP = SSNBR ACT_COMP HSGPA NEWSTAT SCH_CODE SAT_TOTL
       FOREIGN ORG_ST STATE QUALIFY SCH_CODE);
     MERGE F02(IN=A) SIS.SIS_AI(IN=B); /* GET HIGH SCHOOL CODE INFO */
       BY SCH_CODE;
       IF A;
  PROC SORT DATA=F02MERGE;
     BY SSNBR;
  DATA F02MERG2 (KEEP = SSNBR ACT_COMP HSGPA NEWSTAT SCH_CODE SAT_TOTL
       FOREIGN ORG_ST STATE QUALIFY SCH_CODE HS_DATE1);
     MERGE F02MERGE(IN=A) SIS.SIS_RB(IN=B); /* GRADUATION DATE INFO */
       BY SSNBR;
       IF A;

  IF (ACT_COMP GE 19 OR SAT_TOTL GE 890)
    THEN QUALIFY = 'YES';
  IF HSGPA GE 3
    THEN QUALIFY = 'YES';
  IF (FOREIGN NE 'US'
    OR ORG_ST NE 'TN'
    OR STATE NE 'TN'
```

```
OR HS_DATE1 LT 200201)
THEN QUALIFY = 'NO ';    /* SELECT INITIALLY QUALIFIED CANDIDATES */
```

This code is a little hard to follow and some things are done that seem unnecessary. In the first DATA step Qualify is initialized to “NO “. In the last DATA step, the conditional logic can also set Qualify to “NO “. It is not clear why it would be reset unless there are conditions above that could change it to “YES” and then the last condition would need to reset the variable again. The conditional logic is not very well written. ELSE IF is not used and could be re-written much more simply. The inclusion of the space after the no in the DATA step seems to imply that the programmer believes that all values for Qualify must be three characters long. This may cause difficulties later on if others use the data that has been generated from this program.

```
data studentnew;
merge ai1 as1 aa1 hs1 wh1 hw1 mr1 tot1 s4 low1;
by scode;
run;
title 'studentnew';
proc print data=studentnew;
run;
data s2;
set studentnew;
keep scode sname pctenroll ai as aa hs wh hw mr tot tot1 tot3;
run;
title 's2';
proc proc print data=s2;
run;
data s5;
set s2;
aiper=ai/tot;
asper=as/tot;
aaper=aa/tot;
hsper=hs/tot;
whper=wh/tot;
hwper=hw/tot;
mrper=mr/tot;
totper=tot/tot1;
totper1=tot/tot3;
proc print data=s5;
run;
```

Aside from the duplicate PROC in the last print statement, this code is also hard to follow. The data set names are not clear and do not seem to be sequential. For instance, the data set s5 is created from s2 but datasets s3 and s4 do not appear in the code. The layout of the code makes it difficult to read since all of the text starts in the first column.

DON’T KNOW SAS?

SAS is used heavily in my office and it is part of the skills listed in the advertising for jobs that I post. It isn’t the only skill that I am looking for though. If a candidate has a good résumé and application but is lacking SAS skills, I may still want to interview that person. I will ask that person to send in a sample of code that they have written in one of the languages listed on their résumé. I may not be an expert in that language but I should be able to get a feel for the flow of the program and see how it is documented.

Knowing other linear programming languages can translate well to SAS. If the language uses do loops, sub-routines, and functions, for example, the person may just need to learn some new syntax. The concept of programming is not foreign to them. However, if their only programming experience is in HTML, those skills may not translate as easily to SAS.

CONCLUSION

There are many ways to judge a job candidate's fit for the position you are looking to fill. Requesting code samples is one way that you can see examples of what the person believes to be their best work. Reviewing the sample will give you a sense of their level of programming ability as well as a starting point to expand your conversation into other areas. Asking candidates why they used a certain PROC or approach helps you understand their thought process. Asking how else something can be done helps you understand the depth of their knowledge.

REFERENCES

For additional information on interviewing SAS Programmers, see Jenine Milum's paper "Assessing SAS® Skill Level during the Interviewing Process" in the SAS Global Forum 2007 proceedings (http://www2.sas.com/proceedings/forum2007/126-2007.pdf).

Previously cited web references:
http://support.sas.com/documentation/onlinedoc/code.samples.html
http://en.wikipedia.org/wiki/SAS_System
http://www.sascommunity.org

ACKNOWLEDGMENTS

I would like to thank Jenine Milum whose paper "Assessing SAS® Skill Level during the Interviewing Process" provided the springboard for the concept of this paper.

The editing assistance of Robert Jackson, Associate Director of Academic Affairs Technology, was helpful in keeping this paper organized.

CONTACT INFORMATION

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

Stephanie R. Thompson
Rochester Institute of Technology
9 Lomb Memorial Drive
Rochester, NY 14623-5603
Work Phone: (585) 475-7922
E-mail: SRThompso@cs.com
Web: http://finweb.rit.edu/irps/