

Development of Automated Test Scoring

Using the SAS System

by Jon Patton
Miami University

ABSTRACT

Test scoring is an important component in evaluating student performance in any academic setting. The SAS System played a major role in developing a test scoring system of programs that grades multiple choice exams where responses to each question are bubbled in on scantron forms. These responses are compared to the exam key to determine the grade. Reports generated include overall test statistics and bar charts, individual student reports that includes questions missed, and an item analysis for each question. Other capabilities of the system include (1) the capability of rescoring exams without rescanning in case questions are dropped or the key is changed., (2) verifying student course enrollment by querying a series of Oracle tables containing student data, (3) allowing the use of scrambles (permutation of test questions) to minimize cheating, and (4) automatically sending a summary of test results and data to the professors universal disk space if requested. A SAS background is not required for the user to operate this system. The use of Unix shell scripting, PERL and SAS make this system very user friendly.

INTRODUCTION

At most academic institutions, faculty administer exams to evaluate student performance. Multiple choice exams are a popular choice for many since they can be graded easily. Institutions that have test scoring services allow the generation of results from these exams in a timely manner. At Miami University in Oxford, Ohio, test scoring was done by a series of PL-1 programs that executed on an IBM mainframe running MVS. Due to anticipated Y2K problems, this mainframe was soon to be dismantled. This necessitated a replacement for the test scoring software. The source code to some of these programs could not be found (the old programs ran as executables on the IBM) so the entire system had to be recreated. This system was designed to run on an AIX server using the UNIX operating system and would offer the same services and output as the old system. Some of the major features are:

1. Provide overall test statistics and a bar chart showing distribution of scores.
2. Provide individual reports to each student indicating questions missed with the student's answer and the correct answer provided.
3. Student verification of course enrollment.
4. Ability to handle courses cross-listed with other academic departments.
5. Separate listing of student scores for each section and course (if cross listed).
6. Item analysis for each question where the percentage of each response is provided for each question.
7. Ability to use up to four scramble keys for large sections. A scramble key corresponds to the same test as the original key but the order of the questions are permuted to minimize cheating.
8. Allow rescoring of an exam without rescanning the answer sheets if the faculty member later decides to change the key or drop a question.
9. Ability to update overall course statistics if students take the exam late.

10. Ability to automatically transfer test results and raw data to the faculty's Novell disk space if requested. This information is usually imported by the faculty member into a spreadsheet.
11. Automatically updates an online test log after a test has been scored. Each record contains the department, course number, faculty's name, and number of students taking exam.
- 12/ Include security features that prevent the students' scores and id from being accessible by others.
13. Allow anyone with minimal computer experience to operate these programs.

In order to provide the above features, a combination of SAS 6.12, Perl, and Unix shell scripting was employed to make the program interactive and flexible. Perl is used to create SAS scripts that are included in the SAS program that produces the output. It also creates Unix shell scripts that provided the FTP (file transfer protocol) and other utilities. The major Unix shell script presents the operator with a series of menus after he/she logs on to the AIX server.

OPERATIONAL PROCEDURE

The student takes the test bubbling in the answers on a custom designed scantron form. The form allows exams of up to 150 questions of which 50 may be 10-choiced. On the form is a field of columns where the student bubbles their unique Miami ID. This is used for student identification and eventually is used to determine enrollment verification. For tests that use scrambled keys, a field containing the form number is also filled in.

After the answer sheets are collected, the faculty member submits the answer sheets to the operator and fills out the test scoring request form. The sheets are then run through the scanner (with the answer key placed on top). An application program corresponding to the customized scantron form creates a text file which is FTPed to another AIX server that serves as a temporary location. The operator then goes to the PC that has a Telnet connection to the AIX server containing the Test Scoring Programs. The operator sees the following menu (which was developed as a UNIX shell script):

Select from the choices below:

- 1) Score a test
 - 2) Rescore an existing test
 - 3) Score additional sheets for a test
 - 4) View the logs
 - 5) Evaluation
 - 6) Rescore Evaluations
 - 7) Change your password
 - 8) Logoff
- Choice:

The operator would select choice 1 if the test is to be scored the first time. Option 2 would allow a test to be rescored without rescanning the answer sheets. This would be useful if a question is to be dropped or the key changed. If some students take a test late, choice 3 would require only the new sheets to be rescanned.

The scores from these sheets would then be integrated with the previous scores to update the overall test statistics.

Choice 4 is useful if the operator wants to view the log of tests previously graded or to check whether the last FTP attempt of sending the test results to the faculty's disk space succeeded. After an exam has been scored, an entry in the test log is automatically created. This contains information such as test number, the faculty's name, department, course, the date the test was scored, and number of students taking the exam. This for example would be useful if the operator wishes to rescore an exam and needs the test number. If an FTP attempt to a faculty's disk space fails for any reason, an entry is added to the FTP error log. This contains the FTP error message describing the cause of the failure.

Choices 5,6, and 7 are used for tasks outside the scope of test scoring.

When an option is selected, a corresponding Unix shell script is activated. Each shell script has a very simple structure:

- (1) it activates a Perl program that serves as the main driver
- (2) it executes the print command (produced by the Perl program) that sends output to the selected printer.
- (3) it deletes any SAS output for security purposes.

Each Perl program prompts the operator for information about the test and then uses this to create SAS and UNIX scripts that are later included in other modules. The information that the operator is prompted to enter is illustrated below. The prompts appear in bold type.

Select from the choices below:

- 1) Score a test
- 2) Rescore an existing test
- 3) Score additional sheets for a test
- 4) View the logs
- 5) Evaluation
- 6) Rescore Evaluations
- 7) Change your password
- 8) Logoff

Choice: 1

Enter 5-digit Test #: 00001

How many modifier letters (0,1,or 2)? :0

Send raw data to Professor's NDS? (1=Yes,0=No):1

Send test results to Professor's NDS?

(1=Yes,0=No):1

What is Professor's Unique ID?:pattonjm

Select Printer to Send Output

(0=Local,1=gas130):1

The test number is five digits and is used by Perl to create dataset names for the raw (scanned) data, the output file sent to the faculty's NDS (Novell Disk Space), and the SAS data set name containing the test information. The names for these will all contain the 5 digit test number. Perl use these names to produce filename and libname statements that are included in other SAS modules. The other prompt that requires some explanation is the faculty's unique ID. It is prompted only if he/she wants the raw data or the test results sent to their NDS.

The other functions of the Perl program are:

1. creates SAS and UNIX scripts, FTP scripts, and print commands based on the above information.
2. executes the SAS program providing the test scoring output.
3. removes files or scripts containing secure information.

After the operator has entered all of the preliminary information about the test, Perl launches a SAS program that brings up an FSEDIT session for entering more information about the test. The first edit screen is illustrated below:

```

FSEDIT WORK.ENTRY
Obs 1 Screen 1
Command ==>
NOTE: This application uses 47 screens.
INSTR                               Jon M.
Patton_____
Department                          SAN
Course Number                       483
Course Modifier                     ____
TERM                                2
YEAR                                2001
Title for Test Results              Test
#1
Number of Scrambles                 0
# of T/F                           0
Number of 5-Choice                  25
Number of 10-Choice                 10
total                               35
Penalty Fraction per T/F            0
Penalty Fraction per 5-choice       0
Penalty Fraction per 10-choice      0
Separate section listing            1
Student Output?                     1
Copy of individ responses            1
Desire computerize gradebook        1
Question 1 on Form 2                ____

```

The above is the first out of 47 screens. The next several screens allow the entering of scramble keys. The last item on the first screen above starts this type of entry. If a scramble key is used, the operator would enter the question on form 1 (which corresponds to the main key that is scanned) that question 1 on form 2 corresponds. Since an exam can have up to 150 questions and four different forms of the test can be given, this part of the entry process takes up most of the 47 screens. The other screens allow the operator to change the main key or to drop a question that may be ambiguous.

This entry process creates a permanent SAS data set for each test. If a test has to be rescored or additional sheets need to be scored, this data set reappears on the screen for editing. Any data on the screens can be modified. After all data has been entered, the operator exits the edit session and SAS then reads the scanned data and produces the necessary output. The output is automatically routed to the printer of the operator's choice. Each faculty receives a printed copy of the test results and the amount of output depends on the faculty's choices. The faculty can elect to have the raw data or test results sent to his/her NDS. This is useful for importing into a spreadsheet or conducting further analysis.

OUTLINE OF THE SAS PROGRAMS

There is a SAS program corresponding to each of the first three choices from the UNIX menu. Each of these programs consist of the inclusion of several SAS modules that are shared by the other two. The following is the SAS program for scoring an exam the first time:

```

options ls=132 nodate ps=45 pageno=1 replace;
x 'cd $TSCORE';
%include testfile;
data entry;
%include lib;
length  instr $ 20. depart $ 3. course $ 3.
        modifier $ 2. term $ 1. year $ 4.
        title $ 40. scram 3. tf 3. five 3.
        ten 3. n 3. tfp 3. fivep 3. tenp 3.
        sectlist 3. stdout 3. response 3.
        gradbook 3. s1-s450 3. dq1-dq150 3.
        ckey1-ckey150 3.;
array ii{150} i1-i150;
array dd{150} dq1-dq150;
do i=1 to 150;
    ii{i}=i;
    dd{i}=0;
end;
%include testid;
testdate=date();
proc fsedit data=entry label;
    x 'cd $TESTSCORE_HOME';
%include label;
data response;
    infile data missover;
%include input;
status=1;
%include score;
proc sort out=resp1;
    by id1;
%include banner1;
%include student;
%include distrib;
%include highlow;
%include individ;
%include header;
file response;
%include key;
%include rawdata;
%include gradbook;
%include log;
x 'cd $TSCORE';
%include sysfile;
set entry;
run;

```

The SAS scripts such as **testfile**, **lib**, **testid**, and **sysfile** are created from the Perl program and are largely based on the test number assigned by the operator. These consist of mostly filename and libname statements that are referenced by other modules. The remaining modules have specific functions and the SAS programs that rescore an entire exam or sheets added to an exam share most of these. For example, as the name of the modules suggest, **input.sas** reads the scanned data into SAS variables, **score.sas** scores the exam against the key (the first sheet that is scanned), and **highlow.sas** conducts a high-low analysis for each question. (The percent of top 27% scoring students who correctly answered the question are compared to the percentage of the lowest 27% students who answered correctly.) These are used in all three of the major SAS programs. The modular approach is especially useful if a change in the code needs to be implemented. In most cases the overall structure of the program remains intact as only a couple of the modules need to be modified.

PASSTHROUGH TO ORACLE

One of the major requirements is to verify student enrollment. This requires a pass-thru to Oracle to get this information since the student tables are stored in an Oracle database. A student is identified by their unique logon ID bubbled on the answer sheet, and only this information is used to search the tables. A very efficient way to accomplish this is to create a temporary Oracle table from a SAS data set using PROC DBLOAD (Buffum, 1996) and then removing the table after the passthrough is completed.

The following code creates an Oracle table called TEMP9 from the SAS data set REDUCE.

```

proc dbload dbms=oracle data=REDUCE;
%include logparms;
table=TEMP9;
load;
run;

```

The data set REDUCE contains the student ID, department, course, and term. The code in logparms.sas include the necessary logon and password information to access the database.

```

proc sql;
connect to oracle (%log_parm);
x 'cd $TESTSCORE_HOME';
create table student as select * from connection
to oracle
(select id1,depart,course1,term1,spriden_id,
        spriden_last_name,spriden_first_name,
        ssbsect_seq_num,ssbsect_crse_num,
        ssbsect_subj_code,ssbsect_term_code
from temp9,spriden,sfrstcr,szbuniq,ssbsect a,
        ssrxlst x1
where sfrstcr_pidm=spriden_pidm
and sfrstcr_term_code=a.ssbsect_term_code
and sfrstcr_crn=a.ssbsect_crn
and spriden_change_ind is NULL
and szbuniq_unique_id=RTRIM(temp9.id1)
and spriden_id=szbuniq_banner_id
and a.ssbsect_term_code=x1.ssrxlst_term_code
(+)
and a.ssbsect_crn=x1.ssrxlst_crn (+)
and ((a.ssbsect_crse_num=temp9.course1
and a.ssbsect_subj_code=temp9.depart
and a.ssbsect_term_code=temp9.term1)
or exists
(select 1 from ssbsect b, ssrxlst x2
where b.ssbsect_term_code = a.ssbsect_term_code
and b.ssbsect_term_code = x2.ssrxlst_term_code
and b.ssbsect_crn = x2.ssrxlst_crn
and x2.ssrxlst_xlst_group =
        nvl(x1.ssrxlst_xlst_group,'++')
and x2.ssrxlst_crn
        <>nvl(x1.ssrxlst_crn,x2.ssrxlst_crn)
and b.ssbsect_crse_num=temp9.course1
and b.ssbsect_subj_code=temp9.depart
and b.ssbsect_term_code=temp9.term1)));
disconnect from oracle;
quit;
run;

```

The student Banner ID (different from their unique logon ID), is obtained from their logon ID using the table SZBUNIQ. The table SPRIDEN contains the Banner ID of the student, the student's first and last name, and a dynamic ID called the pidm. The pidm is assigned to each student in the remaining Oracle tables during each access.

The table SSBSECT assigns a course reference number (ssbsect_crn) for the department and course. The table SFRSTCR uses this to determine the section that the student is enrolled. If the course is cross-listed with other courses, the Oracle table SSRXLST is checked. Course reference numbers having the same group code (indicated by the variable ssrxlst_xlst_group) would be cross-listed.

The major purpose of this access is to obtain the full name of the student, the section that the student is enrolled, and the Banner ID of the student. This Banner ID appears in the class list and is included as part of the test scoring output. If the student is not found to be enrolled in the given course or cross-listed courses, the student unique ID is printed in an exception report.

The next passthrough to ORACLE is to drop the temporary table TEMP9:

```
proc sql;
connect to oracle (%log_parm);
X 'cd $TESTSCORE_HOME';
%put &sqlxmsg;
execute(drop table temp9) by oracle;
disconnect from oracle;
quit;
run;
```

OTHER OUTPUT FEATURES

PROC UNIVARIATE provides univariate statistics for the exam and PROC CHART provides a horizontal bar chart of the distribution of scores. Since the output was designed to resemble that of the previous program, data step program using PUT statements were used to format many of the reports. Besides the listing of student raw scores, percents, and z-scores for each section, each student receives an individual report indicating which question he/she missed.

CONCLUSION

The test scoring system of programs has been used successfully for over a year despite the necessity of implementing occasional changes. For example, the Social Security of the student was bubbled in as the identifier; but due to privacy issues, it had to be replaced by the Student unique ID. This required only a change in the PROC SQL code illustrated above.

. It would require several pages to describe the workings of each individual module. So I hope to have covered many of the design aspects of the system; and how the integration of SAS, Perl, and UNIX shell scripts have worked well to make this system easy to use

REFERENCES

Buffum, Henry W., (1996) 'Strategic Uses of SAS Data Step Programming and SQL Passthrough to Query Oracle Databases', Proceedings of the Twenty-First Annual SAS Users Group International Conference, Cary NC: SAS Institute Inc.

ACKNOWLEDGMENTS

This would not be possible if it were not for the following people: Dean Harris, Brian Simms, Brian Korblick, and David Schaefer who helped with the SQL coding; Carrie Ledford who designed the UNIX menu script, and Micah Cooper who provided assistance with the FTP scripts.

THE AUTHOR

Dr. Jon Patton is a Teaching Technologies Senior Analyst for Miami Information and Computing Service. He is an adjunct professor in both the Systems Analysis and Decision Sciences Department. He teaches courses in statistics and mathematical modeling . He can be reached at:

Jon Patton
Miami Computing and Information Service
Miami University
351A Gaskill Hall
Oxford, Ohio 45056
Work Phone: (513) 529-8751
Fax: (513) 529-5577
Email :pattonjm@muohio.edu