Paper 180-2019

Who Do We Need To Follow-Up With? Developing A Process To Track Study Response Rates

Vincent Chan, IMPAQ International LLC

ABSTRACT

Low response rates are often a challenge in randomized control trials that bases an outcome measure off data collected through a survey or other forms. In this paper, we go over the process of developing a SAS program to create response rate reports in the context of an evaluation of a teacher professional development program. In this education study, a literacy assessment is hosted online by a testing company, but the project team needs to continually follow-up with participating teachers to ensure their students take the assessment during the testing window. Each week and sometimes ad-hoc, the team (nonprogrammers) requires a generated report to check on test response rates calculated at the treatment, course, teacher, school, and district level. This report is used to determine whether additional follow-up is needed, and if a certain course has reached for an acceptable percentage of completed assessments.

In addition to the development of the process, this paper will highlight macro techniques used to automize response rate calculations and how we utilize the Dynamic Data Exchange (DDE) method to output these rates into an accessible and easy-to-update Microsoft Excel spreadsheet. Lastly, the paper will share successes and challenges in the process, lessons learned in quality assurance and data validation, and possible improvements.

INTRODUCTION

Data collection is an integral part of any evaluation study. In the field of education research, developers of various programs want to test the effectiveness of their curriculum through a two-year randomized control trial. This specific study evaluates the effectiveness of a teacher professional development program, and whether application of various reading strategies in the classroom improves student literacy. Outcomes are measured through a literacy assessment and student survey. Random assignment is conducted at the school level, and the study follows teachers throughout the course of the study. Data is collected on all students that the teachers teach in each school year. The study has partnered with a testing company which will support the administration of a literacy assessment at the end of the spring semester of each school year. A student will only need to take the assessment once for the study, but are allowed to complete multiple tests if they are in more than one study class. Additionally, students will need to complete a student survey in the scheduled session(s) for each study class they attend. Students and classrooms only allow a small window of time to administer the literacy assessment and the survey due to the impending end of the school year. Once the school year ends, teachers will not be able to schedule additional make-up time for the assessment or survey. Now, the most important question is, how do we utilize SAS to combine all rosters and reports and create a process to provide a quick glance at response rates so that the team can actively follow-up and ensure timely completion of these items?

ATTACKING THE BUSINESS PROBLEM

IDENTIFYING REQUIRED METRICS AND STATISTICS

First, we tackle this by brainstorming what is the required information for the team to conduct follow-up. After meeting about this, the team came up with several useful metrics and conditions:

- Response rate must be calculated by treatment status, by school, by teacher, by classroom
- Response rates must be broken down by “submitted” forms and “in-progress” forms
- Rates must account for the number of students who opt-out of the survey[1]

[1] Students opt to take or opt-out of taking the survey by signing a consent form.

- Rates should not include extra non-study students[2]
- Flags should be created to alert team member to problematic response rates
- Report must show the administration window for each classroom/school
- An additional response rate report should be done for unique students only[3]
- An additional set of rates that counts a student as completing the survey or test if they took them in any of their classes (if they are enrolled in more than one study class)

Now that we understand what the team requires, we need to define what constitutes the response rate. The response rate can be defined as:

$$Rate \ = \ \frac{n_{responses}}{N_{\text{schools/students/classrooms}}}$$

Response rate = number of observations / N at unit level

The team then uses these response rates to identify which schools or teachers have not started the assessment or survey yet if the initially scheduled date has passed, and follow-up with them to encourage completion of these items as soon as possible. The team has defined 50% as an acceptable rate threshold, which will not require any additional follow-up.

Because the members of the project who will be conducting follow-up do not have SAS programming experience, we will create an Excel report for their viewing. This list of metrics is then "visualized" in an Excel template and sent to the team for further review. Because of budget and time constraints, it is important to review this in advanced to minimize additional programming effort later.

The team identified the following rates to be created at each level of reporting:

Student survey

1. Survey response rate
2. Survey response rate excluding opt-outs
3. Survey response rate of students who took it in any of their classes
4. Percentage of incomplete surveys
5. Average percentage of pages completed

Literacy assessment

1. Assessment response rate by classroom
2. Assessment response rate of students who submitted a test in any of their classes
3. Assessment response rate including incomplete tests (where we count incompletes as response)
4. Number of in-progress tests and percentage of test completed
5. Number of in-progress tests for tests started in any of the student's classes
6. For in-progress tests, the median percentage of unanswered questions

CREATING THE STUDENT SAMPLE (DENOMINATOR)

Now that we have identified the metrics required by the team, we identify the data sources that will be required to create the response rates. We have the following files:

[2] Non-study students take the survey and assessment at the same time as study students. However, their scores and responses are excluded from the analysis. Their responses are assigned a unique ID.

[3] Because study students could take multiple subject courses from different study teachers during the school year, some students will have multiple surveys and assessments. For example, a student could take English with Study Teacher A and Social Studies with Study Teacher B. Students are required to take the survey in both classes, but only need to take the assessment in one class. A student can take

- student roster file that contains basic student information like school, classroom, teacher, subject
- separate student roster file that contains an opt-out indicator, which has been manually tracked as signed consent forms are received
- class roster that contains student testing administration begin and end dates by school, teacher, and class ID
- student score reports by student ID downloaded from the testing company
- student survey responses by student ID

The simplest strategy is to create a master student universe file that can be used as the key to merge with all other datasets. We create this by merging the rosters with opt-out indicators with the official student roster file. This student-level file is then used to aggregate to the classroom, school, district, and treatment status level after scores and responses are merged on. Table 1. Student Roster File (simplified)Table 1 shows how this roster file appears. The class level roster that contains student test start dates will be merged only when we calculate response rates by classroom.

Table 1. Student Roster File (simplified)

Student ID	State	District	School	School ID	Treatment Status	Teacher ID	Classroom ID	Section Number	Opt out?
Student1	State1	Dist1	School1	Schid1	0	Tid1	History 8_Teacher1_Section1	1	1
Student2	State1	Dist1	School1	Schid1	0	Tid1	History 8_Teacher1_Section2	2	0

…

GATHERING THE NUMBERS (NUMERATOR)

Assessment score reports

The literacy assessment is an online form that is managed by the testing company. The team has been given access to the administration interface to directly download score reports. The score report for students who submitted the online assessment comes in a clean comma-separated values (CSV) file. However, the students that have open, incomplete tests are listed only in a PDF report. The PDF contains a list of incomplete students, one class per page. Even after much back and forth, the testing company was unfortunately not able to modify their report output on the backend or provide any data query output that could be saved a more accessible, importable format. Because our metrics will need to include these in-progress tests, the team must figure out a process for combining both sources into one clean data file.

After trial-and-error and research of different methods, we decided to batch convert the PDF reports to Excel using Adobe Acrobat Professional software. Although this was not an ideal conversion method, the software was able to save data from all pages onto one worksheet. This file is then imported using SAS and cleaned.

Next, we merge both files to obtain one clean file that contains names, student IDs, and class taken for all students who submitted tests along with their scores, and if a test is in-progress, the test start time and number of items answered. We create one binary indicator which flags student-classroom observations that have a completed test and a second binary indicator which flags student-classroom observations that have an in-progress test.

Lastly, since a student taking the test in any of his/her classes counts has having completed the test in all classes, we collapse the cleaned file on Student ID only, and for each of the binary indicators, take the max value for each Student ID. We then merge this file back to the previous cleaned file to add these additional indicators.

We also note that there is a special case where a student could complete a test in one class but have an in-progress test in another class. We do not want to count that student as having an in-progress test, and therefore add code to modify the indicators. Table 2 shows an example of how the resulting dataset appears.

Table 2. Student Assessment Score Report

Test token (Student ID)	Classroom ID	Scale Score	DRP Taken?	DRP Taken in Any Class?	In-progress Test?	In-progress Test in Any Class?
SID1	History 8_Teacher1_Section1	80	1	1	0	0
SID2	History 8_Teacher1_Section2	76	1	1	0	0
SID1	ELA 8_Teacher2_Section6	91	1	1	0	0
SID3	ELA8_Teacher2_Section6		0	0	0	1
SID3	Science 8_Teacher3_Section11		0	0	1	1

…

Student survey response reports

The student survey responses are saved as a clean comma-separated values (CSV) file format. The file contains responses and response times by student-classroom. We create the same binary indicators that flags completed surveys and nonsubmitted surveys. Next, we collapse the cleaned files on Student ID and take the max of the two created binary indicators. We then merge this file back to the first file to add these additional indicators of survey completion in any class. Table 3 shows a sample of the resulting survey report observations.

Table 3. Student Survey Report

Test token (Student ID)	Classroom ID	Survey Taken?	Survey Taken in Any Class?	In-progress Survey?	In-progress Survey in Any Class?	Number of Pages Completed?
SID1	History 8_Teacher1_Section1	1	1	1	1	10
SID2	History 8_Teacher1_Section2	1	1	0	0	23
SID1	ELA 8_Teacher2_Section6	1	1	0	0	23
SID3	ELA8_Teacher2_Section6	0	1	1	1	2
SID3	Science 8_Teacher3_Section11	0	1	0	0	0

…

MERGING ALL DATA AND OUTPUTTING RESULTS

Next, we create code to merge the master student roster file with the literacy assessment report file and student survey response report, joining by unique student ID and classroom ID. Since the current data will contain nonstudy students, we create two analytic datasets, one that includes and one that excludes nonstudy students.

Although our response rates have different levels of analyses, we want to automate the calculation of response rates and output to Excel. We create a macro that can process the student-level data and has the flexibility to output response rates at different aggregate levels based on different parameters. The calculation of the rates is done within the macro depending on the number of students in the analytic datasets. We also include useful flag variables that will alert the reviewer of any classes that require follow-up.

Within the macro, we decide to use the PROC SQL statement with a GROUP BY clause, as we can specify the calculations and number formats as well in one statement. We macro-tize the GROUP BY conditions to be the input parameters for the macro. Rates are calculated by first summing each of the indicators to be a number of students at specified aggregate level, then dividing by the number of students at the level.

Lastly, we output the response rates to different tabs of an Excel spreadsheet using the DDE method, and replacing the range of numbers in each sheet.

USEFUL TECHNIQUES

Throughout this process, we spent many hours developing the approach to the problem before programming the solution. In essence, we are creating reports by classroom; teacher; treatment status; state and treatment status; district and treatment status; and school and treatment status levels. We double the number of reports because we need to report all students and then study students only. Writing out twelve different PROC SQL statements is one way to approach this, but we need to include the possibility of adding more metrics or modifying a calculation. Therefore, we use a macro to allow different parameter inputs and multiple calls. Adding this flexibility to the macro helps the programmer to be responsive to team demands, and is something we should keep in mind when working on any programming project. Macros help to automatize seemingly simple functions, but can also be helpful to borrow when other project tasks require the same output or result.

In addition, we setup our Excel template to include conditional formatting that can highlight any classrooms or problematic units that require follow-up. Many of SAS output mechanisms to Excel either overwrite such conditional formatting in addition to the values. The DDE method is used in this case to output the response metric. Once we setup the template, we only need to make modifications to the template prior to output and all values are updated once the code is run. This eliminates the need to either program formatting within SAS code or manually format the Excel table afterwards. Other team members can freely enter the spreadsheet and be able to quickly check what classes require attention.

CHALLENGES

The development of this process required much forethought and planning. Since other team members are not SAS programmers, we need to spend more time at the beginning to ascertain what they are looking for. Despite all the brainstorming and planning at the beginning, the team still needed to be flexible in including additional metrics and reporting levels, and occasionally came back to ask for more metrics after programming was finished. Applying these changes was a challenge, but was much easier to resolve due to the advanced planning.

The testing company also was technically challenged in providing response rate reports that were clean and in an easy to import format for the in-progress tests. The PDFs outputted by the testing site were not ideal, and the conversion process to Excel was prone to errors. Nevertheless, we were able to adapt our processes for what the subcontractor was or was not able to do. It was important to have many conversations at the beginning of the project to find out company capabilities and finalize deliverable deadlines. In the beginning of the process, our contact person was not the actual technical lead who would be working with us on the data. We had to follow-up continually to identify the correct person and elevate the problems encountered to the correct support level. Once we were able to get the right person, we were able to get better support and answers to our questions, even if the company did not have the technical capacity to get us what we needed.

FUTURE IMPROVEMENTS

A simple search on SAS communities forums will show that DDE has been around for awhile, but is a rather outdated technology. There are now many other ways for SAS to interact with Excel, which may be more convenient and accessible for nonprogrammer team members. The replacement solution would be to use a combination of Visual Basic and the SAS Add-in for Microsoft Office. The use of scripts and stored processes allows for automation and access to other features of Excel like PivotTables.

After this is setup, we would have a process like the following. First, we can have nonprogrammer members of the team save the newest reports and data into a document folder. Then, we can setup an

Excel spreadsheet that can contain buttons, which when clicked will batch run the SAS response rate program and update all output. The SAS program will automatically import the data and process the calculations. This preserves all conditional formatting in the spreadsheet while making changes only to the numbers. To ensure that the numbers are correct, we can also build in quality control outputs to easily check for data irregularities.

CONCLUSION

From the above discussion, we can see that spending time in advanced to brainstorm the solution proved to be very important for the process later on. It allowed the programmer to be more responsive to team needs and have a smoother process. With future improvements, this process can be expanded with more efficiencies and savings in time and effort.

CONTACT INFORMATION

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

Vincent Chan
IMPAQ International LLC
vchan@impaqint.com