Spreadsheets in Education (eJSiE)

Volume 6 | Issue 1 Article 4

February 2013

Spreadsheet Activities with Conditional Progression and Automatically Generated Feedback and Grades

Thomas C. Juster
University of South Florida, Tampa, juster@usf.edu

Recommended Citation
Juster, Thomas C. (2013) Spreadsheet Activities with Conditional Progression and Automatically Generated Feedback and Grades, *Spreadsheets in Education (eJSiE)*: Vol. 6: Iss. 1, Article 4.
Available at: http://epublications.bond.edu.au/ejsie/vol6/iss1/4

This In the Classroom Article is brought to you by the Bond Business School at ePublications@bond. It has been accepted for inclusion in Spreadsheets in Education (eJSiE) by an authorized administrator of ePublications@bond. For more information, please contact Bond University's Repository Coordinator.

Spreadsheet Activities with Conditional Progression and Automatically Generated Feedback and Grades

Abstract
Spreadsheet activities following the Spreadsheets Across the Curriculum (SSAC) model have been modified using VBA programming to automatically generate feedback, calculate grades, and ensure that students complete them in a linear fashion. Feedback is based not only on the value of cells, but also on the formulas used to compute the values. These changes greatly ease the burden of grading on instructors, and help students more quickly master tasks and concepts by providing immediate and directed feedback to their answers. Students performed significantly better on the new spreadsheet activities compared to traditional SSAC versions, with 87% achieving perfect scores of 100%.

Keywords
Feedback, macros, spreadsheets, SSAC, VBA programming

Spreadsheet Activities with Conditional Progression and Automatically Generated Feedback and Grades

Thomas Juster
Department of Geology
University of South Florida
juster@usf.edu

Abstract

Spreadsheet activities following the Spreadsheets Across the Curriculum (SSAC) model have been modified using VBA programming to automatically generate feedback, calculate grades, and ensure that students complete them in a linear fashion. Feedback is based not only on the value of cells, but also on the formulas used to compute the values. These changes greatly ease the burden of grading on instructors, and help students more quickly master tasks and concepts by providing immediate and directed feedback to their answers. Students performed significantly better on the new spreadsheet activities compared to traditional SSAC versions, with 87% achieving perfect scores of 100%.

Keywords: Feedback, macros, spreadsheets, SSAC, VBA programming.

1. Introduction

Spreadsheets Across the Curriculum (SSAC) is a project created to develop and disseminate spreadsheet-based activities (called "modules") that teach quantitative literacy concepts in the context of solving relevant scientific problems [1, 2]. Over 70 modules are available for free download on the SSAC website [2], and are being used by instructors at the secondary and college level. The SSAC modules draw context from many areas of the mathematical, physical, and social sciences, and thus can be used in a wide array of undergraduate classes.

SSAC modules are self-guided standalone computer activities, and consist of an Excel spreadsheet paired with a PowerPoint presentation. The presentation introduces both scientific and mathematical concepts, describes a problem that can be solved with the spreadsheet, and explains how to do it using examples. Typically, presentations consist of 10-20 slides that mix text, graphics (sometimes including animations), and screenshots of the Excel spreadsheet being created. At the conclusion of the presentation the student is given a final capstone task to complete in the spreadsheet that builds on the lessons explained in the presentation. In addition, the final task requires the student to answer critical-thinking questions based on the spreadsheet calculations.

I have both designed SSAC modules and used them in my classes, and in general find them to be useful and effective. There are, however, three problems that consistently arise.

First, SSAC modules can be difficult and time-consuming to grade. In many modules students are essentially 'turned loose' on the spreadsheet and instructed to create a table or graph, which can be located anywhere on the worksheet. Early adopters of SSAC

modules quickly realized that this presented a grading nightmare, and adopted a template model to constrain the students' inputs to predictable areas [3]. Even when graders know where to look for the answers, however, determining whether they're right or wrong can be difficult. For example, a small error in one formula can propagate through the entire spreadsheet, making everything "wrong". Not only is this error difficult to identify, but it poses a dilemma—how much do you penalize the propagated wrong answers, and any inferences made from them? In addition, instructors are often interested not only in whether or not students get the right answer, but in how they get it. Students unfamiliar with Excel may simply use it as a canvas for entering numbers they calculated elsewhere—often with a hand calculator. Not only is this use contrary to the spirit of the module, but it robs the students of the learning associated with logical programming via formulas, which is one of the pedagogical advantages of teaching with spreadsheets [4-6].

The grading issue is especially problematic in large classes. At the University of South Florida we have used SSAC spreadsheets in classes with enrolments of up to 175 students, where they are often graded by just one or two graduate assistants. The grading workload associated with such high enrolments can be daunting.

Second, because grading is difficult, the feedback students receive on their completed spreadsheet modules is often minimal and of poor quality. Feedback is usually late—arriving when the graded spreadsheets are returned a week or more after they were submitted—and consequently of little use to the student [7]. Graders may be unqualified or unwilling to probe the details of a spreadsheet to discover where an error was made, being content to simply point out where the answers are wrong and to make a quick comment about what they should have been. All too often the feedback to the user consists of comments like, "The value [in this cell] is wrong, you must have made an error somewhere."

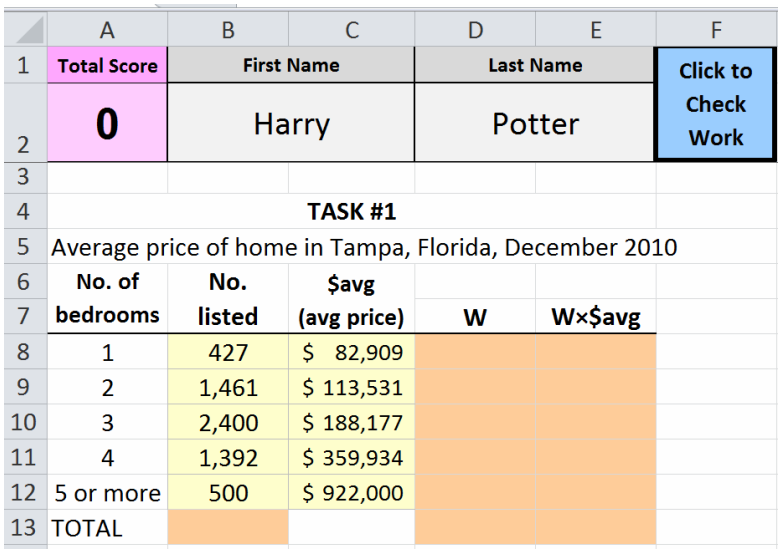
Finally, although the spreadsheets are designed to guide students through a problem slowly, modelling solutions and techniques with examples before assigning a capstone task that reprises these concepts, this is not how students often approach them. Instead, students go directly to the final task and start there. Many students have confessed this much when seeking help, admitting that they skipped over the preliminary explanations entirely—assuming (sometimes just hoping) that the Excel and math skills they already possessed would be sufficient to get them through.

The macro-enabled SSAC modules I describe in this paper provide solutions to each of these problems. Grading is automatic, freeing instructors to concentrate on evaluating answers to the higher-order critical thinking questions that appear on a subsequent quiz. Feedback is also automatic—and immediate—guiding the students toward the right answer without telling them what it is. As described below, the macros not only check to see that an answer is correct but also that it was obtained the correct way. And, finally, the activities in these spreadsheets are organized into progressively more complex tasks that must be completed sequentially, and perfectly, in order to advance to the next. In other words, they force students to take the baby steps that are written into the PowerPoint instructions.

For want of a better term, I call these spreadsheets "conditional progression, auto-feedback and grades" spreadsheets, or CPAFG spreadsheets. The next sections describe how the CPAFG spreadsheets are constructed, followed by an example.

2. Organization of CPAFG Spreadsheet Activities

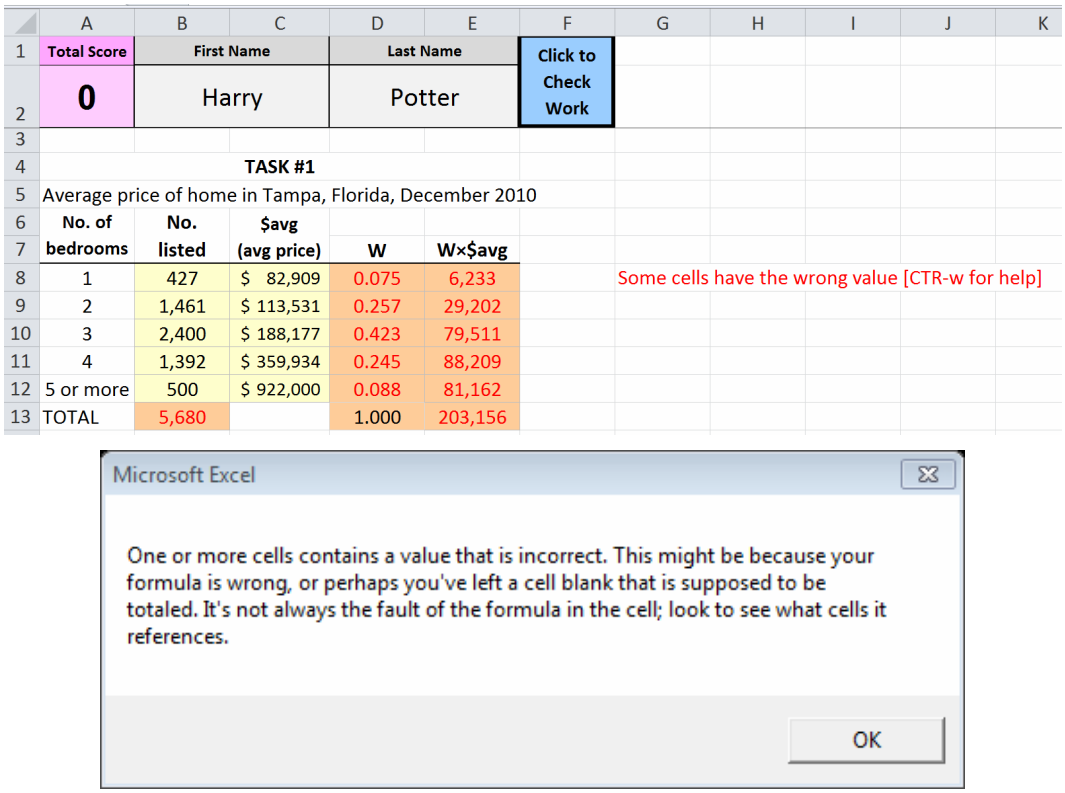
Every CPAFG spreadsheet activity is divided into 2-5 preliminary tasks and one final task. Each task requires the student to enter data or formulas and perform calculations in a portion of the spreadsheet. As in all SSAC spreadsheets, the cells are color-coded [8]: yellow for cells that contain numbers, and orange for cells that contain formulas. The preliminary tasks become progressively harder as the student advances through the activity, and the final task repeats and synthesizes skills taught in earlier tasks.



	A	B	C	D	E	F
1	Total Score	First Name		Last Name		Click to Check Work
2	0	Harry		Potter		
3						
4	TASK #1					
5	Average price of home in Tampa, Florida, December 2010					
6	No. of	No.	$avg			
7	bedrooms	listed	(avg price)	W	W×$avg	
8	1	427	$ 82,909			
9	2	1,461	$ 113,531			
10	3	2,400	$ 188,177			
11	4	1,392	$ 359,934			
12	5 or more	500	$ 922,000			
13	TOTAL					

Figure 1: Example of CPAFG spreadsheet on the density of the lithosphere, the outermost rigid layer of Earth. The key quantitative literacy concept in this module is the weighted mean, which is needed to compute the density of the layered lithosphere. This first task introduces the weighted mean in the context of home prices. Subsequent tasks apply the concept to the lithosphere

Upon opening the spreadsheet, only the first task is displayed (Fig. 1). Following instructions in the companion PowerPoint presentation, the student enters values and formulas into the cells to accomplish the task, and then clicks on a blue box labelled "Click to Check Work". This action triggers a macro that evaluates the input and displays feedback if anything is wrong. The feedback is color-coded with the font in the incorrect cell; for example, if a value in a cell is wrong the font is changed to red and a message in red text appears (Fig. 2). Using the specified control-key sequence displays a pop-up text box with a more detailed description of the problem.



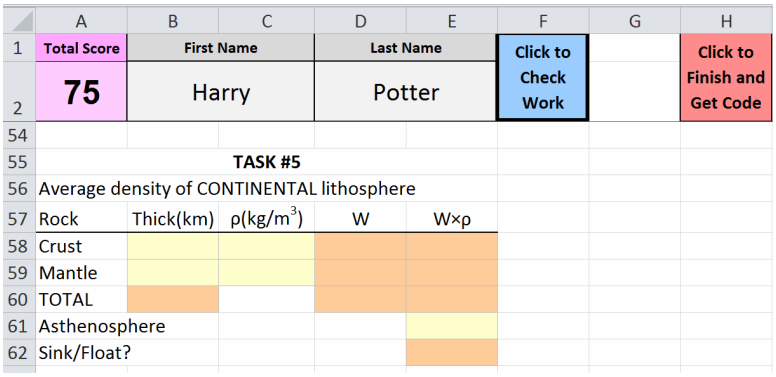
	A	B	C	D	E	F	G	H	I	J	K
1	Total Score	First Name		Last Name		Click to Check Work					
2	0	Harry		Potter							
3											
4			TASK #1								
5	Average price of home in Tampa, Florida, December 2010										
6	No. of	No.	$avg								
7	bedrooms	listed	(avg price)	W	W×$avg						
8	1	427	$ 82,909	0.075	6,233		Some cells have the wrong value [CTR-w for help]				
9	2	1,461	$ 113,531	0.257	29,202						
10	3	2,400	$ 188,177	0.423	79,511						
11	4	1,392	$ 359,934	0.245	88,209						
12	5 or more	500	$ 922,000	0.088	81,162						
13	TOTAL	5,680		1.000	203,156						

Figure 2: Example of feedback response in CPAFG spreadsheet lithospheric density (top). The formula in cell B13 is wrong, triggering an error. This error has propagated into other cells, which are also flagged as incorrect. Pressing <ctrl-w> brings up the text box (bottom) with additional explanation on the source of the error.

The student then modifies answers in response to the feedback, and repeats the process by clicking "Click to Check Work" again. Once the first task has been completed perfectly—and not before—the total score is updated to include the value of this task, and the second task is revealed. The procedure for the subsequent tasks is exactly the same as for the first, with the exception of the final task.

The final task (Fig. 3) differs from the preliminary tasks in two ways. First, the final task doesn't need to be completed perfectly to earn any points; points are awarded for any answers that are correct. Second, students have the option of concluding the activity at any time by clicking a second button, "Click to Finish and Get Code", which is only revealed and available during the final task. The code is simply an alphanumeric sequence, unique to each student, which can be presented to the instructor as proof of

completion of the module. Because the code[1] encrypts the student's name, score, and time when completed, it is unique and non-transferable, ensuring that it can't be 'loaned' from one student to another.



	A	B	C	D	E	F	G	H
1	Total Score	First Name		Last Name		Click to Check Work		Click to Finish and Get Code
2	75	Harry		Potter				
54								
55	TASK #5							
56	Average density of CONTINENTAL lithosphere							
57	Rock	Thick(km)	ρ(kg/m^3)	W	W×ρ			
58	Crust							
59	Mantle							
60	TOTAL							
61	Asthenosphere							
62	Sink/Float?							

Figure 3: Final task in CPAFG spreadsheet on the density of the lithosphere. (Notice that the top two rows are frozen, which obscures rows 3-53 where tasks 1-4 are displayed.) The student has already accumulated 75 points by completing the four previous preliminary tasks perfectly, and can earn the final 25 points by completing this task. The final task is more difficult than the previous ones, in that the student is responsible for deciding what values to enter into the yellow cells (B58:C59; E61) in addition to inputting the formulas in the orange ones (B60; D58:E60; E62). The student can also choose to finish and accept his or her score by clicking on the red button, "Click to Finish and Get Code".

3. Layout of CPAFG Spreadsheets

The spreadsheet consists of four worksheets, three of which are hidden from the user and protected from exposure using an add-in encryption program[2]. The visible worksheet is labelled "Answers"; this is where the student enters all data and formulas, and performs all calculations. The second worksheet, "Sheet2", stores data for tasks that will be revealed as the user progresses through the activity. The third worksheet, "Sheet 3", contains the encryption calculations that generate the secret code used to demonstrate completion of the activity. The last worksheet, "KEY", contains the correct answers for all cells (formulas and values). By default, this worksheet is an exact duplicate of the templates used for all tasks, except that correct answers are inserted.

[1] The details of the code-generating algorithm are not discussed in this paper, so that students can't discover them. The instructor, of course, is free to use any algorithm (or none at all).

[2] Excel's built-in protections are weak and easily defeated, requiring a third-party solution to make them truly secure. I use XLSafePro, which installs as an Excel Add-In and produces a completely secured .xlsm version that is otherwise transparent to the user.

Thus, for example, the correct value for cell Answers!D6 (cell D6 on worksheet "Answers") will be found in KEY!D6 (cell D6 on worksheet "KEY"). The instructor can optionally place the correct answers in a different column (but same row) as the Answers worksheet.

4. General Structure of the VBA Code

The spreadsheet uses macros written in VBA 7.0 for MS Excel 2010[3]. There are approximately 1,000 lines of code (depending on the specific application) with three main macro subroutines that are triggered by the user:

Sub Worksheet_Activate: This macro is triggered automatically when the worksheet is opened. It prompts for the user's first and last name, and then initializes variables by calling an internal subroutine InitializeSpreadsheet. InitializeSpreadsheet contains values that are designed to be modified by the instructor whenever a new activity is created: total number of tasks; cell ranges on Sheet2 where each task can be found; and the point values of each task.

Sub CheckTask: This is the main macro, run whenever the user clicks the box "Click to Check Work" (Fig. 1). It evaluates one or more cells in the active task for correctness based on user-supplied criteria, and then displays color-coded feedback based on any errors found. This macro is designed to run repeatedly as students correct their answers based on the feedback received. If there are no errors the point total is updated, the next task is revealed, and the active task is updated to the next one. If the active task is the last one, the subroutine calculates and displays the student's score in addition to checking for errors.

CheckTask has a simple structure. For each task, it calls three subroutines, in order:

PrepareToTest
TestCell(args) [which is repeated for each cell to be evaluated]
ReportErrors

PrepareToTest resets all counters and error flags, and ReportErrors displays the color-coded feedback if errors are detected. The main work of CheckTask is done by the

[3] Some of the code was generated automatically with Excel's Record Macro command or borrowed from free code warehouses on the Internet (e.g., [9]); the rest was written by the author. While there's no doubt the resulting code is inelegant and inefficient (it was written by a geologist, not a computer programmer!)—it works. The take-away point is that if a programming novice can create the VBA code in one long weekend, then the required programming skills are not beyond any motivated instructor.

subroutine TestCell, which must be modified for every cell the instructor wants to test. TestCell contains up to ten optional string arguments which determine its action (Table 1). The arguments can be supplied in any order.

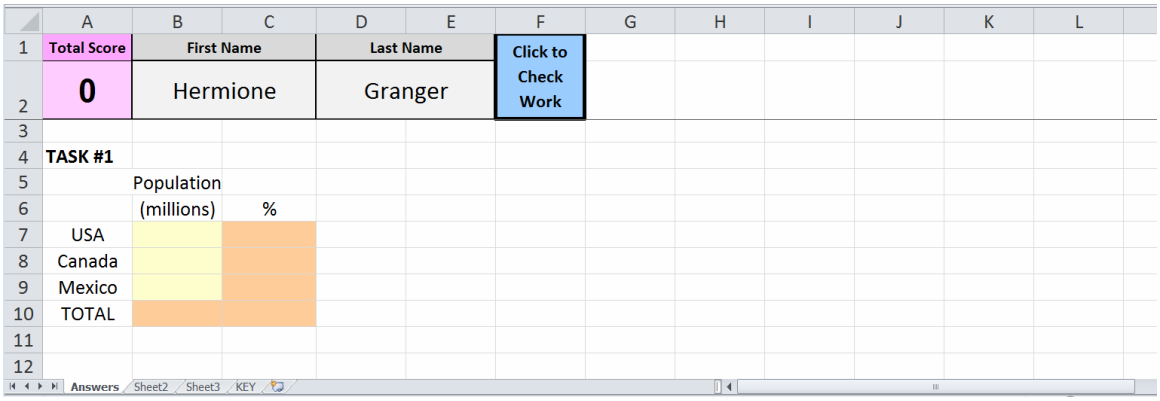
Table 1: Arguments for CheckTask

Argument	Purpose	Example(s)
"BadStr"	Checks the string value in cell against the correct string, located in the same cell address in the KEY worksheet.	TestCell("D2", "BadStr")
"BadVal[/X][:Tol]"	Checks the numeric value in cell against the correct value. Correct values by default are located in the same cell address in the KEY worksheet. Optionally, the user can specify a different column X (but same row) using the "/X" suffix. If an optional fractional tolerance Tol is provided the value in the cell is evaluated against the correct value ± Tol*Correct Value	TestCell("D2","BadVal") TestCell("D2","BadVal/J") TestCell("D2","BadVal:0.01")
"IsBlank"	Checks to see if cell is blank	TestCell("D2","IsBlank")
"NoAbs"	Checks to see if cell formula contains an absolute cell reference	TestCell("D2", "NoAbs")
"NoFor"	Checks to see if cell contains a formula (e.g., starts with "=")	TestCell("D2","NoFor")
"NoFun[:F]"	Checks to see if cell formula contains any of the common *Excel* functions AVERAGE, COUNT, MIN, MAX, or SUM. If an optional colon followed by a string F is found, checks to see if the cell contains only the *Excel* function F	TestCell("D2","NoFun") TestCell("D2","NoFun:COUNTIF")
"NoOp"	Checks to see if cell formula contains a mathematical operator +, -, *, or /	TestCell("D2","NoOp")
"NoRef:C1[,C2…]"	Checks to see if cell formula contains a cell reference specified by C1. Up to 4 additional cell references can be provided, separated by commas, and all must be present	TestCell("D2","NoRef:F5") TestCell("D3","NoRef:D1,D2")
"NoRng"	Checks to see if cell formula contains a cell range [C1:C2]	TestCell("D2","NoRng")
"Pts:N"	Sets value of correct answer as N points for that cell (only used in Final Task)	TestCell("D2","Pts:2")

Sub SubmitScore: This macro is triggered when the student clicks the box "Click to Finish and Get Code", which is only available on the final task. SubmitScore evaluates the answers on the final task, awards points if earned, calculates the students total score and displays an encrypted code. SubmitScore can only be triggered once, and once run the student's score is baked into the encrypted code and cannot be increased. If CheckTask detects that the student has completed the final task perfectly, it automatically triggers SubmitScore.

5. An Example

Consider a spreadsheet activity designed to teach students how to use Excel to calculate sums and percentages. The activity requires students to look up the population of the three countries in North America, enter them in to a spreadsheet, sum them up, and calculate the percentage of the total population of North America for each country. The "Answers" worksheet of the CPAFG spreadsheet, where the students will do their calculations, is shown in Figure 4. Students will enter numbers in cells B7:B9 and formulas into cells B10 and C7:C10.

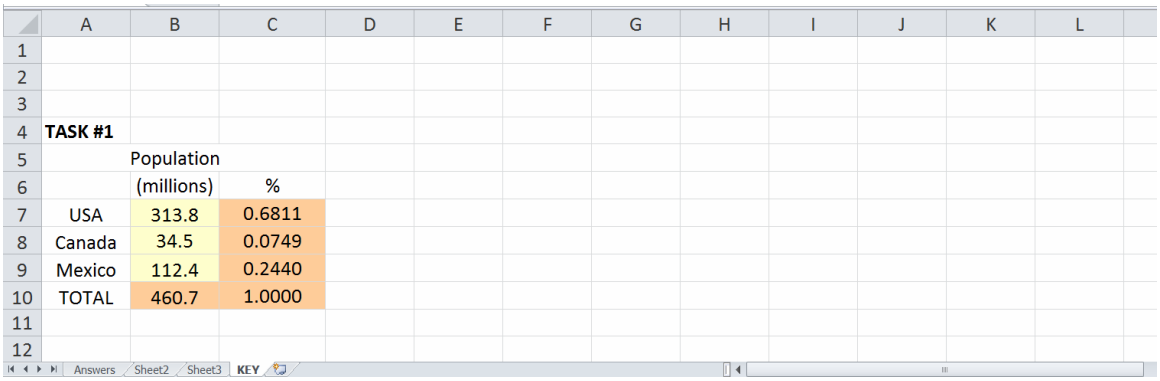


	A	B	C	D	E	F	G	H	I	J	K	L
1	Total Score	First Name		Last Name		Click to Check Work						
2	0	Hermione		Granger								
3												
4	TASK #1											
5		Population										
6		(millions)	%									
7	USA											
8	Canada											
9	Mexico											
10	TOTAL											
11												
12												

Answers | Sheet2 | Sheet3 | KEY

Figure 4: "Answers" worksheet for example problem

This template is copied into the "KEY" worksheet, and the correct answers are input (Figure 5):



	A	B	C	D	E	F	G	H	I	J	K	L
1												
2												
3												
4	TASK #1											
5		Population										
6		(millions)	%									
7	USA	313.8	0.6811									
8	Canada	34.5	0.0749									
9	Mexico	112.4	0.2440									
10	TOTAL	460.7	1.0000									
11												
12												

Answers | Sheet2 | Sheet3 | KEY

Figure 5: "KEY" worksheet for example problem

In addition, however, you want to ensure that the students perform the calculations in a way that takes advantage of Excel programming. Specifically, you want them to use the SUM function with a range argument in cells B10 and C10; e.g., for B10:

=SUM(B7:B9)

and to use a formula that uses an absolute cell reference in cells C7:C9; e.g., for cell C7:

=B7/$B$10

You also want to ensure that they enter the correct values into cells B6:B8.

To satisfy these demands, the following VBA statements are created in the subroutine CheckTask:

CheckCell("B7", "IsBlank", "BadVal:0.01")
CheckCell("B8", "IsBlank", "BadVal:0.01")
CheckCell("B9", "IsBlank", "BadVal:0.01")
CheckCell("B10", "IsBlank", "NoFor", "NoRng", "NoFun:SUM", "NoRef:B6,B8", "BadVal:0.01")
CheckCell("C7", "IsBlank", "NoFor", "NoOp", "NoAbs", "NoRef:B7,B10", "BadVal:0.01")
CheckCell("C8", "IsBlank", "NoFor", "NoOp", "NoAbs", "NoRef:B8,B10", "BadVal:0.01")
CheckCell("C9", "IsBlank", "NoFor", "NoOp", "NoAbs", "NoRef:B9,B10", "BadVal:0.01")
CheckCell("C10", "IsBlank", "NoFopr", "NoRng", "NoFun:SUM", "NoRef:C6,C8", "BadVal")

The instructions employ the tolerance option for "BadVal" to account for potentially different sources for the population numbers entered into cells B7:B9, which will propagate into the results for the calculated cells B10 and C7:C9.

Suppose the student enters the following numbers and formulas into the spreadsheet as shown in Figure 6 (I've selected "Show Formulas" to display the formulas in each cell):

	A	B	C	D	E
1	**Total Score**	**First Name**		**Last Name**	
2	**0**	Hermione		Granger	
3					
4	**TASK #1**				
5		Population			
6		(millions)	%		
7	USA	312	=B7/437		
8	Canada		=B8/B10		
9	Mexico	125	=B9/$B$10		
10	TOTAL	=SUM(B7:B9)	=SUM(C7,C8,C9)		
11					
12					

Answers Sheet2 Sheet3 KEY

Figure 6: "Answer" worksheet with input answers and formulas revealed

Upon clicking the blue box, "Click to Check Work", this is what the student will see (Figure 7):

	A	B	C	D	E	F	G	H	I	J	K	L
1	**Total Score**	**First Name**		**Last Name**		**Click to Check Work**						
2	**0**	Hermione		Granger								
3												
4	**TASK #1**											
5		Population										
6		(millions)	%									
7	USA	312.0	71.4%									
8	Canada		0.0%									
9	Mexico	125.0	28.6%									
10	TOTAL	437.0	100.0%									
11												
12												

Some cells are empty [CTR-b for help]
Some cells should use a range [A:A] [CTR-g for help]
Some cells should use absolute cell references [$] [CTR-a for help]
Some cells are lacking reference to another cell, or have a bad reference [CTR-r for help]
Some cells have the wrong value [CTR-w for help]

Answers Sheet2 Sheet3 KEY

Figure 7: CPAFG spreadsheet response to inputs in Figure 6

Note that even though the value entered for the population of the USA in cell B7 (312) differs from that given in the key (313.8), it is within the tolerance (1%) and is accepted.

There are a number of sources for the errors the CPAFG spreadsheet reports:

- The student failed to enter a population for Canada in cell B8;
- The population of Mexico in cell B9 is outside the range of permissible entries given the tolerance of 0.01: 112.4 ± 1.7;
- Because both the population of Mexico is wrong and no value is entered for Canada, the total population in cell B10 is wrong;
- The formula in cell C7 does not reference cell B10; instead it uses the value found in that cell (437);
- The formula in cell C8 references B10, not $B$10. Although this will work, it's bad practice because this formula cannot be copied to other cells and retain its meaning;
- The formula in cell C9 is correct, but because the values in the cells it references are wrong, it too has the wrong value;
- The formula in cell C10 doesn't use a range. Although this works fairly easily in this example, clearly this will not be a good choice when there are more cells to sum.

It's instructive to consider the kinds of inputs the CheckCell statement above will reject for cell B10, even if the value is correct (Table 2).

Table 2: Rejected Inputs for Cell B10 Despite Correct Value

Input	Why rejected
460.7	Entry is a number, not a formula
=460.7	There's no range, no SUM function, and no cell reference
=313.8 + 34.5 + 112.4	Lacks cell references, SUM function
=B7 + B8 + B9	Lacks SUM function
=SUM(313.8, 34.5, 112.4)	Lacks a range and lacks cell references
=SUM(B7, B8, B9)	Lacks a range

Once the student has corrected all mistakes, pressing the blue box will bring up a congratulatory message (Figure 8):

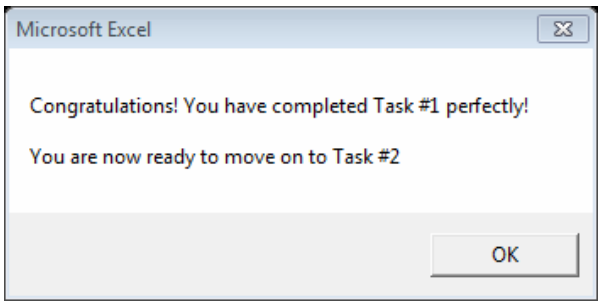


Figure 8: Box that appears when Task #1 is completed perfectly

At this point a second task would be revealed beneath the first, and the entire process starts again.

6. Discussion

Automatically generated feedback in spreadsheets is not a new idea [10, 11], nor is the use of macros to automate complex spreadsheet tasks [12, 13]. However, the CPAFG spreadsheet activities described here have two new advantages. First, unlike methods that use conditional formatting [10, 11], which is based on the value of cells, the CPAFG spreadsheets can generate automatic feedback by evaluating the formulas used to generate numbers, not just the numbers themselves. Second, by releasing tasks conditionally—upon successful completion of the previous one—students are forced to develop their skills progressively, much as the instructor intended.

The first advantage is significant. As noted by Baker and Sugden, spreadsheets are a valuable educational tool because they promote abstract reasoning and "require that users become rule-makers" [14]. Consider, for example, the task of computing a percentage in cell C7 of Figure 4. The correct formula,

=B7/$B$10

creates a mathematical function that operates on surrounding cells. To construct it, the user requires a firm understanding of absolute and relative cell references, and an ability to abstract the needed expression for cell C7 to a general rule applicable to any other cell. When the student copies this formula to N additional cells (e.g., to the 2 cells C8 and C9 in the example in Figure 4) he or she is essentially performing a logical programming operation equivalent to the "for loop":

for [counter] = 1 to N do . . . [function]

The higher-order thinking required to create this abstraction is a far cry from the skill needed to simply compute the percentage in C7 using basic math:

=313.8/463

This difference emphasizes the importance of encouraging students to view Excel as a programming tool instead of simply a form for entering data. By accepting a correct answer that was obtained in the 'wrong' way the instructor is not taking advantage of the inherent power of Excel to develop logical reasoning skills. The formula-based feedback in the CPAFG spreadsheets helps students develop a sense of symbolic abstraction that is needed to understand much of math and science.

To assess the effectiveness of the CPAFG method, I compared the scores of students completing the traditional and CPAFG spreadsheets as an activity in Hazards of the Earth's Surface, an online undergraduate service course at the University of South Florida. Prior to 2011 only the traditional spreadsheets were used; in 2011 students were given a choice, and in 2012 the CPAFG spreadsheets were the only option. The results are shown in Table 3. Compared to their cohorts who completed the traditional modules, students who completed the CPAFG modules performed significantly better ($p \leq 0.02$) on all of them, from the most basic (Hazard-Risk, Part I) to the most complex (Volcanoes, Part II). In fact, the modal score for the group of CPAFG modules was 100%, achieved by 87% of the students.

Table 3: Student Outcomes for CPAFG spreadsheets vs. Traditional SSAC Equivalents

Module	Traditional Mean % Score (std dev)	Traditional *n*	CPAFG Mean % Score (std dev)	CPAFG minimum score* (%)	CPAFG *n*	p-value#
Hazard-Risk I	84.6 (15.5)	23	99.6. (2.7)	80	57	<0.001
Hazard-Risk II	78.7 (15.8)	25	96.6 (8.0)	73	51	<0.001
Lithospheric Dens.	78.7 (14.8)	13	99.7 (1.1)	80	13	<0.001
Earthquakes I	83.9 (11.7)	18	98.9 (3.8)	73	16	<0.001
Volcanoes I	78.0 (18.0)	12	95.5 (7.2)	73	10	0.008
Volcanoes II	77.2 (23.3)	10	99.1 (3.0)	67	11	0.02
TOTAL (6 activities)	80.3 (16.6)	104	98.2 (5.5)		167	

*The minimum score represents the total value of the required preliminary tasks; the remaining points are earned by completing the final task properly. #The p-value was computed from a two-tailed t-test of the null hypothesis that the difference between the two means was due entirely to chance, and is very small except for Volcanoes II, which had a very high variance for the traditional module.

This result was expected, since the CPAFG spreadsheets both enable and guarantee high scores. Because students must successfully complete the preliminary tasks before moving on to the final task, each module has a minimum score (Table 3), which in some cases exceeded the average score for the traditional versions. In addition, students are not only permitted but encouraged to use the automatically generated feedback on the final task to refine their answers until they were 100% correct. It is satisfying that most students made this choice[4].

The advantage offered by conditional progress can be recognized in the nature of requests for help between the two groups of students. Students working on the traditional SSAC modules rarely sought help, and when they did the questions almost always pertained to the final task, which in many cases was the only one graded. The assistance required often required the instructor to explain basic concepts—ones that were covered in the PowerPoint presentation and modelled with examples (but were not, apparently, studied by the students). In contrast, students working the CPAFG SSAC modules asked many more questions, which almost always pertained to

[4] In fact, anecdotal evidence suggests that the major reason students didn't refine the final task until it too was perfect was simply a lack of time: they were up against a deadline and admitted that they submitted the activity before they wanted to.

preliminary tasks. Students sought help as a matter of necessity—since they couldn't move on to the next task until the present one was completed perfectly. And because access to the final task was contingent on successful completion of the preliminary tasks, by the time students actually reached it they'd truly mastered the required skills and had no need to seek help.

For example, the module on lithospheric density required students to master the concept of weighted mean, which was introduced in early tasks in the context of home prices (Figures 1 and 2). Of the 13 students who chose to complete the CPAFG module, five asked questions via email. All but one of the questions referred to Task 1 and addressed the concept of weights. One student also asked a question about Task 4 (the penultimate task), but this question had more to do with the way the question was phrased in the PowerPoint presentation than with the mathematical concepts. This early task assistance was apparently sufficient to enable the students to master the material, because all but one finished the last (and hardest) task perfectly, even though none of them sought help with it.

In contrast, none of the students who chose to complete the lithosphere density spreadsheet module in its traditional format asked a single question about the assignment. These students averaged only 79% on the module, and many didn't even attempt the final task. This lack of engagement is, indeed, typical. Of the 100+ students who completed spreadsheet modules in the traditional format between 2007 and 2011, only five (5%) contacted the instructor for help, and in each case the question revealed a profound lack of understanding of the underlying mathematical concept. (Example: "I don't understand what 'frequency' means, and don't know what to enter into cell B5.") As shown in Table 3, the average score of these students was only 80.3%. The large standard deviation (16.6%) is principally due to a large number of students who received very low scores (14% scored ≤ 65% and 9% scored ≤ 50%). Such poor scores were, of course, impossible on the CPAFG spreadsheet versions.

Although the CPAFG SSAC spreadsheet modules appear to offer significant pedagogical advantages over traditional versions, these advantages come at a cost. First, VBA support for Microsoft Excel is primarily found in the PC versions. Although Office 2011 for the Mac promises VBA compatibility, testing has shown inconsistent results. Until the VBA-encoded spreadsheets can be run reliably on the Mac platform, students will be required to run them on PCs, which is inconvenient for some of them.

Second, due to a quirk of Excel, VBA macros automatically clear the "Undo" cache, so that once students have triggered the "CheckTask" macro they are unable to recover past operations. This is a minor inconvenience, but one that students consistently note in the comments about the modules.

Finally, and perhaps most significantly, CPAFG spreadsheets are not trivial to create and debug, especially for someone unfamiliar with VBA programming. The investment, however, may well be worth it for instructors planning on using a suite of spreadsheet-based activities in a class. Instructors interested in creating CPAFG spreadsheets are encouraged to examine the code found in the appendix.

References

[1] Vacher, H.L. and Lardner, E. (2010). Spreadsheets across the curriculum, 1: The idea and the resource. *Numeracy* 3(2), Article 6.
Available at: http://scholarcommons.usf.edu/numeracy/vol3/iss2/art6/

[2] Spreadsheets Across the Curriculum website.
Available at: http://serc.carleton.edu/sp/ssac_home/index.html

[3] Spreadsheets Across the Curriculum: The geology of the National Parks collection.
Available at: http://serc.carleton.edu/sp/ssac/national_parks/index.html.

[4] Hsaio, F.S.T. (1985). Micros in mathematics education—uses of spreadsheets in CAL. *International Journal of Mathematics Education in Science, and Technology* 16(6): 705-713.

[5] Lovászová, G. and Hvorecký, J. (2003). On programming and spreadsheet calculations. *Spreadsheets in Education (eJSiE)*: Vol. 1: Iss. 1, Article 3.
Available at: http://epublications.bond.edu.au/ejsie/vol1/iss1/3/

[6] Abramovich, S., Nikitina, G.V., and Romanenko, V.N. (2010). Spreadsheets and the development of skills in the STEM disciplines. *Spreadsheets in Education (eJSiE)*: Vol. 3: Iss. 3, Article 5.
Available at: http://epublications.bond.edu.au/ejsie/vol3/iss3/5/

[7] Dihoff, R., Brosvic, G.M., Epstein, M.L., and Cook, M.J. (2004). Provision of feedback during preparation for academic testing: Learning is enhanced by immediate but not delayed feedback. The *Psychological Record* 54: 207-231.

[8] How do you use SSAC? From the SSAC website.
Available at: http://serc.carleton.edu/sp/ssac/how.html

[9] See: http://www.ozgrid.com/

[10] Lehman, M.W. and Herring, C.E. (2003). Creating interactive spreadsheets to provide immediate feedback. *Journal of Accounting Education* 21(4): 327-337.

[11] Wetzel, L.R. and Whicker, P.J. (2007). Quick Correct: A method to automatically evaluate student work in MS Excel spreadsheets. *Spreadsheets in Education (eJSiE)*: Vol. 2: Iss. 3, Article 1.
Available at: http://epublications.bond.edu.au/ejsie/vol2/iss3/1

[12] Yamani, A. and Kharab, A. (2001). Use of a spreadsheet program in electromagnetics. *IREE Transactions on Education* 44(3): 292-297.

[13] Oke, S.A. (2004). Spreadsheet applications in engineering education: A review. *International Journal of Engineering Education* 20(6): 893-901.

[14] Baker, J. and Sugden, S.J. (2007). Spreadsheets in education—The first 25 years. *Spreadsheets in Education (eJSiE)*: Vol. 1: Iss. 1, Article 2.
Available at: http://epublications.bond.edu.au/ejsie/vol1/iss1/2/