**CACIE Tool #09** – ***patchbowl***

**Version** **1.0**

**QA**: **QA**

1. **Description and Purpose**

The CACIE-patchbowl script modifies STOMP soil zonation files to patch holes in the silt layers of the perching silt layer in the 200 E Area. This script will read an initial zonation file containing index codes representing the materials in the model and output a new zonation file that replaces cells neighboring the perching silt with the appropriate material code (defined by the user).

1. **Functional Requirements**

The following are the functional requirements of the patchbowl tool:

FR-1: Read the header of a STOMP plot file to obtain the number of nodes in the <i>, <j>, <k> directions of the STOMP model

*Acceptance Criteria –* test demonstrates that the software has correctly parsed the plot file provided to obtain the number of nodes in the <i>, <j>, <k> directions, respectively

FR-2: Accept user inputs:

* + - file path with file name (for input file)
    - material code number 1 (integer)
    - material code number 2 (integer)
    - material code number 3 (integer)
    - file path with file name (for output file)

*Acceptance Criteria –* test demonstrates that the software has accepted the user inputs

FR-3: Read the input zonation file that is to be “patched” or modified according to the material codes provided by the user.

*Acceptance Criteria* – test demonstrates that the script has read the input zonation file indicated

FR-4: A buffer of material 2 is used to replace material 3 where material 3 is adjacent to material 1 and in a layer equal to or immediately lower than that of material 1 (“adjacent” meaning cells that share a grid cell face).

*Acceptance Criteria* – test demonstrates that the script has appropriately reassigned the input material codes according to the conditions identified.

FR-5: Output file with new zonation

*Acceptance Criteria* – test demonstrates that a new file is created with updated zonation

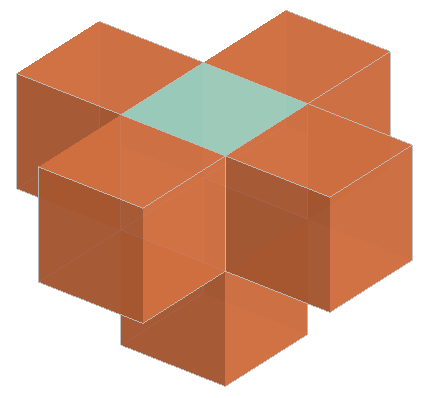
1. **Software Requirements Specifications**

The programming language used for this script was Perl. No external libraries or packages or other scripting/coding languages were used in this script. This script was developed on Linux in Perl v5.18.2.

1. **Software Design Description**

This script reads the STOMP grid definition from a plot file, loads in the input zonation file, and accepts arguments for 3 material code numbers (referred to in general as material 1, material 2, and material 3). Using the grid definition, a multiarray is built of the input zonation file.

Whenever the code encounters material 1 in a cell, the script will check the 4 cells in the same layer that share faces with the cell and will also check the cell that shares the bottom face in the next layer down. Any of those 5 cells that have material 3 as their soil type will be replaced with material 2, if they do not have material 3 then they will be left as is. Refer to Figure 1 for a representation of which cells are potentially modified by the script when the conditions are met. The modified zonation file is exported as a space-delimited text file.



Orange

Green

**Figure 1. Potential Cells to be Modified (Orange) in Relation to Cells Matching Material 1 (Green)**

Required inputs to execute this script are first shown in an example command line statement, then described in detail reading the statement from left-to-right:

$ perl [path/to/directory]/ca-patchbowl.pl plot.1 input.zone 2 3 4 output.zone

* “perl”: invokes the Perl environment on the system
* “[path/to/directory]/ca-patchbowl.pl”: the full or relative path of the patchbowl script
* “plot.1”: a STOMP “plot” file
* “input.zone”: this is a text file representing the material assignment to each STOMP grid cell in the model domain. The file format is space-delimited integers.
* “2 3 4”: these numbers are the indices representing materials 1, 2, and 3, respectively.
* “output.zone”: the new output zonation file

Code review was done by Dennis Fryer and notes from that review are found in Appendix B.

1. **Requirements Traceability Matrix**

The requirements traceability matrix for the CA-patchbowl script is presented in Table 1.

| **Table 1. Requirements Traceability Matrix** | | |
| --- | --- | --- |
| **Functional Requirement ID** | **Acceptance Test ID** | **Test Case** |
| QA Level | CACIE-patchbowl -IT-1 | Installation Test |
| FR-1 | CACIE- patchbowl -TC-1 | On execution, the log file should have the following lines:   * nx,ny,nz = 5, 5, 26 |
| FR-2 | CACIE- patchbowl -TC-2 | On execution, the log file should have the following line:   * Mat1 = 2, Mat2 = 3 |
| FR-3 | CACIE- patchbowl -TC-3 | On execution, the log file should have the following lines:   * Zone file count = 650 * IJK count = 650 |
| FR-4 | CACIE-patchbowl-TC-4 | Outputs a new text file (e.g. *output.zone*) |
| FR-5 | CACIE- patchbowl -TC-4 | Verify that the number of materials patched is recorded in the log and corresponds with the changed values in the *output.zone* file (compared against the *input.zone* file). |

1. **Test Plan and Cases**

The test plan for the ca-patchbowl script is presented in Table 2.

| **Table 2.**  **patchbowl Installation and Acceptance Test Plan** | | | |
| --- | --- | --- | --- |
| **patchbowl [Installation or Acceptance] Testing**  **CACIE-patchbowl – [IT or AT]-##** | | **Date:** | |
| **Tool Runner File Location for this test:**  */opt/tools/pylib/runner/runner.py* | | **Test Performed By:** | |
| **Testing Directory: <** [**\\olive\backups\CAVE\CA-CIE-Tools-TestEnv\patchbowl\_test**](file:///\\olive\backups\CAVE\CA-CIE-Tools-TestEnv\patchbowl_test) **>** | | | |
| **Test Step** | **Test Instruction** | **Expected Result** | **Test Result  (Pass/Fail)** |
| **CACIE-patchbowl – IT-1** | | | |
| Navigate to the testing directory | | | |
| 1 | Invoke Tool Runner and test installation of the tool as follows:  *./runner\_run\_IT-1\_ca-patchbowl.pl.sh*.  Perform this command in a terminal logged into the Linux cluster (indicated in the testing directory). | | |
| 2 | Verify Tool Runner is invoked and executed. | Verify that the Tool Runner log is generated. |  |
| 3 | Verify Patchbowl tool is invoked and executed. | Verify that the print statement “Can’t open file No such file or directory” is recorded in the *runner\_run\_IT-1\_screen.log* file (should be the 9th line of the file). |  |
| **CACIE-patchbowl – TC-1** | | | |
| Navigate to the Testing Directory | | | |
| 1 | Invoke the tool using the following command: *./ca-patchbowl.pl\_TC-1.sh.* Perform this command in a terminal logged into the Linux cluster (indicated in the testing directory). | | |
| 2 | Open the *ca-patchbowl.pl\_TC-1\_screen.log* and verify that the printed statements match what is expected. | The log file should contain the following lines of text:   * nx,ny,nz = 5,5,26 |  |
| **CACIE-patchbowl – TC-2** | | | |
| Navigate to the Testing Directory | | | |
| 1 | Open the *ca-patchbowl.pl\_TC-1\_screen.log* and verify that the printed statements match what is expected. | The log file should contain the following line of text:   * Mat1 = 2, Mat2 = 3 |  |
| **CACIE-patchbowl – TC-3** | | | |
| Navigate to the Testing Directory | | | |
| 1 | Open the *ca-patchbowl.pl\_TC-1\_screen.log* and verify that the printed statements match what is expected. | The log file should contain the following lines of text:   * Zone file count = 650 * IJK count = 650 |  |
| **CACIE-patchbowl – TC-4** | | | |
| 1 | Open the *ca-patchbowl.pl\_TC-1\_screen.log* and verify that the printed statements match what is expected. | The log file should contain the following lines of text:   * Number of Patched Materials = 11 |  |
| 2 | Open the *output.zone* file and verify that the number of instances of the integers provided match the expected number of instances.  The count of a given integer in this file can be found via a text editor or spreadsheet (Word, Excel or Notepad ++ will work) and perform a search on each individual soil index indicated and verify that the count matches the expected number (shown in the next cell). If using Excel, then you can use the “Text to Columns” command delimiting by spaces and use a “Countif()” formula to obtain the number of cells of a given soil type (e.g. =COUNTIF($A$1:$Y$26,"=1")). | The number of instances of materials based on their integers should be the following:   |  |  | | --- | --- | | Soil Index | Count | | 1 | 100 | | 2 | 17 | | 3 | 180 | | 4 | 253 | | 5 | 100 | |  |
| 3 | Open the input.zone file and verify the number of instances of each soil index (refer to step 2 for suggestions about how to do this). The expected counts are shown in the next cell. | The number of instances of materials based on their integers should be the following:   |  |  | | --- | --- | | Soil Index | Count | | 1 | 100 | | 2 | 17 | | 3 | 169 | | 4 | 264 | | 5 | 100 | |  |
| 4 | Take the difference between the counts of Soil Index 4 in input.zone vs output.zone. Verify that this number matches with the print statement identified in Step 1. | *input.zone* – *output.zone* (Soil Index 4) = 11 |  |
| 5 | Repeat Step 4, but for Soil Index 3. The number should be same | *input.zone – output.zone* (Soil Index 3) = 11 |  |

1. **Acceptance Test Report**

To complete the Acceptance Testing use Appendix A. The test cases are described as follows:

* The tool runner was invoked and the patchbowl script resolved to “TEST” in the log file (see Figure 4).
* Acceptance Test 1 is in Table A-1. It passed. The toolrunner was invoked by running “ca-patchbowl.pl\_TC-1.sh”, and the two files indicated, ca-patchbowl.pl\_TC-1\_logfile.txt and ca-patchbowl.pl\_TC-1\_screen.log, were created. The expected text is in the ca-patchbowl.pl\_TC-1\_screen.log. Screenshots of this execution process and text file verification are shown in Figure 2 -Figure 4.
* Acceptance Test 2 is in Table A-2. It passed. The tool “ca-patchbowl.pl\_TC-1.sh”, when invoked, created ca-patchbowl.pl\_TC-1\_screen.log, and it contained the expected text indicated by the test case. A screenshot of ca-patchbowl.pl\_TC-1\_screen.log is shown in Figure 3.
* Acceptance Test 3 is in Table A-3. It passed. The expected lines of text were found in ca-patchbowl.pl\_TC-1\_screen.log. A screenshot of ca-patchbowl.pl\_TC-1\_screen.log is shown in Figure 3.
* Acceptance Test 4 is in Table A-4. It passed. The expected lines of text were found in ca-patchbowl.pl\_TC-1\_screen.log. A screenshot of ca-patchbowl.pl\_TC-1\_screen.log is shown in Figure 3. The files output.zone and input.zone were found to have the correct number of instances of materials based on their integers. Screenshots of the process of checking these files are found in Figure 6 - Figure 8.

Details of these tests, when they were conducted, by whom, and if they Passed or Failed are present in each table in Appendix A.

1. **User Guide**

The following steps may be helpful for using this script:

* 1. If the model has a “perching silt layer” (ex: B Complex), run steady state simulation for one (1) time-step to create a plot file (e.g. *ss/plot.1*)
  2. Provide inputs to the script as described in the Software Design Description (Section 4), the following guidelines apply:
     1. The plot file should correspond with the model you want to change the zonation for, hence the need to run the model for a single time step to generate a plot file. Use this plot file for the patchbowl script.
     2. Provide the file path and name to the zonation card to be modified (e.g. *ss/input.zone*)
     3. The typical soil text symbol that needs to be patched in the B Complex model (as an example) is “ccu2”. Find the corresponding soil property index for the soil type you want to patch holes for (should correspond with the *ss/input.zone* file you provide)
     4. The material typically used to patch “ccu2” is “ccu3” (example for B Complex). Find the corresponding material code number that is used in the *ss/input.zone* file.
     5. The material that typically is replaced in the B Complex model example has a text symbol of “ccug”. As with the other two materials mentioned in previous bullets, make sure that the index corresponds with the material to be replaced in the *ss/input.zone* file.
     6. Provide an output file path and file name that is unique from the *ss/input.zone* file.
  3. Execute the patchbowl script using the command line instructions provided in the Software Design Description (Section 4) using the parameters described in the previous bullets.
  4. The resulting output file represents the modified zonation card that patches the selected material type by applying a buffer, as described in the Software Design Description (Section 4).

**Appendix A**

**Acceptance Testing Logs**

| **Table A-1.**  **patchbowl Acceptance Test Case 1** | | | |
| --- | --- | --- | --- |
| **patchbowl Acceptance] Testing**  **CACIE-patchbowl – AT-1** | | **Date: January 28, 2020** | |
| **Tool Runner File Location for this test:** | | **Test Performed By: Mary Weber** | |
| **Testing Directory: <** [**\\olive\backups\CAVE\CA-CIE-Tools-TestEnv\patchbowl\_test**](file:///\\olive\backups\CAVE\CA-CIE-Tools-TestEnv\patchbowl_test) **>** | | | |
| **Test Step** | **Test Instruction** | **Expected Result** | **Test Result  (Pass/Fail)** |
| Navigate to the Testing Directory | | | |
| 1 | Invoke the tool using the following command: *./ca-patchbowl.pl\_TC-1.sh.* Perform this command in a terminal logged into the Linux cluster (indicated in the testing directory). (See Figure 2) | | |
| 2 | Open the *ca-patchbowl.pl\_TC-1\_screen.log* and verify that the printed statements match what is expected. | The log file should contain the following lines of text:   * nx,ny,nz = 5,5,26 | PASS  See Figure 3, line 7 says nx,ny,nz = 5, 5, 26 |

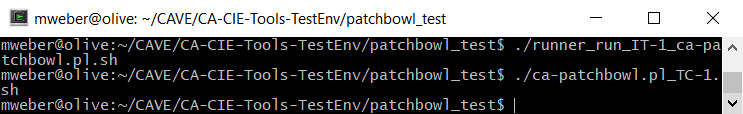


Figure . Screenshot of the Cygwin terminal after running "./ca-patchbowl.pl\_TC-1.sh"

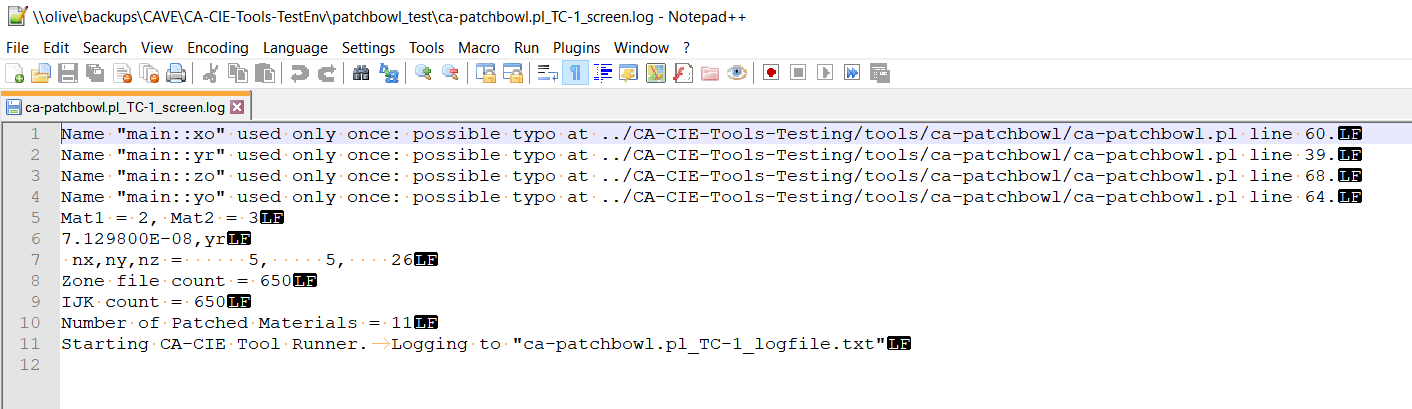


Figure . Screenshot of ca-patchbowl.pl\_TC-1\_screen.log

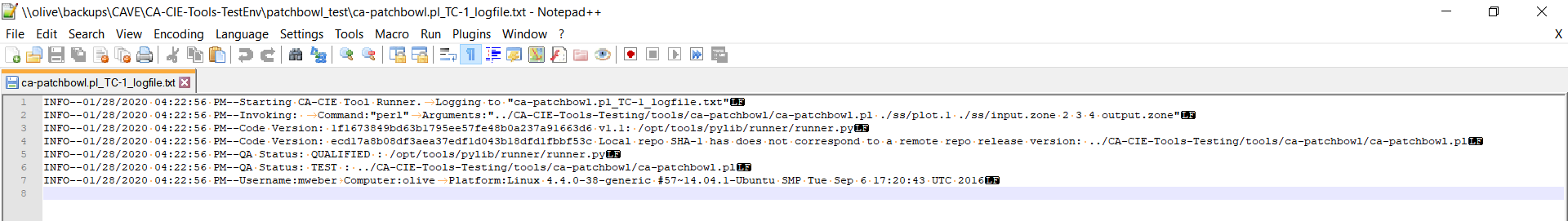


Figure . A screenshot of ca-patchbowl.pl\_TC-1\_logfile.txt (Log File from Tool Runner)

| **Table A-2.**  **patchbowl Acceptance Test Case 2** | | | |
| --- | --- | --- | --- |
| **patchbowl Acceptance] Testing**  **CACIE-patchbowl – AT-2** | | **Date: January 28, 2020** | |
| **Tool Runner File Location for this test:**  */opt/tools/pylib/runner/runner.py* | | **Test Performed By: Mary Weber** | |
| **Testing Directory: <** [**\\olive\backups\CAVE\CA-CIE-Tools-TestEnv\patchbowl\_test**](file:///\\olive\backups\CAVE\CA-CIE-Tools-TestEnv\patchbowl_test) **>** | | | |
| **Test Step** | **Test Instruction** | **Expected Result** | **Test Result  (Pass/Fail)** |
| Navigate to the Testing Directory | | | |
| 1 | Open the *ca-patchbowl.pl\_TC-1\_screen.log* and verify that the printed statements match what is expected. | The log file should contain the following line of text:   * Mat1 = 2, Mat2 = 3 | PASS  See Figure 3, line 5 contains: Mat1 = 2, Mat2 = 3 |

| **Table A-3.**  **patchbowl Acceptance Test Case 3** | | | |
| --- | --- | --- | --- |
| **patchbowl Acceptance] Testing**  **CACIE-patchbowl – AT-3** | | **Date:** | |
| **Tool Runner File Location for this test:**  */opt/tools/pylib/runner/runner.py* | | **Test Performed By:** | |
| **Testing Directory: <** [**\\olive\backups\CAVE\CA-CIE-Tools-TestEnv\patchbowl\_test**](file:///\\olive\backups\CAVE\CA-CIE-Tools-TestEnv\patchbowl_test) **>** | | | |
| **Test Step** | **Test Instruction** | **Expected Result** | **Test Result  (Pass/Fail)** |
| Navigate to the Testing Directory | | | |
| 1 | Open the *ca-patchbowl.pl\_TC-1\_screen.log* and verify that the printed statements match what is expected. | The log file should contain the following lines of text:   * Zone file count = 650 * IJK count = 650 | PASS  See Figure 3, lines 8 and 9 contain:  Zone file count = 650  IJK count = 650 |

| **Table A-4.**  **patchbowl Acceptance Test Case 4** | | | |
| --- | --- | --- | --- |
| **patchbowl Acceptance] Testing**  **CACIE-patchbowl – AT-4** | | **Date: January 28, 2020 and January 29, 2020** | |
| **Tool Runner File Location for this test:**  */opt/tools/pylib/runner/runner.py* | | **Test Performed By: Mary Weber** | |
| **Testing Directory: <** [**\\olive\backups\CAVE\CA-CIE-Tools-TestEnv\patchbowl\_test**](file:///\\olive\backups\CAVE\CA-CIE-Tools-TestEnv\patchbowl_test) **>** | | | |
| **Test Step** | **Test Instruction** | **Expected Result** | **Test Result  (Pass/Fail)** |
| Navigate to the Testing Directory | | | |
| 1 | Open the *ca-patchbowl.pl\_TC-1\_screen.log* and verify that the printed statements match what is expected. | The log file should contain the following lines of text:   * Number of Patched Materials = 11 | PASS  See Figure 3, line 10 contains:  Number of Patched Materials = 11 |
| 2 | Open the *output.zone* file and verify that the number of instances of the integers provided match the expected number of instances.  The count of a given integer in this file can be found via a text editor or spreadsheet (Word, Excel or Notepad ++ will work) and perform a search on each individual soil index indicated and verify that the count matches the expected number (shown in the next cell). If using Excel, then you can use the “Text to Columns” command delimiting by spaces and use a “Countif()” formula to obtain the number of cells of a given soil type (e.g. =COUNTIF($A$1:$Y$26,"=1")). | The number of instances of materials based on their integers should be the following:   |  |  | | --- | --- | | Soil Index | Count | | 1 | 100 | | 2 | 17 | | 3 | 180 | | 4 | 253 | | 5 | 100 | | PASS  Imported output.zone into excel, used “Text to Columns” to separate them as space-delimited, deleted the first column because it was empty.  Saved the file as “output\_zone\_check.xlsx”  See Figure 5 |
| 3 | Open the ./ss/input.zone file and verify the number of instances of each soil index (refer to step 2 for suggestions about how to do this). The expected counts are shown in the next cell. | The number of instances of materials based on their integers should be the following:   |  |  | | --- | --- | | Soil Index | Count | | 1 | 100 | | 2 | 17 | | 3 | 169 | | 4 | 264 | | 5 | 100 | | PASS  Imported output.zone into excel, used “Text to Columns” to separate them as space-delimited, deleted the first column because it was empty.  Saved the file as “input\_zone\_check.xlsx”  See Figure 6 |
| 4 | Take the difference between the counts of Soil Index 4 in input.zone vs output.zone. Verify that this number matches with the print statement identified in Step 1. | *input.zone* – *output.zone* (Soil Index 4) = 11 | PASS  See Figure 7 |
| 5 | Repeat Step 4, but for Soil Index 3. The number should be same but reverse in sign. | *input.zone – output.zone* (Soil Index 3) = -11 | PASS  See Figure 8 |

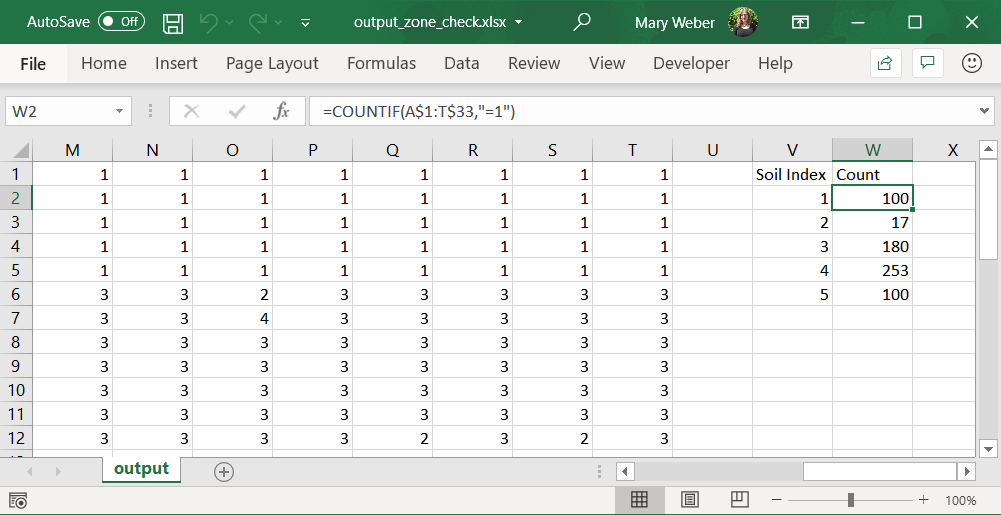


Figure . Count of each of the soil types in output.zone, as saved in output\_zone\_check.xlsx

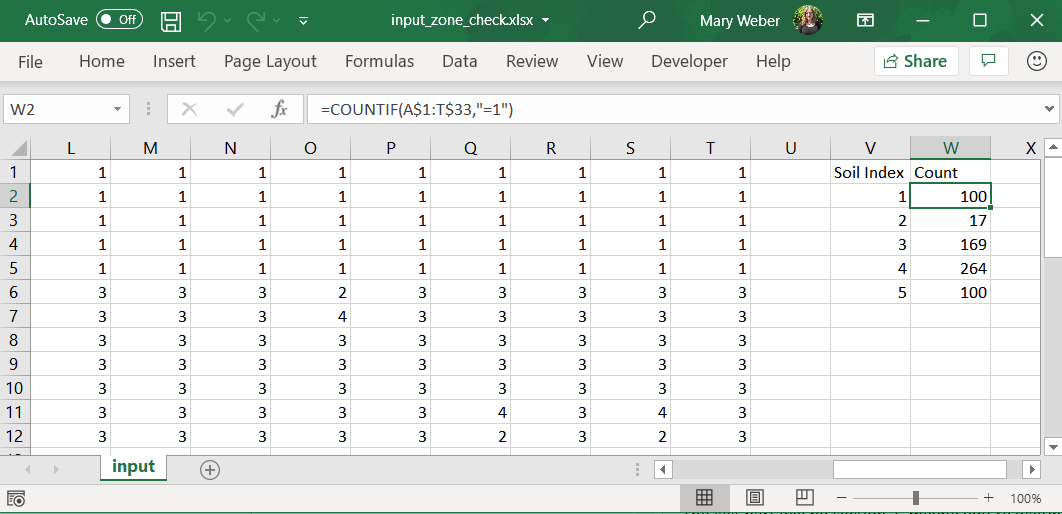


Figure . Count of each of the soil types in input.zone, as saved in input\_zone\_check.xlsx

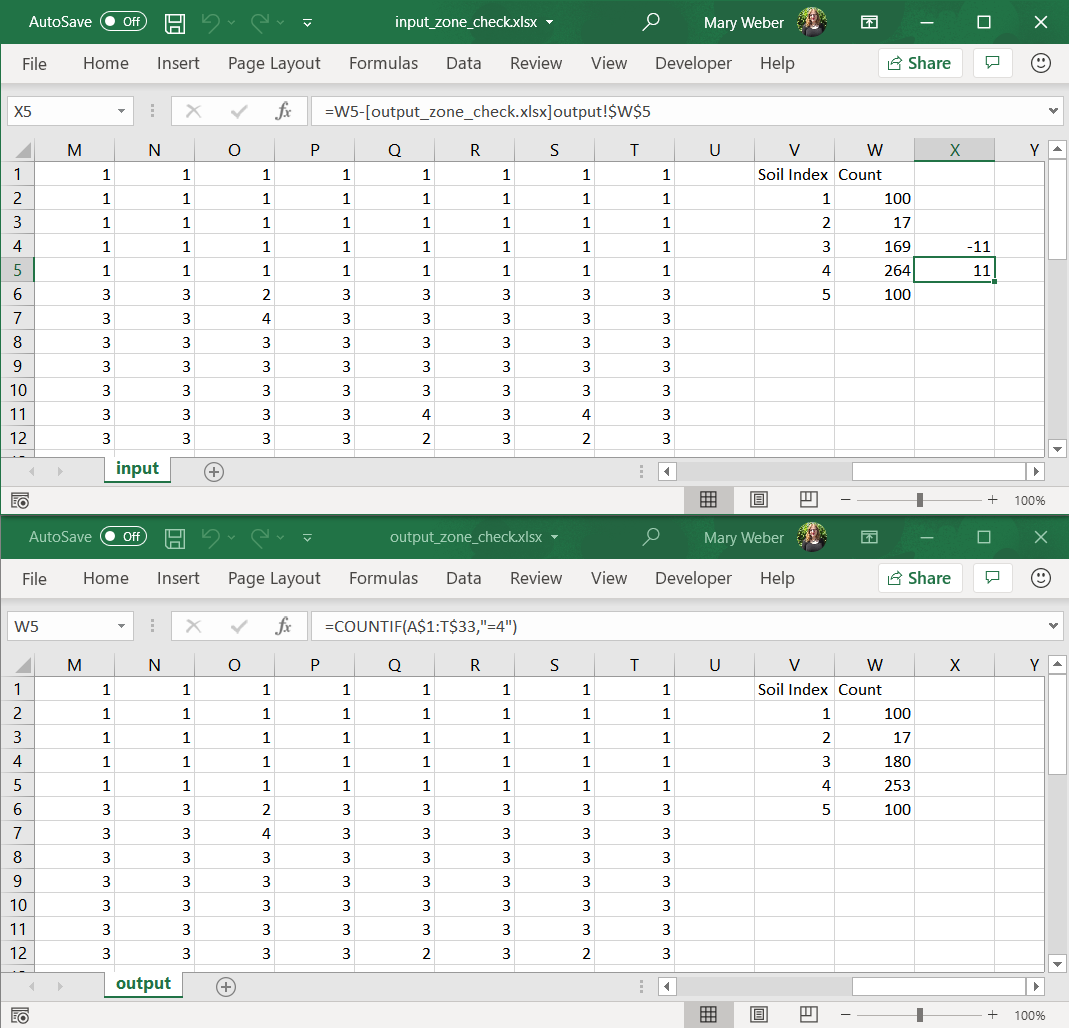


Figure . Screenshot of input\_zone\_check.xlsx, showing the math indicating the difference between the count of Soil Index type 4 in input.zone and output.zone

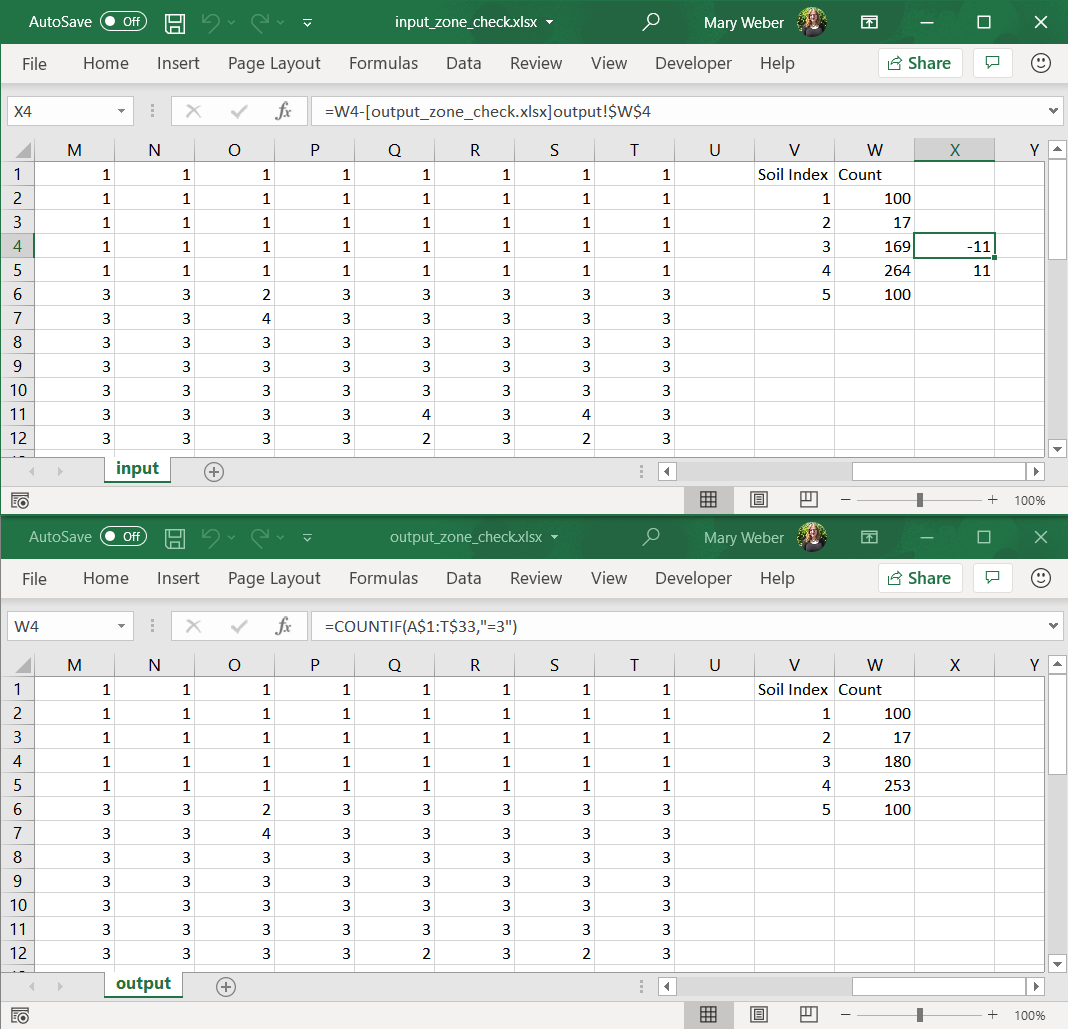


Figure . Screenshot of input\_zone\_check.xlsx, showing the math indicating the difference between the count of Soil Index type 5 in input.zone and output.zone

**Appendix B**

**Code Review Summary**

| **Table B-1. ca-patchbowl Code Review Summary** | | | |
| --- | --- | --- | --- |
| **Code Line** | **Comment** | **Notes** | **Resolution** |
| 16 | User feedback | Material 3 is not printed to the screen | Handle in next revision |
| 39 | Remove dead code | Year is extracted and printed, but not used for anything | Handle in next revision |
| 55 | Remove dead code | Origin of the model domain is parsed, but never used | Handle in next revision |
| 98 | Conditional without obvious purpose, possible dead code | Not sure if it’s possible to get a negative zone number in the current modeling configuration | Handle in next revision |
| 115 | Improve code efficiency/Error Handling | The conditional statements used to isolate which cells to replace the values are seemingly redundant.  Example is if a cell == material 3, then it is not possible that the same cell is material 2.  If anything, this might indicate if we want to ensure that the user has not assigned erroneous material combinations. This should be checked for at initial execution and not in this portion of the code | Handle in next revision |