Using Python to Simplify and Automate the USDA SSURGO Database for Spatial Visualization

A Case Study: Visualizing Factors Influencing Sinkhole Formation in the Karst Counties of Virginia

Alex Todd
Graduate Research Assistant
University of Virginia | Civil Engineering
Charlottesville, VA
alt5ry@virginia.edu

Dan Plattenberger
Graduate Research Assistant
University of Virginia | Civil Engineering
Charlottesville, VA
dap2mu@virginia.edu

***Abstract*—This paper outlines the creation and usage of a Python tool to automate the construction of a simplified table for use in the visualization of risk factors associated with sinkhole formation. Using this tool, data from the United States Department of Agriculture's (UDSA) extensive Soil Survey Geographic (SSURGO) database can be readily pulled and organized into a single, simplified table that is applicable to analyzing sinkhole risk from a geotechnical engineer's perspective. The tool provides repeatability and efficiency when considering any region that contains SSURGO data. Finally, a case study involving the karst counties of Virginia is executed in order to demonstrate the effectiveness and power of the tool.**

Keywords—Python; Soil Survey Geographic Database; Sinkholes; ArcMap; Visualization; Sinkholes; Risk

I. INTRODUCTION

The United States Department of Agriculture's (USDA) Soil Survey Geographic (SSURGO) database is a complicated and thorough database containing information about soils that has been collected over the past century by National Cooperative Soil Survey scientists and researchers. The database covers nearly the all of the United States and Territories, Commonwealths, and Island Nations served by the USDA National Resources Conservation Service (NRCS) [1]. The data, which has been accumulating from field investigations dating back as far as the 1950s, is most commonly represented on a scale of 1:12,000 or 1:24,000 [2]. The information depicts analyses of component soils and their properties, parent material types, flooding frequencies, crop yields, and limitations for recreational development, among many more components, which were gathered by scientists walking over the land and visually observing the soil, followed by lab analyses performed on soil samples [1].

Data found in the SSURGO database describing the regions of interest are represented based on closed polygon regions known as map units. These map units contain information characterizing areas with unique soil properties, interpretations, and productivity [1]. The data is provided in both the form of tabular and spatial map data, which can be joined together using a common key known as the map unit key, and metadata containing information on the descriptions of and methodology for obtaining the tabular information provided. The database is detailed and extensive, and navigation through it is complicated and time-consuming. This plan aims to shorten the process by isolating only tables and fields of interest and extracting them from the database into a single, condensed location to analyze.

II. MOTIVATION

The United States alone contains a total of 3007 counties, and each county with data available contains 68 tables within the SSURGO database. This sums up to approximately 204,500 SSURGO data tables available containing unique information. Often, management by landowners, townships, and counties only need small pieces of the available information for natural resource planning or infrastructure planning [1]. This study aims at creating a method to automate retrieving only relevant data from the SSURGO database through a simple and quick process in Python. The hope is that the user can adjust the input parameters to produce a single output table containing fields of interest. The code aims to select appropriate fields, perform the joins between their corresponding tables based on foreign and primary keys assigned by SSURGO, and automate the process to repeat across multiple counties.

A second important motivation behind this project is simplicity of data retrieval across operating systems and the ability to use free, open-source programs while doing so. Currently, the SSURGO database metadata suggest that tabular data be imported into Microsoft Access, a Microsoft Office program that must be purchased to use, where structured query language (SQL) coding can be written to select out desired fields and perform joins between tables. The SSURGO metadata provides a document containing SQL queries that can be used to perform simple tasks like joining tables or selecting fields from individual tables, but these queries are straightforward and do not consider detailed or challenging requests, such as selecting multiple fields from different tables and performing multiple joins all in one step across multiple counties' databases, for example. The hope is that this automated code will be a method outputting desired fields from

the SSURGO database in the fewest steps and through versatile programs that will be available as open sources to anyone wishing to follow this procedure.

III. Introducing the Case Study

To exemplify the use of this new methodology, a case study was devised where data representing certain factors that trigger sinkhole development in Virginia was to be extracted from the database tabular units. Sinkholes develop in karst terrain, or landscape underlain by carbonate bedrock that has been chemically dissolved due to acidic groundwater percolating into voids and pores in the bedrock. As the voids grow in size, they eventually reach a point where the overlying soil can no longer support its own weight, and the cover collapses, leaving behind a large funnel-shaped void on the surface known as a sinkhole. Karst terrain naturally develops in response to multiple geologic and environmental factors, such as parent material type, drainage class, and depth of soil overlying the bedrock. Since these three factors are all included as fields in the SSURGO database tables, the proposed technique can retrieve those fields specifically and produce an output table that can be joined with the spatial data to visualize the spatial spread of each field across the region of interest.

IV. Methodology

A. Study Area

The region of interest was 26 counties located in the Valley and Ridge Province of Virginia, a region known to be heavy in karst, because it is bounded on the west by the Ridge Province in West Virginia and on the east by the Blue Ridge Mountain range, ultimately locating them in a long valley containing extensive folds and fractures of carbonate (highly dissolvable) bedrock [3] developed as a result of differential weathering over geologic time [4]. Counties in the SSURGO database are represented using a code consisting of the state abbreviation followed by an assigned three-digit FIPS county code (e.g. VA015) [5].

B. Parameters of Interest

While there are a great number of factors that influence karst formation and sinkhole risk, this case study focused only upon the most prominent factors established from previous studies. These factors include depth to bedrock [6], parent material origin [7], soil component percentage, and drainage class [*personal communication, David A. Hubbard*]. Together, these data can be used by geotechnical engineers across the country to analyze specific areas of interest. However, because of the complexity and size of the data tables established by SSURGO, reaching these data manually for analysis proves tedious and difficult. Further, these factors are stored in numerous tables in the SSURGO data model. Using this python tool allows users to quickly select only the parameters of interest and output them in a more convenient, inclusive text file that is ready to import into mapping software, join with spatial data, and project onto a map.

C. Downloading Data

The tabular data from which important fields would be retrieved was downloaded from the SSURGO Web Soil Survey Downloading Platform [8]. The platform allows a user to input the state and the desired county names, and to download a zipped file containing that county's corresponding tabular and map data. Once each county has been downloaded, it is important to keep all county databases in a single directory. Because of the user input requirements in the Downloading Platform, automation for this step cannot be readily completed.

D. Automating the Data Processing in Python

Due to the large number of databases and specific tables from which data was to be extracted, automation was necessary to avoid repetitive, manual tasks and to remain time-efficient. For example, retrieving a single county's parameters of interest would require an engineer to manually open the SSURGO metadata, locate the tables of interest, find important parameters, and use SQL coding to copy them into a smaller table to analyze. While this process may be efficient for very small databases, it proves nearly impossible for ones that include extremely large quantities of data. Further, this process would need to be repeated for each county of interest, and thus proves to be inefficient when the focus is a large area region.

Instead, a Python code was produced that looped through each county and its appropriate tables, selecting only fields of interest, joining them based on foreign and primary keys (easily defined in a provided Data Model [9]), and rewriting them into a new abridged table. The purpose of the tool is to limit user input as much as possible and output a useful table for locating and defining sinkhole risk factors wherever there is SSURGO data available.

V. Results

Appendix A illustrates an example use of the tool. Included in the user input field are county codes of interest in the state of Virginia. These county codes are assigned by SSURGO. The user must also input the state abbreviation where prompted, in this case, VA. Finally, the user must input the column names of the parameters of interest. These column names are readily available from the metadata files located on the SSURGO website [10]. Once the user has input these fields, the Python code is ready to be executed.

Upon running the code, the user can navigate to the working directory in Python and view the final output table. If the tool executes without error, the output table should consist of all fields requested as input parameters, eliminating the fields only used to successfully join tables together, 'cokey' and 'copmgrpkey.' The tool will also eliminate any duplicate rows that result from SSURGO storage practices. This text file, (delimited by '|') is then ready to be joined with spatial data in software such as ArcMap in order to create a map of sinkhole risk. Example output maps using this tool are included in Figure 1.

Identify applicable sponsor/s here. If no sponsors, delete this text box. *(sponsors)*

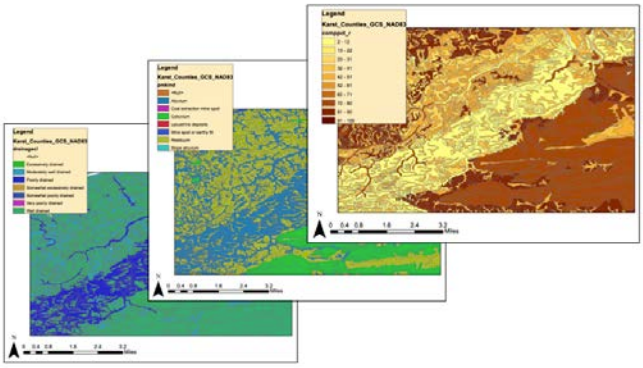

Fig. 1 *– Example output maps visualizing risk factors in sinkhole formation. Created using described Python script and ArcMap.*

VI. Discussion

This methodology should be successful in producing an output table containing only fields of interest that can be spatially joined with the map units to visualize the distribution of specific features, such as drainage class or bedrock type across a region. The apparent benefits of using Python to accomplish such a task are vast. Primarily, the tool automates tedious and repetitive tasks that would otherwise consume large amounts of time. The tool is also useful in reproducing consistent results concerning sinkhole risk, regardless of the location of interest. Finally, Python is available to everyone and operates across platforms, allowing more users to readily use and modify the tool to fit their specific needs.

A. Recommendations for the Future

It would be beneficial to add a method for extracting the tabular information directly from the Internet rather than requiring an initial download of each database containing relevant information. However, currently, this option is not available due to the required manual user input in the SSURGO Downloading Platform.

Additionally, as seen in Appendix A, the Python code currently only accounts for four different input tables, and adding a fifth table would require a quick manipulation of the code to repeat extraction through five tables instead of four. Ideally, the code would be able to detect five input table sources and adjust automatically to loop through all five without requiring manipulation of the original code. However, due to time constraints, the current results expect the user to manually alter the code by adding any additional tables required, a simple yet longer process than the initial code goal was to represent an entirely time-efficient and automated method.

Finally, the code requires the input columns to include the primary and foreign key fields so that joins can be completed in the Python code. In this version of the script, those fields are required and are later removed from the output to simplify the output table. However, ideally, the code could include a section that detects the input tables and automatically includes the corresponding primary and foreign keys to perform the joins. These are all ideas for an even more automated code, however, due to time constraints, they were considered outside the scope of this project.

Acknowledgments

The authors would like to give thanks to Dr. Jonathan Goodall for his constant encouragement and willingness to help on this project. Most scientists are aware that Python coding can be tedious and difficult to troubleshoot, and Professor Goodall always dedicated whatever time was needed to assist in the debugging process.

References

[1] "Description of SSURGO Database." Soil Survey Staff, Natural Resources Conservation Service, United States Department of Agriculture. Available online at *http://www.nrcs.usda.gov/wps/portal/nrcs/detail/soils/survey/?cid=nrcs142p2_053627*. Accessed [12/3/2014]

[2] "Soil Survey Geographic Database." Digital Coast: Office for Coastal Management. Available online at *http://coast.noaa.gov/digitalcoast/data/ssurgo*. Accessed [12/3/2014]

[3] Belo, B.P. Natural Hazard Mitigation Planning for Karst Terrains in Virginia. Master Thesis, Virginia Polytechnic Institute and State University, Blacksburg, VA, May 2003.

[4] William and Mary Department of Geology. *Valley and Ridge Province, Geology of Virginia*. Available at *http://web.wm.edu/geology/virginia/provinces/valleyridge/valley_ridge.html*. Accessed [7/15/2014]

[5] Merwade, Venkatesh. "Downloading SSURGO Soil Data from Internet." Civil Engineering, Purdue University. Available online at *https://web.ics.purdue.edu/~vmerwade/education/ssurgo.pdf*. Accesed [12/8/2014].

[6] Green, J.A., Marken, W.J., Alexander, Jr., E.C. and S.C. Alexander. Karst Unit Mapping using Geographic Information System Technology, Mower County, Minnesota, USA. In *Environmental Geology*, Volume 42, Issue 5, 2002, pp.457-461.

[7] Hubbard, Jr. D.A. Sinkhole Distribution of the Valley and Ridge Province, Virginia. In Geotechnical and Environmental Applications of Karst Geology and Hydrology: Proceedings of the Eighth Multidisciplinary Conference on Sinkholes, April 2001, pp.33-36.

[8] Soil Survey Staff, Natural Resources Conservation Service, United States Department of Agriculture. Soil Survey Geographic (SSURGO) Database for Virginia. Available at *http://websoilsurvey.nrcs.usda.gov/*. Accessed [11/15/2014].

[9] "SSURGO 2.2 Data Model: Soil Tabular and Spatial Data Tables." United States Department of Agriculture. Available online at *http://www.nrcs.usda.gov/Internet/FSE_DOCUMENTS/nrcs142p2_050900.pdf* Accessed [11/13/2014].

[10] "SSURGO 2.3.2 Tables and Columns." United States Department of Agriculture. Available online at *http://www.nrcs.usda.gov/wps/PA_NRCSConsumption/download?cid=stelprdb1241114&ext=pdf*. Accessed [11/13/2014].

Appendix A

This appendix will detail a case study in which risk factors for sinkhole formation in the karst counties of Virginia are isolated and retrieved from SSURGO databases and are then projected onto a map using spatial data in ArcMap. The majority of the tasks required for such a study are readily executed in the Python tool, described in detail below. Future users of the tool should use this case study as a guideline to prepare the code to fit their own needs.

On the right will be the full script used to compile the final output table for this case study and on the left will be descriptions of the code, including potential changes users will need to make.

Introduction (1-10)

Import Packages (11-14)
User will not likely change these inputs unless functions using other Python tools are desired

As the code currently stands, all files will reference and be written to your working directory. (16-20)

User Input (21-57)
Listed in single quotations are the counties of interest for the case study. These county numbers are defined by SSURGO. Each should be separated with a comma. (24-27)

Virginia is the State of interest so 'VA' is included in single quotations (29)

*Output_Filename**N**: This case deals with data in SSURGO tables entitled 'component,'cpmat,'muaggatt,' and 'cpmatgrp' so the output tables which will included only the data we desire from each of these tables are named similarly, as txt files. (38, 43, 48, 53) If additional data in other tables are desired, create another section similar to lines (38-41).*

*Column_Heading**N**: These will be written into the first row of the output table and columns will be joined based on them. Separate desired fields with '|' with no spaces before or after. Names are from SSURGO database and should be located in SSURGO metadata [10]. After last field, write '\n' to return to next row, no spaces.*

*ColumnNumbers**N**: From SSURGO metadata [10] corresponding to each desired field in Column_Headings**N**. SSURGO column number minus unity.*

*TableOfInterest**N**: Name of file created when SSURGO data was downloaded.*

```
1  ''' Objective:  To simplify the steps used in retrieving data from
2                  SSURGO Databases, specifically focusing on fields
3                  that might be helpful in geotechnical engineering
4      Authors:    Dan Plattenberger and Alex Todd
5                  University of Virginia (UVA), Civil Engineering
6      References: Jonathan Goodall, Ph.D., UVA
7      Sources:    United States Department of Agriculture (USDA)
8                  Soil Survey Geographic (SSURGO) Database '''
9
10 # ------------------------------------------------------------------------
11 # Import appropriate packages
12 import csv
13 import pandas as pd
14 import os
15
16 ''' NOTE: - Navigate to your working directory.
17           - Save your SSURGO county data to the working directory,
18             or change the file paths for SSURGO_Filenames in code
19 '''
20
21 # Define user input ------------------------------------------------------
22
23 # Define counties in region of interest (SSURGO County Assignments)
24 Counties = ['005', '015', '017', '021', '023', '043', '045', '069',
25     '071', '091', '105', '107', '121', '139', '155', '161', '163',
26     '167', '169', '171', '173', '185', '187', '191', '195', '197']
27
28 # Define state abbreviation for state of interest
29 State = 'VA'
30
31 # Define the specific SSURGO tables and columns you wish to output
32 # Output_Filmename = General output name for abridged table
33 # Column_Headings = Column titles separated by '|'
34 # Column_Numbers = Specific columns to select from SSURGO table
35             # (NOTE: Column Numbers = SSURGO Column Number - 1)
36 # TableOfInterest = references saved SSURGO tabular data title
37
38 Output_Filename1='Output_component.txt'
39 Column_Headings1="comppct_r|majcompflag|drainagecl|mukey|cokey\n"
40 Column_Numbers1 = [1,5, 23,107,108]
41 TableOfInterest1 = 'comp.txt'
42
43 Output_Filename2 = 'Output_cpmat.txt'
44 Column_Headings2="pmkind|pmorigin|copmgrpkey\n"
45 Column_Numbers2 = [3,4,5]
46 TableOfInterest2 = 'cpmat.txt'
47
48 Output_Filename3 = 'Output_muaggatt.txt'
49 Column_Headings3="brockdepmin|mukey\n"
50 Column_Numbers3 = [5,39]
51 TableOfInterest3 = 'muaggatt.txt'
52
53 Output_Filename4 = 'Output_cpmatgrp.txt'
54 Column_Headings4="pmgroupname|cokey|copmgrpkey\n"
55 Column_Numbers4 = [0,2,3]
56 TableOfInterest4 = 'cpmatgrp.txt'
57
```

Create Abridged Tables of Each SSURGO Table Used (59-63)
No user input

Headings Included in Tables (65-70)
Headings included in lines (39, 44, 49, and 54) are written into each abridged table.

Pulling Desired Data From Each Table (71-127)
Loops through each county and each downloaded SSURGO data to pull desired fields.

Locating tables of interest (73-84)
If user uses more SSURGO tables than included in original code, add lines similar to (76-77), -- up to N additional tables -- to line (84).

Reading each table of interest (85-94)
If user uses more SSURGO tables than included in original code, add lines similar to (86-87), -- up to N additional tables -- to line (94).

Pulling desired data from each table (95-127)
If user uses more SSURGO tables than included in original code, add lines similar to (96-102), -- up to N additional tables -- to line (127).

```
58 # ---------------------------------------------------------------------
59 # Creates 4 new files
60 Output_Table1 = open(Output_Filename1, 'w')
61 Output_Table2 = open(Output_Filename2, 'w')
62 Output_Table3 = open(Output_Filename3, 'w')
63 Output_Table4 = open(Output_Filename4, 'w')
64
65 # Adds headings to each file
66 Output_Table1.write(Column_Headings1)
67 Output_Table2.write(Column_Headings2)
68 Output_Table3.write(Column_Headings3)
69 Output_Table4.write(Column_Headings4)
70
71 # For each county, perform the following steps:
72 for County in Counties:
73
74     # Locate table of interest
75     print "Working on County " + County
76     SSURGO_Filename1= State + County + '/tabular/' + \
77         TableOfInterest1
78     SSURGO_Filename2= State + County + '/tabular/' + \
79         TableOfInterest2
80     SSURGO_Filename3= State + County + '/tabular/' + \
81         TableOfInterest3
82     SSURGO_Filename4= State + County + '/tabular/' + \
83         TableOfInterest4
84
85     # Read table of interest
86     SSURGO_Table1 = csv.reader(open(SSURGO_Filename1, 'rb'), \
87         delimiter='|')
88     SSURGO_Table2 = csv.reader(open(SSURGO_Filename2, 'rb'), \
89         delimiter='|')
90     SSURGO_Table3 = csv.reader(open(SSURGO_Filename3, 'rb'), \
91         delimiter='|')
92     SSURGO_Table4 = csv.reader(open(SSURGO_Filename4, 'rb'), \
93         delimiter='|')
94
95     #  Select appropriate columns
96     for row in SSURGO_Table1:
97         new_row = ""
98         Last_Column1 = Column_Numbers1[-1]
99         for Column_Number in Column_Numbers1[:-1]:
100            new_row = new_row + row[Column_Number] + "|"
101        new_row = new_row + row[Last_Column1] + "\n"
102        Output_Table1.write(new_row)
103
104    for row in SSURGO_Table2:
105        new_row = ""
106        Last_Column2 = Column_Numbers2[-1]
107        for Column_Number in Column_Numbers2[:-1]:
108            new_row = new_row + row[Column_Number] + "|"
109        new_row = new_row + row[Last_Column2] + "\n"
110        Output_Table2.write(new_row)
111
112    for row in SSURGO_Table3:
113        new_row = ""
114        Last_Column3 = Column_Numbers3[-1]
115        for Column_Number in Column_Numbers3[:-1]:
116            new_row = new_row + row[Column_Number] + "|"
117        new_row = new_row + row[Last_Column3] + "\n"
118        Output_Table3.write(new_row)
```

Pulling Data From Each Table Cont. (119-127)

Close Opened Files (128-133)
If user uses more SSURGO tables than included in original code, add lines similar to (129), -- up to N additional tables -- to line (133).

Open Output Tables for Joining (136-145)
If user uses more SSURGO tables than included in original code, add lines similar to (137), -- up to N additional tables -- to line (140+).
If user uses more SSURGO tables than included in original code, add lines similar to (141), -- up to N additional tables -- to line (145).

Joining Each Abridged Table to Create One Output Table (146-153)
If additional SSURGO tables were used, they will need to be joined using designated SSURGO column names, similar to line (147)

Write Primary Output CSV File (154-157)
No user input required. Separates fields with '|'

Remove Duplicate Rows from CSV (158-163)
No user input required

Optional: Delete Individual, Abridged Tables from Directory (164-167)
Comment this section to keep smaller, abridged tables created earlier.

```
119
120     for row in SSURGO_Table4:
121         new_row = ""
122         Last_Column4 = Column_Numbers4[-1]
123         for Column_Number in Column_Numbers4[:-1]:
124             new_row = new_row + row[Column_Number] + "|"
125         new_row = new_row + row[Last_Column4] + "\n"
126         Output_Table4.write(new_row)
127
128 # Closes all files
129 Output_Table1.close()
130 Output_Table2.close()
131 Output_Table3.close()
132 Output_Table4.close()
133
134 print "Done creating individual tables!"
135
136 # Read each output table
137 Muaggatt='Output_muaggatt.txt'
138 Component='Output_component.txt'
139 Cpmatgrp = 'Output_cpmatgrp.txt'
140 Cpmat = 'Output_cpmat.txt'
141 Muag = pd.read_csv(Muaggatt, delimiter='|')
142 Comp = pd.read_csv(Component,delimiter='|')
143 Cpmatg = pd.read_csv(Cpmatgrp,delimiter='|')
144 Cpm = pd.read_csv(Cpmat,delimiter='|')
145
146 # Inner join output tables based on common keys
147 MuagComp = pd.merge(Muag,Comp,how='inner', on='mukey')
148 MuagCompCpmatg = pd.merge(MuagComp,Cpmatg,how='inner',on='cokey')
149 All_WithCokey = pd.merge(MuagCompCpmatg,Cpm,how='inner',\
150     on='copmgrpkey')
151 All_WithoutCokey = All_WithCokey.drop('cokey', 1)
152 All_Final = All_WithoutCokey.drop('copmgrpkey', 1)
153
154 # Write joined tables to CSV
155 All_Final.to_csv(path_or_buf= 'Final_Table_With_Duplicates.csv', \
156     index=False, index_label=False, sep="|")
157
158 # Remove duplicates from CSV
159 Final_Output = pd.read_csv('Final_Table_With_Duplicates.csv', \
160     delimiter='|').drop_duplicates()
161 Final_Output.to_csv(path_or_buf= 'Final_Table.txt', index=False, \
162     index_label=False, sep="|")
163
164 # Delete unnecessary files from directory
165 os.remove(Muaggatt), os.remove(Component), os.remove(Cpmatgrp)
166 os.remove(Cpmat), os.remove('Final_Table_With_Duplicates.csv')
167
168 print 'Done! Check your working directory for the file.'
```

Running the previous script will output a text file, pictured in *Figure 2*. This text file may be imported into mapping software such as ArcMap and joined with spatial data acquired from SSURGO to produce risk maps suck as the ones displayed in *Figure 1*.

```
brockdepmin|mukey|comppct_r|majcompflag|drainagecl|pmgroupname|pmkind|pmorigin
|834316|80|Yes|Well drained|alluvium derived from sandstone and shale|Alluvium|Sandstone and shale
|834317|80|Yes|Well drained|alluvium derived from sandstone and shale|Alluvium|Sandstone and shale
76.0|834318|50|Yes|Well drained|residuum weathered from sandstone|Residuum|Sandstone
76.0|834318|30|Yes|Excessively drained|residuum weathered from sandstone|Residuum|Sandstone
69.0|834319|82|Yes|Well drained|residuum weathered from shale and siltstone and/or fine grained sandstone|Residuum|Shale and siltstone
69.0|834319|82|Yes|Well drained|residuum weathered from shale and siltstone and/or fine grained sandstone||Sandstone
91.0|834320|80|Yes|Well drained|residuum weathered from shale and siltstone and/or fine grained sandstone|Residuum|Shale and siltstone
91.0|834320|80|Yes|Well drained|residuum weathered from shale and siltstone and/or fine grained sandstone||Sandstone
41.0|834321|30|Yes|Well drained|residuum weathered from shale and siltstone|Residuum|Shale and siltstone
41.0|834321|55|Yes|Well drained|residuum weathered from shale and siltstone|Residuum|Shale and siltstone
41.0|834322|80|Yes|Well drained|residuum weathered from shale and siltstone|Residuum|Shale and siltstone
41.0|834322|15|Yes|Well drained|residuum weathered from shale and siltstone|Residuum|Shale and siltstone
41.0|834323|25|Yes|Well drained|residuum weathered from shale and siltstone|Residuum|Shale and siltstone
41.0|834323|70|Yes|Well drained|residuum weathered from shale and siltstone|Residuum|Shale and siltstone
41.0|834324|55|Yes|Well drained|residuum weathered from shale and siltstone|Residuum|Shale and siltstone
41.0|834324|40|Yes|Well drained|residuum weathered from shale and siltstone|Residuum|Shale and siltstone
74.0|834325|85|Yes|Well drained|residuum weathered from limestone|Residuum|Limestone
74.0|834326|85|Yes|Well drained|residuum weathered from limestone|Residuum|Limestone
74.0|834327|85|Yes|Well drained|residuum weathered from limestone|Residuum|Limestone
74.0|834328|45|Yes|Well drained|residuum weathered from limestone|Residuum|Limestone
74.0|834329|45|Yes|Well drained|residuum weathered from limestone|Residuum|Limestone
74.0|834330|35|Yes|Well drained|residuum weathered from limestone|Residuum|Limestone
74.0|834330|60|Yes|Well drained|residuum weathered from limestone|Residuum|Limestone
|834331|85|Yes|Well drained|alluvium derived from sandstone and shale|Alluvium|Sandstone and shale
|834331|2|No |Poorly drained|alluvium derived from sandstone and shale|Alluvium|Sandstone and shale
|834332|2|No |Poorly drained|alluvium derived from sandstone and shale|Alluvium|Sandstone and shale
|834332|85|Yes|Well drained|alluvium derived from sandstone and shale|Alluvium|Sandstone and shale
|834333|50|Yes|Well drained|alluvium derived from sandstone and shale|Alluvium|Sandstone and shale
|834333|2|No |Poorly drained|alluvium derived from sandstone and shale|Alluvium|Sandstone and shale
|834334|2|No |Poorly drained|alluvium derived from sandstone and shale|Alluvium|Sandstone and shale
|834334|50|Yes|Well drained|alluvium derived from sandstone and shale|Alluvium|Sandstone and shale
|834335|80|Yes|Moderately well drained|alluvium derived from sandstone and shale|Alluvium|Sandstone and shale
|834335|2|No |Poorly drained|alluvium derived from sandstone and shale|Alluvium|Sandstone and shale
|834336|30|Yes|Well drained|colluvium derived from shale, siltstone, and some fine-grained sandstone|Colluvium|Sandstone and shale
|834336|30|Yes|Moderately well drained|alluvium derived from sandstone and shale|Alluvium|Sandstone and shale
|834336|2|No |Poorly drained|alluvium derived from sandstone and shale|Alluvium|Sandstone and shale
|834336|30|Yes|Well drained|alluvium derived from sandstone and shale|Alluvium|Sandstone and shale
76.0|834337|60|Yes|Excessively drained|residuum weathered from sandstone|Residuum|Sandstone
76.0|834338|60|Yes|Excessively drained|residuum weathered from sandstone|Residuum|Sandstone
```

Fig. 2 *– Portion of final output test file for case study. Heading row lists column names. Fields separated with '|'. Ready to import into mapping software for visualization.*