9/30/2019

Faye Chant

A Data Preparation Tool with ArcPy.

GEOG5991M: Assessment 2

Table of Contents

[Introduction 1](#_Toc20654368)

[Scope 1](#_Toc20654369)

[Method 1](#_Toc20654370)

[Issues 3](#_Toc20654371)

[Testing & Further development 4](#_Toc20654372)

[Conclusions 4](#_Toc20654373)

[Reference List 4](#_Toc20654374)

# 

# Introduction

# Scope

# Method

The workflow was initiated by consulting with the main user, who had defined the scope of the tool, to detail the current working practice. The tool was built using a staged and iterative process. This meant getting the code working stage by stage. For each stage the initial code was written, tested, debugged, and the repeated as necessary building the code up slowly.

The following stages were defined:

Stage 1. Field modification. This involved two aspects:

* 1. Retain specified fields, and delete all other fields.
  2. Reformat certain fields as required.

Stage 2. Sorting and generate new ID.

Stage 3. Testing the number of points in the feature class, and if necessary splitting into smaller feature classes.

Stage 4. Graphics – create an overview plot of the point shapefile and a reference grid labelling each cell corresponding to the smaller feature classes.

The code for stage one was captured initially by creating a new Toolbox within ArcMap and adding a new model. The tools required in ArcMap were built into the model and the model exported as a script. This was edited in Notepad++ and run in the command prompt. This was due to difficulties in getting the ArcPy library installed within Spyder (see section xx for discussion).

The following Stage one steps were outlined:

1. Define the name of the feature class as entered by the user using “GetParameterAsText”.
2. Make a copy of the feature class to work on using the “CopyFeatures\_Management” tool, so the original was always retained intact.
3. Use the “ListFields” tool to extract all the field names in the feature class. Using this list and a defined list of fields to keep, a list of fields to delete was built.
4. Use the “DeleteFields\_Management” tool to delete the unwanted fields as specified in the list built.

Once this was running, this was expanded upon to re-format some of the retained fields. As it is not possible to modify the type of a field, the following steps were inserted into the workflow between steps 2 and 3 above:

1. Add new fields to the shapefile in the correct format
2. Use “CalculateField” to copy over the old fields to the correctly formatted ones.

Stage 2 was relatively simple and was carried out by using the “Sort\_management” tool to sort the feature class by the “shape” field. This uses the centroid of each feature in the feature class and results in the file being sorted west to east and north to south. The “CalculateField” tool was used again to add in a new id by defining an “auto increment” function which started at 0 and added 1 with each new record in the feature class.

Stage 3 was more complex. The “Fishnet” tool was used from ArcPy to create a grid over the feature class. This has a parameter to feed in the extents from a template file to define the extents of the fishnet. This worked in a model in ArcMap, but did not work as a script, and so the “Describe” function was used to create parameters corresponding to the grid origin, YAxis orientation, and the opposite corner of the grid. The main fishnet code was run using a while loop which counted the number of points in the shapefile, and created a fishnet starting at 2rows by 2 columns. It used a spatial join between the point feature class and the fishnet polygon feature class to count the points within each cell of the fishnet. It any of the cells contained more points than the threshold, the fishnet was repeated using more cells and this continued until all cells were below the threshold. The final fishnet was retained, and temporary files deleted.

Next the points within each cell of the final fishnet were extracted using the “Intersect” and “Select” tools. Intersect retained each point and all the corresponding fields from the cell of the fishnet within which it was found. This meant that the cell number could then be used as the “where” clause for the select tool, which saves the output to a new shapefile. This was built into another while loop, which continued while the cell id number was less than the total number of cells of the fishnet.

At this stage, the python file was quite long and so was adapted to be more object oriented. A new python file containing the framework for the stereoheighting preparation tool was created . This contained four classes which could be called in the SHPrep.py file. The class “Files” contained all objects and methods relating to files to be used in the tool, e.g. the original file name/path, the working copy etc. It also contained methods relating to manipulating these files such as printing (as a checking mechanism), or deleting temporary files at the end. The class “Fields” contained the objects and methods relating to manipulating the fields within the feature class e.g. setting the structure of the new fields, and the calculate field function. The “Features” class contained the objects and methods used to describe and derive parameters from the feature class being used. The “Fishnet” tool contained the methods used to run stage 3 of the workflow, optimising the fishnet and exporting the separate shapefiles.

Due to the issues in getting ArcPy installed on Spyder, Stage 4 was carried out as a separate python file which was written, and run in Spyder. This contained a class called Files, similar to the one described above. This used both the MatPlotLib and GeoPandas external libraries. The shapefiles were read in using the Geopandas “Read\_file” tool as this automatically reads files of many types. The “subplots” function from MatPlotLib was used to create a figure with one subplot. Both the final fishnet and the points file were plotted onto this figure. The aspects, set to be equal. The axes were labelled, and a lambda function used to plot the cell number on each grid cell of the fishnet. This plot was saved out to a .jpg file. This overview plot can be used to locate each individual cell shapefile.

Documentation for this was built using Sphinx. This has an autodoc feature which extracts docstrings written in the code files as restructured text and builds them into a series of .html files.

A UML was initiated using JetBrains PyCharm, and edited in PowerPoint.

The final UML is in Figure 1.

The project was pushed to a public repository on GitHub.

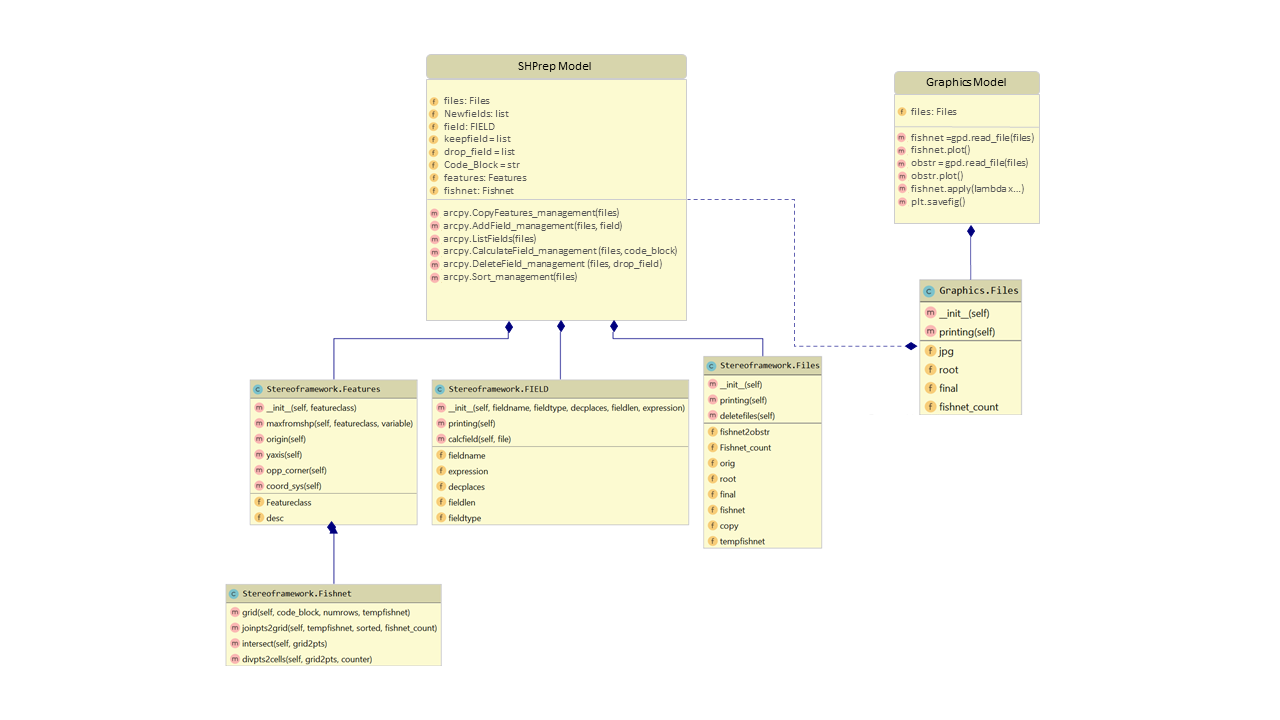
  
Issues

Figure : UML of the data preparation tool

Minor issues were encountered with compilation time errors, especially around trying to call the classes and methods from the stereoframework.py file in the SHPrepe.py file. For example getting the inheritance from the “Files” class to the “Features” class correct. Some runtime errors were encountered with the while loops. These issues were overcome with a bit of reading from books such as Phillips, 2015).

Most of the major issues with this project centred around import time errors due to having multiple versions of python installed on the computer. Anaconda installed versions 2.7 and 3.7, and ArcGIS installed ArcPy version 2.7 of Python. Spyder returns an import time error from the line”import ArcPy”. The library could not be installed or imported. This issue was not fixed, a workaround was found to have a separate code file for the graphics part of the tool which was intended run in a GUI.

A similar issue was found with Sphinx when running the build function initially. This was overcome partially by setting the system path to point at the ArcPy library, rather than the anaconda installed Python libraries. However, despite this Sphinx failed to find the Env function within ArcPy used to set the workspace, and used to build file paths. Therefore, a different version of the code file was built for Sphinx removing the code lines which imported arcpy, and set the env.workpace parameter. The “GetParameterAsText” lines were replaced with “Input().” A “rootdir = ” variable was created setting the workspace location in the “Files” class.

Another import time error was return when running the graphics script within a toolbox on ArcMAp. Geopandas could not be imported as it was not installed into the ArcGIS ArcPy library. Geopandas was installed into the Anaconda Python libraries. The graphics tool was therefore removed from the toolbox, and can be run directly on the command line, via a batch file or within Spyder. A remedy to this could possibly be to try and install geopandas into the ArcPy library to fix this.

Lastly, there were issues with getting the html pages built using sphinx onto GitHub to be read as static webpages associated with the tool repository. Managed to build a new branch on github called gh-pages. And added the html files from the \_build branch directly into the main folder, deleting anything else. This displays the webpages, but wont read the theme set in GH, so looks very basic.

# Testing & Further development

The computer system upon which this tool will run will not allow access to the command prompt, or allow .bat files to run. Therefore, it was necessary to ensure that this tool ran not only on the command line, and through a .bat file, but also as a script within a toolbox in ArcMap.

Due to the administrative constraints of the computer system upon which this tool will run, it was not possible to test this tool on real data within the timeframe. The next stage will involve user testing and further debugging. In addition, a logic test will be added at the beginning of stage one to check the type and format of specific input fields, and then only proceed to the field manipulation where necessary.

Stage four was a validation tool, required to run after the prepared point files(s) have been processed. It was not possible to complete the validation tool within the time available. However, this toolkit will continue to evolve, and this functionality will be added as a separate tool at a later date.

# Conclusions

The scope for this tool was to build a tool or set of tools to automate data preparation of specific point data supplied as a shapefile. This work was currently done manually within ArcMap. The four objectives were to:

1. Reformat point data
2. Sort and re number the id of the point data
3. Where the file contained too many points, divide the file up into smaller shapefiles.
4. Carry out specific validation checks, once the point data had been processed.

Three out of four of these were achieved, and the tool passed initial testing to run on the command line, through a batch file and as a script in ArcMap. This tool is not complete and will continue to evolve following a period of user testing.

# Reference List

Chan, J. 2017. Learn Python in one day and Learn It Well.

Garrard, C. 2016. Geoprocessing with Python.

Phillips, D. 2015. Python3. Object-oriented Programming.

<https://matplotlib.org/3.1.1/>

<http://geopandas.org/>

<https://www.sphinx-doc.org/en/master/>

<https://docs.python.org/3/>

ArcGIS 10.4.1 help files.