Python Scripting and Data Management

By Nickolas Haldeman, May 4, 2007

Data Management

Proper data management is the first hurdle to clear when creating a successful Geographic Information System (GIS) used to capture, store, analyze, and manage spatial data. By creating working folders containing geodatabases which are capable of storing raster (grid) or vector (point, line, or polygon) data, a GIS user can quickly and easily manage several large datasets and make these files easily transferable.

In the past data was distributed from one user to another in one of several formats. With the advent of the personal geodatabase, GIS file transfers have become much easier and safer. A geodatabase is easier because multiple files can be neatly store in a single database, making transfer quicker. Geodatabases are also a safer form of file transfer because the developer can assign a projection to the geodatabase which requires all the files contained inside to be the same projection. A projection is simply a method used to represent the curved surface of the earth on a plane. ESRI's ArcGIS is now able to project on the fly, meaning, it will take data that isn't necessarily in the correct spatial coordinate system and display it with data in a different projection. This may not seem like much of an issue, but when an analysis is run, having different projections on the data may mean that the results will be completely different.

The transfer of data between GIS users by geodatabase is still playing catch-up to the ever expanding world of GIS. Transferring data using geodatabases not only make the producers life easier but it also makes the users data management simpler.

Python Scripting

Python Scripting is a fairly recent development in the world of GIS, similar to the notion of geodatabases. With the introduction of ArcGIS 9, ESRI switched its programming platform from C++ or Visual Basic to Python. Python is an Open Source, interpreted, dynamically typed, object-oriented scripting language. Like most things with ESRI, it was chosen because its ease of use. All of the tools in ArcGIS 9.x are script based, allowing the user to view the code behind each tool within the Toolbox.

Python has a wide range of GIS applications in the ESRI world allowing users to do such things as multiple ring buffers, calculate fetch on a water body, or another function on an endless list of applications.

The realms of data management and python scripting are also correlated. Python scripting allows users to do such things a batch process and define projections of multiple layers eliminating the tedious tasks associated with each. Although Python lacks the fancy graphical user interface (GUI) associated with some scripting programs its ability to run Very High Level Language, exceeds most.

Applying Python Scripting in Data Management

I thought this would be a particular good specialization topic because Python scripting can be used to easily perform some of the repetitive tasks associated with data management. Also the amounts of online resources about Python are limitless, making the possibilities for its use the same.

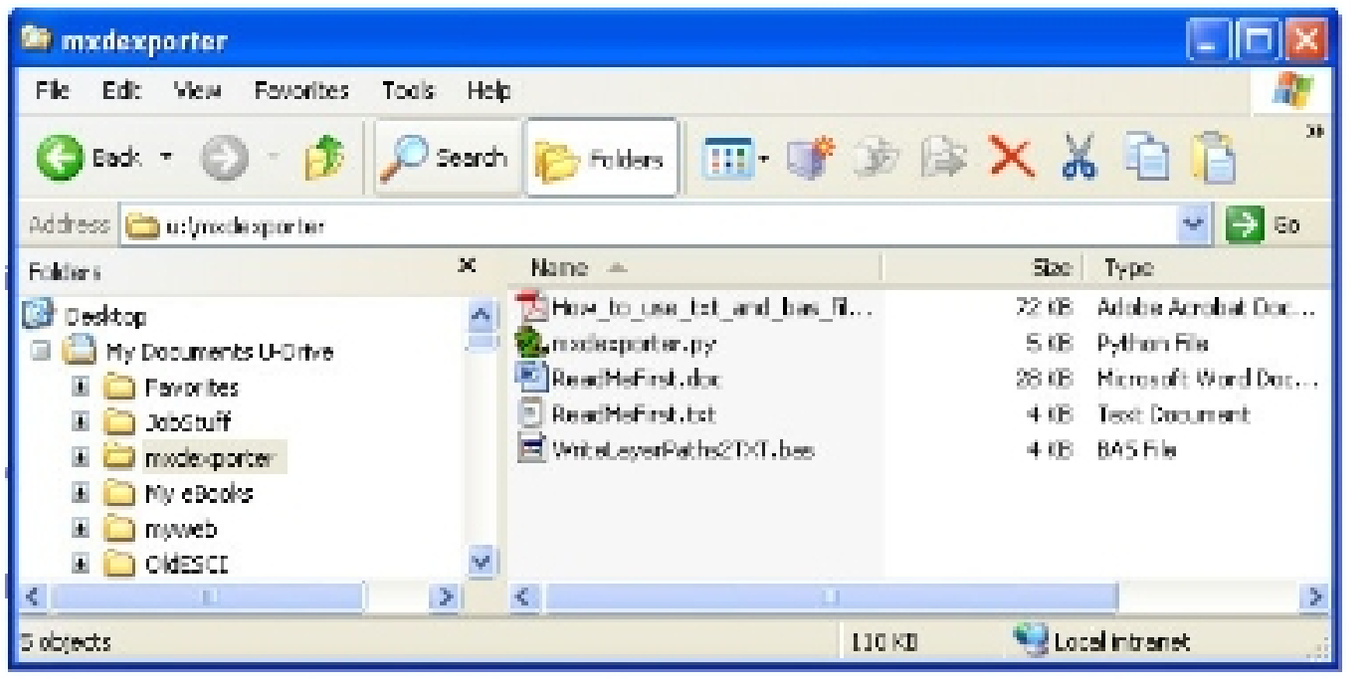
The ESRI support center provides useful directory of downloadable scripts with a variety of applications, which can by searching for Python at http://arcscripts.esri.com/.

The script I'm most interested in because of its applications to my project, automates the copying of all the shapefiles, personal geodatabase feature classes, grids, tif, and jpg files in an mxd project to a new folder. This will make the copying a project much easier, working similar to an export which captures an image of the .mxd. This will also make the data management easier.

Implementation

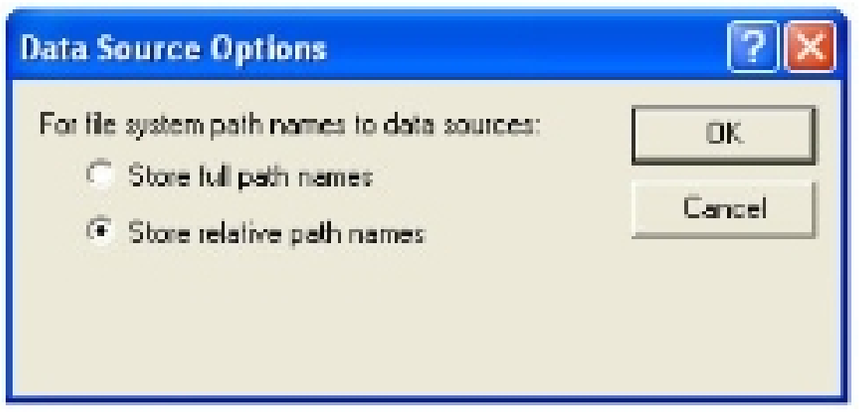
1. Download and uncompress script
    a. Download the script from ESRI's website. All scripts are in a compressed format, so the files must be uncompressed using WinZip or a similar program.

2. Read the critical document and follow the instructions
    a. With most scripts, there is a file contained within the folder that walks you through the steps associated with running the script. In this case **ReadMeFirst.doc**, displays the procedure for running the script.

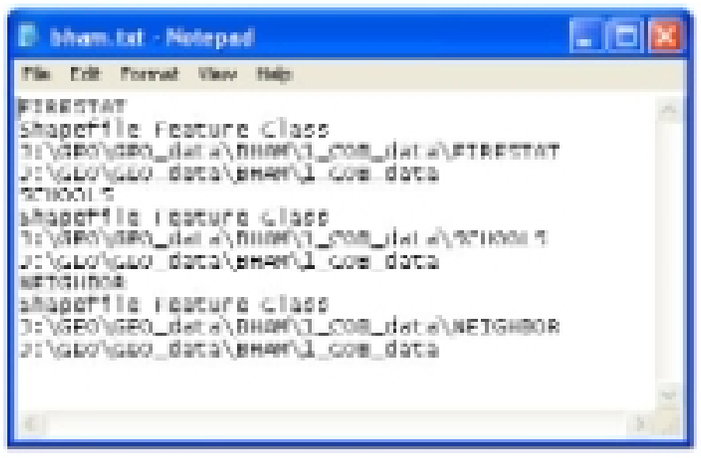


3. Create an empty folder in your working folder

4. Save .mxd
    a. Make sure the .mxd is saved using relative paths. This can be accessed File – Map Properties – Data Source Options



5. Run the tool
    a. Running the tool produces a text file containing each of the files in the .mxd including their type and location



6. Open mxdexporter.py script in WinPython and run script
    a. This will create a new folder containing all the elements listed in the .mxd.

Overall this process seems cumbersome in the beginning but frontloading a project like this will save hours later when in comes time for data management later. Remember also this is just a small glimpse into the potential of Python Scripting, the possibilities are endless and I look forward to exploring this exciting world more.