

**Advanced GIS Class  
Spring 2007**

**Batch Processing of Radar Reflectivity Files for Value Extraction**

**Newfel Mazari**

**Under the direction of: Dr Hongjie Xie.**

**University of Texas at San Antonio**

## **Abstract**

Some GIS tools are easy to use for simple and unique tasks, but for redundant work, such as extracting attribute tables from a large set of GIS format files, I need to use more sophisticated approaches, geoprocessing comes very handy to do this kind of work, especially for batch processing a series of shape files, to extract attribute tables to an ASCII output, all tables have to be merged together, while keeping the file names associated with each output table.

For my thesis research, the problem resides on how to extract reflectivity values from radar shape files, that correspond to a point location, in my case it's my rain gage location.

The output of the batch processing is very promising but I still have not fixed the error handling part of the script.

In fact the whole script crashes if the intersection of the radar file with the rain gage file is null, the intersection is processed but the export XY and field attribute generates the error, because there is no XY coordinates for the intersection and no attribute fields.

My way around this problem is to eliminate manually the bad files, and restart the script again.

I also noticed that the script does not tolerate to reprocess the same files again stored in the workspace folder, so I had to erase them also.

## **Introduction**

The use of radar products for weather forecasting, hydrological modeling or flood prediction and management is widely used, because its low cost and spatial coverage, but many problems still persist with radar products validation and comparison to other ground surface meteorological stations. Precipitation product is the most used nowadays for forecasting and water resources management, several products exist for precipitation estimates, they are usually named product levels or stages; the most basic one is radar base reflectivity (product 19, stage II). Reflectivity  $Z$  is the backscattered radar power, radar scans the atmosphere at regular angles and intervals of time, two modes are used by the NWS (National Weather Service Bureau), 1) clear sky mode 2) precipitation mode.

Radar scans the entire range (230Km radius) in 5 to 15 minutes, 6 minutes scan are used for precipitation mode, and 5 minutes for severe storm conditions. The reflectivity is converted through a power law formula to rain rate  $R$ ,  $Z = a R^b$  the parameters  $a$  and  $b$  are determined by the specific climate conditions for each region, and by weather conditions, different precipitation types may have different optimum parameters. Nevertheless, radar rainfall estimate is a volume scan that is compared to ground surface rain gauges, which creates discrepancies between the two measurements, radar and rain gauge; other problems due to radar physics, hardware, and calibration limitations, adds more to the error induced in radar rainfall estimates, plus atmospheric conditions can increase the radar bias, such as hail contamination, and the virga effect.

The present class project is part of a research project for radar reflectivity validation and comparison to a network of rain gauges installed in a single radar cell, data processing and methods used to extract the reflectivity from radar files are of primary concern to my project, I will introduce in this brief paper a GIS based method for batch processing radar reflectivity files for later analysis and comparison with rain gauge precipitation estimates

## **Data**

For my thesis research, the problem reside on how to extract reflectivity values form radar shape files, that correspond to a point location, in my case its my rain gage location.

Radar reflectivity files come in different formats:

### **-Vector format**

- 1-Shape file
- 2-III-Known Text

### **-Raster format**

- 1-Geotiff
- 2-ESRI ASCII Grid
- 3-ESRI Binary Grid
- 4-GrADS Binary Grid
- 5-HDF
- 6-NetCDF

Only shape file, and Grid formats can be used in a GIS application, at first I selected the grid format to extract the reflectivity values for my rain gage location , but the results Ire very poor, this due to three problems:

- 1-Grid uses resampling procedure to build the raster
- 2-Cels are identical in size (0.980\*0.980Km)
- 3-Each cell value is a neighborhood average

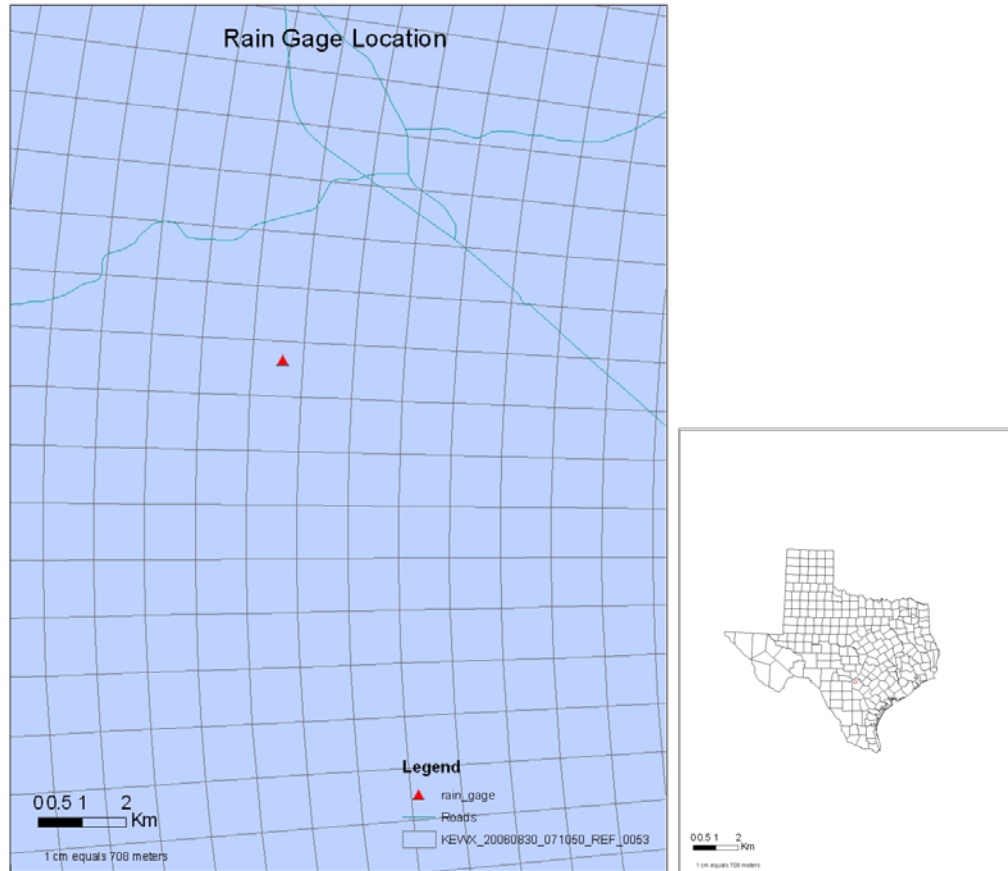
My data consists of over 1500 shape files, stored in a disk, the files Ire downloaded form the NWS radar inventory products, then exported as shape file to the computer hard drive, another shape file for the rain gage is used and stored in a different subfolder, which will be used as basis for the reflectivity value extraction. The location of the rain gage will determine which radar cell value will be extracted; figure 1 (a, b) shows the rain gage location and the radar cells.

## **Methods**

ArcGis provides several useful tools to manipulate shape files; the Analysis Tool box is the most suited for this file format, I will use a second tool form the Statistic toolbox to extract the cell values to and text file output (ASCII file).

My first concern is to locate the cells form each file that contains completely the rain gage, the solution is used the Select layer by Location Tool (figure 2). But a the problem with this tool is that each shape file has to be loaded to Arc Map as a Map layer, which constitutes a big problem when I are dealing with hundreds of large files, so I have to find something that does nit require loading each file as a layer, the second problem is to create a layer form the selection then export its attribute table to an output file, so this tool is useful for a limited files number, because it works manually.

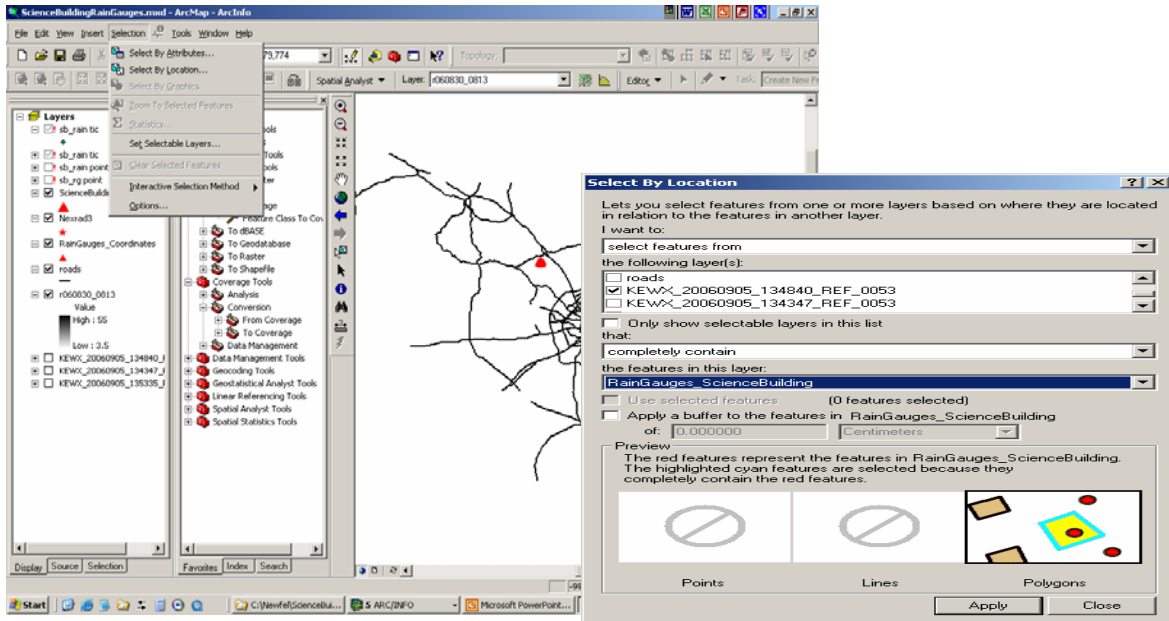
Another option is to use the Intersect tool (Overlay, Analysis Tool box), to force the intersection of a polygon with a point and join their attribute tables, the output of the intersection is set to be a point geometry to limit the usage of disk space, the intersection files will be stored in a different subfolder for later use.



**Figure1** a, b Location of rain gauges and study area.

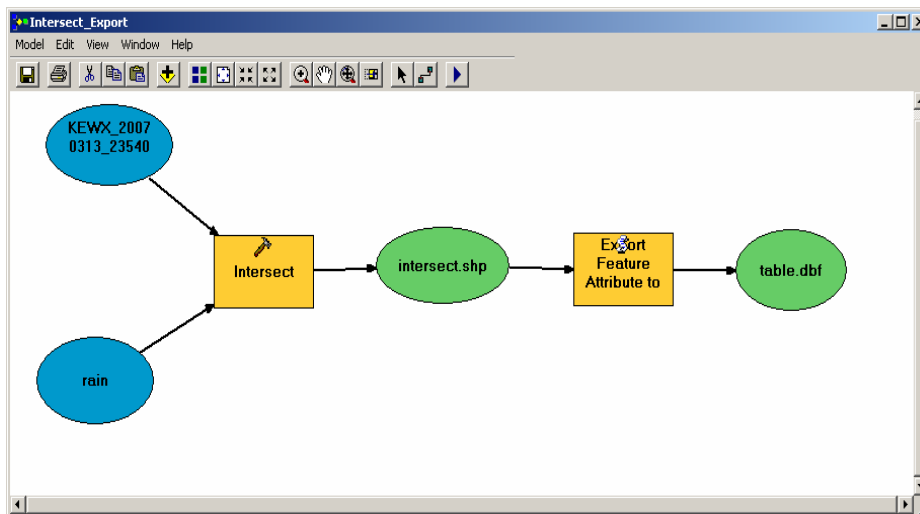
The extraction process is done by a tool in Arc Map called “Export feature Attribute to ASCII”, this tool is script based and stored in the standard toolbox, it exports the X and Y coordinates of the feature plus one choice of field attributes, for my case it will be the “value” Field.

The simplest way to start a geoprocessing session is to use the “Model Builder” in ArcGis (Figure 3), test the model functionality, set all the necessary parameters, save it as a Model then export the model to a script; ArcGis gives three options for scripting languages, visual Basic, Jython and Python, I select the last one for its ease of use, simplicity and portability, meaning it can be run in or outside the “Windows” environment.



**Figure 2:** the Select by Location Dialog box

The next step is to modify the script to process my data set without using Arc Map or Arc Catalog, a loop function is needed to iterate all process until the last file is processed.



**Figure 3:** Model Builder Window and Components in Arc GIS

Figure 4 shows the script exported from the model builder, it consists of several distinct parts:

- 1- The import of the system application tools and methods
- 2- The definition of the workspace, scratch space and output space
- 3- The parameters settings for the workspace and the tools to be used
- 4- The import statement of the necessary tools from the toolbox to do the required processing.

What is left to do, is to add a statement or function inside the script to get a list of all feature classes in the workspace folder to be processed:

```

# -----
# Intersect_Export_Python.py
# Created on: Tue May 01 2007 03:55:28 PM
# (generated by ArcGIS/ModelBuilder)
# -----

# Import system modules
import sys, string, os, win32com.client

# Create the Geoprocessor object
gp = win32com.client.Dispatch("esriGeoprocessing.GpDispatch.1")

# Load required toolboxes...
gp.AddToolbox("C:/Documents and Settings/hydrogisuser/Application Data/ESRI/ArcToolbox/My Toolboxes/Scien

#scriptarguments or Local Variables
inputfeature = "C:\\Newfel\\NewfelScience\\ScienceTest\\KEWX_20060830_075644_REF_0053.shp"
rain = "C:\\Newfel\\NewfelScience\\Science_rain_gage\\rain.shp"
intersect1 = "C:\\Newfel\\NewfelScience\\ScienceTest_Outputs\\intersect1.shp"
table1 = "C:\\Newfel\\NewfelScience\\ScienceTest_Tables\\table1.dbf"

# Process: Intersect...
gp.toolbox = "C:/Documents and Settings/hydrogisuser/Application Data/ESRI/ArcToolbox/My Toolboxes/Scien
gp.Intersect("inputfeature 'rain '", intersect1, "ALL", "", "INPUT")

# Process: Export Feature Attribute to Ascii...
gp.toolbox = "C:/Documents and Settings/hydrogisuser/Application Data/ESRI/ArcToolbox/My Toolboxes/Scien
gp.ExportXYv(intersect1, "value", "Space", table1)

# Local variables...
#intersect1_shp = "C:\\Newfel\\NewfelScience\\ScienceTest_Outputs\\intersect1.shp"
#table1 = "C:\\Newfel\\NewfelScience\\ScienceTest_Tables\\table1"
#KEWX_20060830_075644_REF_0053_2 = "KEWX_20060830_075644_REF_0053"
#rain_2 = "rain"
#KEWX_20060830_075644_REF_0053 = "KEWX_20060830_075644_REF_0053"
#rain = "rain"

```

**Figure 4:** Script as exported by the model builder

Two lines of statements will do the work of listing of the files to be processed, the first one is a regular comment to document the script, and the second line is the one doing the work

```

#get a list of feature classes in the input workspace
fcs = gp.listfeatureclasses()
#for each feature class in the list
fc = fcs.next()

```

After that I need to add a loop to repeat each process until all files are processed:

```

while fc:

    # set the outputname for each fc to be the same as the input
    output = outputWS + "\\ " + fc
    output2 = outputASC + "\\ " + fc[:18] + ".txt"

    # set the time stamp for each file.
    timeStamp = fc[:18]+ " "

    # write the timestamp to a txt file
    file1.write(timeStamp)

```

```

#for each fc in the list, intersect with the point rain-gage
# Process: Intersect...
gp.Intersect_analysis(fc + ";" + rain_gage, output, "ALL", "", "INPUT")

# Process: Export Feature Attribute to ASCII...
gp.ExportXYv_stats(output, "value", "Space", output2)

# open the asc files, read and write its contents to another ascii file
file2 = open(output2, "r")
text = file2.read()
file1.write(text)
file2.close()

fc = fcs.next()

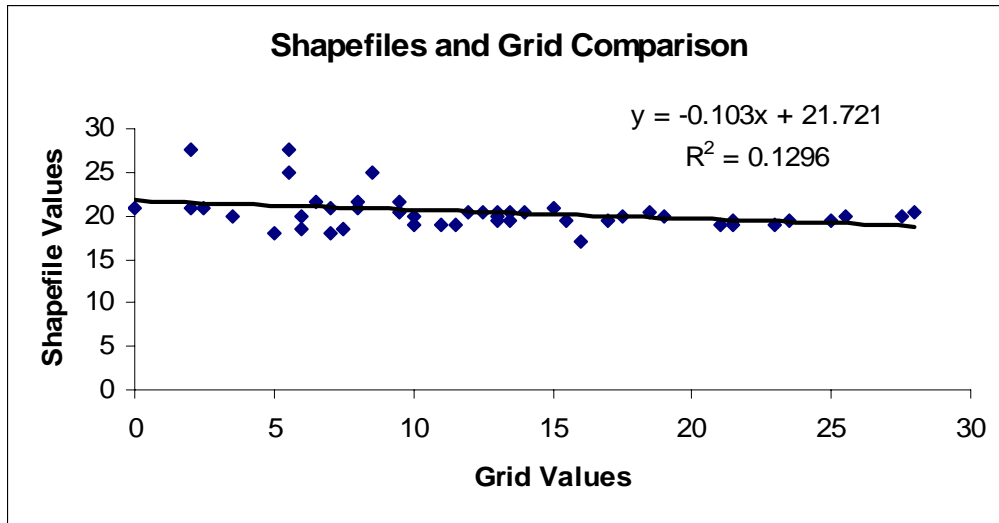
```

As shown above all the process are inside the looping block which has to be indented, for the python system requirement.

The last five lines before the “Next” close are used to merge all output text files into one file Called ”radar. text” (file1) above.

**Results**

In figure 5, I made a comparison betIen shape file values and grid values; the outcome is a very weak correlation with an R-square of 0.13 approximately, so I decided to look for other format than the grid.

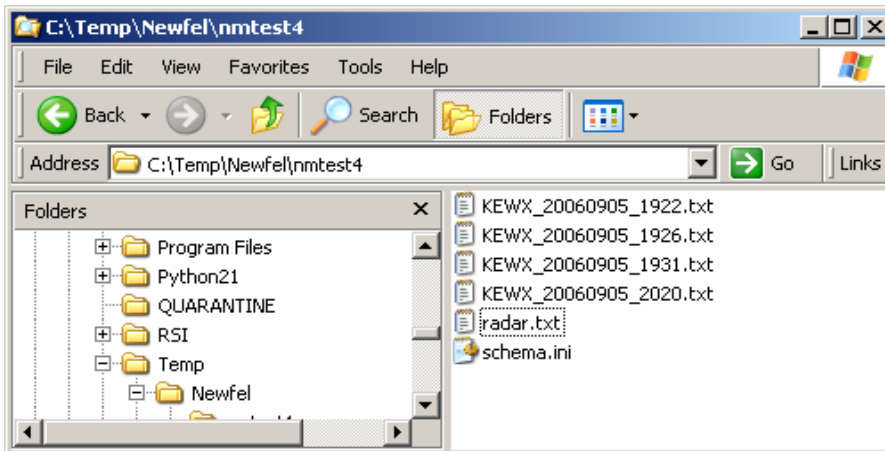


**Figure5: Grid and Shape file, reflectivity values Comparison**

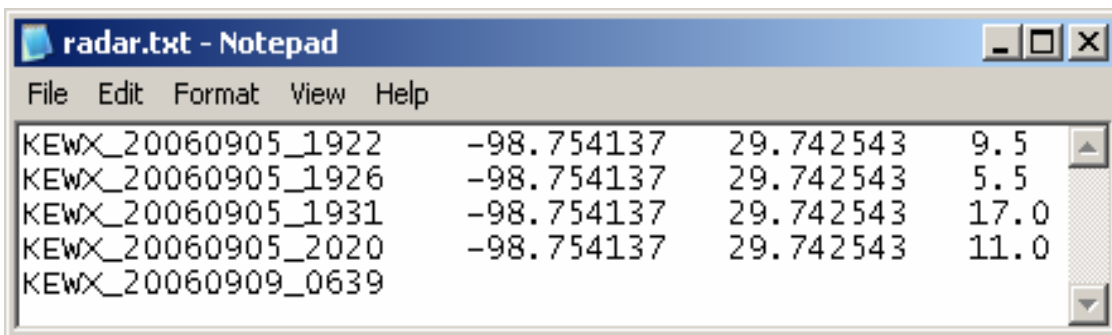
The shape file format is my second choice because it preserves the shape and size of the actual radar cell (10 X 1 Km), they only inconvenient with this format is the files size, and the functionality, in other words, it’s more complicated to extract information form a shape file than form a grid, because the grid saves all attribute information within the cell value, Ire

the shape file uses a non spatial table (dbf. format) to store cell values and other characteristics.

The output of the batch processing method is shown in figures 6, and 7; some problems still exist when there is no radar cell overlaying the rain gage point, the intersection tool works, but the export to ASCII, makes the whole script too generate an error that stops everything the script from proceeding to the next file. I'm still working on this issue, I made several adjustments but without any success yet, because of my limited knowledge of programming this may take me longer than expected to fix the bug.



**Figure 6:** Output text files for each processed shape files



**Figure 7:** Final text file with each shape file output on a single line

### Discussion

The results I had are not very good, more work should be done to handle all exceptions and error messages by the script it self, because at the present time I only have little data to process, in the future after installing a network of 8 rain gages, I will be processing a lot more data so I have to fix my data processing methods and procedures and test them efficiently in order to prepare for the next stage of my thesis.

## **Acknowledgments**

I'm really grateful to PhD. student Xianwei Wang for the advising and the help he provided me to run my batch processing task.

## **References**

- 1- Learning Python  
Lutz and Ascher  
2004, O'Reilly publications
- 2- ArcGIS Desktop Help
- 3- Python tutorial release 2.4.2  
Guido van Rossum and Fred L Drake  
28 September 2005, Python software Foundation