Reading Vector Data with OGR

Open Source RS/GIS Python
Week 1

Open Source RS/GIS modules

- OGR Simple Features Library
  - Vector data access
  - Part of GDAL
- GDAL – Geospatial Data Abstraction Library
  - Raster data access
  - Used by commercial software like ArcGIS
  - Really C++ library, but Python bindings exist

Related modules

- Numeric
  - Sophisticated array manipulation (extremely useful for raster data!)
  - This is the one we'll be using
- Numpy
  - Next generation of Numeric

Other modules

- http://www.gispython.org/ hosts Python Cartographic Library – looks like great stuff, but I haven't used it

Development environments

- FWTools
  - Includes Python, Numeric, GDAL and OGR modules, along with other fun tools
  - Just a suite of tools, not an IDE
  - I like to use Crimson Editor, but this means no debugging tools
- PythonWin
  - Have to install Numeric, GDAL and OGR individually

Documentation

- Python: http://www.python.org/doc/
- GDAL: http://www.gdal.org/, gdal.py, gdalconst.py (in the fwtools/pymod folder)
- OGR: http://www.gdal.org/ogr/, ogr.py
- Numeric: numdoc.pdf, http://numpy.sourceforge.net/numdoc/HTML/numdoc.htm

Module free-standing methods

- Some methods in modules do not rely on a pre-existing object – just on the module itself
  - `gp = arcgisscripting.create()`
  - `driver = ogr.GetDriverByName('ESRI Shapefile')`
- Some methods rely on pre-existing objects
  - `dsc = gp.Describe('landcover')`
  - `ds = driver.Open('c:/test.shp')`

OGR

- Supports many different vector formats
  - shapefiles, personal geodatabases, ArcSDE
  - MapInfo, GRASS, Microstation
  - TIGER/Line, SDTS, GML, KML
  - MySQL, PostgreSQL, Oracle Spatial, Informix, ODBC

Format Name	Creation	Georeferencing
Arc/Info Binary Coverage	No	Yes
Atlas BNA	Yes	No
Comma Separated Value (.csv)	Yes	No
DODS/OPeNDAP	No	Yes
ESRI Personal GeoDatabase	No	Yes
ESRI ArcSDE	No	Yes
ESRI Shapefile	Yes	Yes
FMEObjects Gateway	No	Yes
GeoJSON	No	Yes
Géoconcept Export	Yes	Yes
GML	Yes	Yes
GMT	Yes	Yes
GPX	Yes	Yes
GRASS	No	Yes
INTERLIS	Yes	Yes
KML	Yes	No
Mapinfo File	Yes	Yes
Microstation DGN	Yes	No
Memory	Yes	Yes
MySQL	No	No
OGDI Vectors	No	Yes
ODBC	No	Yes
Oracle Spatial	Yes	Yes
PostgreSQL	Yes	Yes
S-57 (ENC)	No	Yes
SDTS	No	Yes
SQLite	Yes	No
UK .NTF	No	Yes
U.S. Census TIGER/Line	No	Yes
VRT - Virtual Datasource	No	Yes
Informix DataBlade	Yes	Yes

OGR data drivers

- A driver is an object that knows how to interact with a certain data type (such as a shapefile)
- Need the appropriate driver in order to read or write data (ok, technically just to write because OGR handles it on read)
- Can use get_ogr_drivers.py (provided with this week's data) to get list of driver names

- Might as well get in the habit of grabbing the driver for read operations so it is available for writing as well

1. Import the OGR module
2. Use GetDriverByName(<driver name>)

```
import ogr
driver = ogr.GetDriverByName('ESRI Shapefile')
```

Opening a DataSource

- The Driver Open() method returns a DataSource object
- Open(<filename>, <update>)
  - <update> is 0 for read-only, 1 for writeable

```
fn = 'd:/data/classes/python/data/sites.shp'
dataSource = driver.Open(fn, 0)
if dataSource is None:
  print 'Could not open ' + fn
  sys.exit(1)
```

Opening a Layer (shapefile)

- The DataSource method GetLayer (<index>) returns a Layer object
- <index> is always 0 and optional for shapefiles

```
layer = dataSource.GetLayer()
```

Getting info about the layer

- Get the number of features in the layer

```
numFeatures = layer.GetFeatureCount()
print 'Feature count: ' + str(numFeatures)
print 'Feature count:', numFeatures
```

- Get the extent as a tuple (sort of a non-modifiable list)

```
extent = layer.GetExtent()
print 'Extent:', extent
print 'UL:', extent[0], extent[3]
print 'LR:', extent[1], extent[2]
```

Getting features

- If we know the FID (offset in a shapefile) of a feature, we can use the GetFeature(<index>) method on the Layer

```
feature = layer.GetFeature(0)
```

- Or we can loop through all of the features

```
feature = layer.GetNextFeature()
while feature:
  # do something here
  feature = layer.GetNextFeature()
layer.ResetReading() #need if looping again
```

Getting a feature's attributes

- Feature objects have a method called GetField() which returns the value of that attribute field
- There are variations, such as GetFieldAsString() and GetFieldAsInteger()

```
id = feature.GetField('id')
id = feature.GetFieldAsString('id')
```

Getting a feature's geometry

- Feature objects have a method called GetGeometryRef() which returns a Geometry object (could be Point, Polygon, etc)
- Point objects have GetX() and GetY() methods

```
geometry = feature.GetGeometryRef()
x = geometry.GetX()
y = geometry.GetY()
```

Destroying objects

- For memory management purposes we need to make sure that we get rid of things such as features when done with them

```
feature.Destroy()
```

- Also need to close DataSource objects when done with them

```
dataSource.Destroy()
```

The working directory

- Usually need to specify entire path for filenames
- Instead, set working directory with os.chdir()
- Similar to gp.workspace

```
import ogr, sys, os
os.chdir('d:/data/classes/python/data')
driver = ogr.GetDriverByName('ESRI Shapefile')
dataSource = driver.Open('sites.shp', 0)
```

```
# script to count features

# import modules
import ogr, os, sys

# set the working directory
os.chdir('d:/data/classes/python/data')

# get the driver
driver = ogr.GetDriverByName('ESRI Shapefile')

# open the data source
datasource = driver.Open('sites.shp', 0)
if datasource is None:
   print 'Could not open file'
   sys.exit(1)

# get the data layer
layer = datasource.GetLayer()

# loop through the features and count them
cnt = 0
feature = layer.GetNextFeature()
while feature:
   cnt = cnt + 1
   feature.Destroy()
   feature = layer.GetNextFeature()
print 'There are ' + str(cnt) + ' features'

# close the data source
datasource.Destroy()
```

Text file I/O

- To open a text file
  - Set working directory or include full path
  - Mode is 'r' for reading, 'w' for writing, 'a' for appending

```
file = open(<filename>, <mode>)
file = open('c:/data/myfile.txt', 'w')
file = open(r'c:\data\myfile.txt', 'w')
```

- To close a file once when done with it:

```
file.close()
```

- To read a file one line at a time:

```
for line in file:
  print line
```

- To write a line to a file, where the string ends with a newline character:

```
file.write('This is my line.\n')
```

Homework

- Read coordinates and attributes from a shapefile
  - Loop through the points in sites.shp
  - Write out id, x & y coordinates, and cover type for each point to a text file
  - Hint: The two attribute fields in the shapefile are called "id" and "cover"
  - Turn in your code and the output text file