Reading and Writing Vector Data with OGR

Open Source RS/GIS Python
Week 1
Why use open source?

• Pros
  • Affordable for individuals or small companies
  • Very helpful developers and fast bug fixes
  • Can use something other than Windows
  • You can impress people!

• Cons
  • Doesn’t have the built in geoprocessor
  • Smaller user community
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 in class
• NumPy
  • Next generation of Numeric
  • Some of you might use this one if you work at home
Other modules

• [http://www.gispython.org/](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/](http://www.python.org/doc/)
- **Numeric**: [http://numpy.scipy.org/#older_array](http://numpy.scipy.org/#older_array)
OGR

- Supports many different vector formats
  - ESRI formats such as shapefiles, personal geodatabases and ArcSDE
  - Other software such as MapInfo, GRASS, Microstation
  - Open formats such as TIGER/Line, SDTS, GML, KML
  - Databases such as MySQL, PostgreSQL, Oracle Spatial, Informix, ODBC
<table>
<thead>
<tr>
<th>Format Name</th>
<th>Code</th>
<th>Creation</th>
<th>Georeferencing</th>
<th>Compiled by default</th>
</tr>
</thead>
<tbody>
<tr>
<td>Arc/Info Binary Coverage</td>
<td>AVCBin</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Arc/Info .E00 (ASCII) Coverage</td>
<td>AVCE00</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Atlas BNA</td>
<td>BNA</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Comma Separated Value (.csv)</td>
<td>CSV</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>DODS/OPeNDAP</td>
<td>DODS</td>
<td>No</td>
<td>Yes</td>
<td>No, needs libdap</td>
</tr>
<tr>
<td>ESRI Personal GeoDatabase</td>
<td>PGeo</td>
<td>No</td>
<td>Yes</td>
<td>No, needs ODBC library</td>
</tr>
<tr>
<td>ESRI ArcSDE</td>
<td>SDE</td>
<td>No</td>
<td>Yes</td>
<td>No, needs ESRI SDE</td>
</tr>
<tr>
<td>ESRI Shapefile</td>
<td>ESRI Shapefile</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>FMEObjects Gateway</td>
<td>FMEObjects Gateway</td>
<td>No</td>
<td>Yes</td>
<td>No, needs FME</td>
</tr>
<tr>
<td>GeoJSON</td>
<td>GeoJSON</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Géoconcept Export</td>
<td>Géoconcept</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>GeoRSS</td>
<td>GeoRSS</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes (read support needs libexpat)</td>
</tr>
<tr>
<td>GML</td>
<td>GML</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes (read support needs Xerces)</td>
</tr>
<tr>
<td>GMT</td>
<td>GMT</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>GPX</td>
<td>GPX</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes (read support needs libexpat)</td>
</tr>
<tr>
<td>GRASS</td>
<td>GRASS</td>
<td>No</td>
<td>Yes</td>
<td>No, needs libgrass</td>
</tr>
<tr>
<td>Informix DataBlade</td>
<td>IDB</td>
<td>Yes</td>
<td>Yes</td>
<td>No, needs Informix DataBlade</td>
</tr>
<tr>
<td>INTERLIS</td>
<td>Interlis 1 and &quot;Interlis 2&quot;</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes (INTERLIS model reading needs ili2c.jar)</td>
</tr>
<tr>
<td>INGRES</td>
<td>INGRES</td>
<td>Yes</td>
<td>No</td>
<td>No, needs INGRESS</td>
</tr>
<tr>
<td>KML</td>
<td>KML</td>
<td>Yes</td>
<td>No</td>
<td>Yes (read support needs libexpat)</td>
</tr>
<tr>
<td>Mapinfo File</td>
<td>MapInfo File</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Microstation DGN</td>
<td>DGN</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Memory</td>
<td>Memory</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>MySQL</td>
<td>MySQL</td>
<td>No</td>
<td>No</td>
<td>No, needs MySQL library</td>
</tr>
<tr>
<td>Oracle Spatial</td>
<td>OCI</td>
<td>Yes</td>
<td>Yes</td>
<td>No, needs OCI library</td>
</tr>
<tr>
<td>ODBC</td>
<td>ODBC</td>
<td>No</td>
<td>Yes</td>
<td>No, needs ODBC library</td>
</tr>
<tr>
<td>OGDI Vectors</td>
<td>OGDI</td>
<td>No</td>
<td>Yes</td>
<td>No, needs OGDI library</td>
</tr>
<tr>
<td>PostgreSQL</td>
<td>PostgreSQL</td>
<td>Yes</td>
<td>Yes</td>
<td>No, needs PostgreSQL library</td>
</tr>
<tr>
<td>S-57 (ENC)</td>
<td>SS7</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>SDTS</td>
<td>SDTS</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>SQLite</td>
<td>SQLite</td>
<td>Yes</td>
<td>No</td>
<td>No, needs libsqlite3</td>
</tr>
<tr>
<td>UK .NTF</td>
<td>UK. NTF</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>U.S. Census TIGER/Line</td>
<td>TIGER</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>VRT - Virtual Datasource</td>
<td>VRT</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>X-Plane/Flighgear aeronautical da</td>
<td>XPLANE</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>

From http://www.gdal.org/ogr/ogr_formats.html
Available formats

• The version we use in class doesn’t support *everything* on the previous slide

• To see available formats use this command from the FWTools shell:
  
  `ogrinfo --formats`

• Same syntax if using a shell other than FWTools and the gdal & ogr utilities are in your path – otherwise provide the full path to ogrinfo
Detour: Module methods

• Some methods in modules do not rely on a pre-existing object – just on the module itself
  • \texttt{gp = arcgisscripting.create()}
  • \texttt{driver = ogr.GetDriverByName('ESRI Shapefile')}

• Some methods rely on pre-existing objects
  • \texttt{dsc = gp.Describe('landcover')}
  • \texttt{ds = driver.Open('c:/test.shp')}
Importing OGR

- With FWTools:
  ```python
  import ogr
  ```

- With an OSGeo distribution:
  ```python
  from osgeo import ogr
  ```

- Handle both cases like this:
  ```python
  try:
      from osgeo import import ogr
  except:
      import ogr
  ```
OGR data drivers

• A driver is an object that knows how to interact with a certain data type (such as a shapefile)

• Need an appropriate driver in order to read or write data (need it explicitly for write)

• Use the Code from slide 9 to get the desired driver
• Might as well grab the driver for read operations so it is available for writing
  1. Import the OGR module
  2. Use `ogr.GetDriverByName(<driver_code>)`

```python
import ogr
driver = ogr.GetDriverByName('ESRI Shapefile')
```
Opening a DataSource

- The Driver `Open()` method returns a DataSource object

```python
fn = 'f:/data/classes/python/data/sites.shp'
dataSource = driver.Open(fn, 0)
if dataSource is None:
    print 'Could not open ' + fn
    sys.exit(1) #exit with an error code
```
Detour: Working directory

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

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

• Use `GetLayer(<index>)` on a `DataSource` to get a `Layer` object
• `<index>` is always 0 and optional for shapefiles
• `<index>` is useful for other data types such as GML, TIGER

```python
layer = dataSource.GetLayer()
layer = dataSource.GetLayer(0)
```
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) of a feature, we can use `GetFeature(<index>)` on the Layer

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

• Or we can loop through all of the features

```python
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 `GetField(<name>)` method which returns the value of that attribute field

• There are variations, such as
  `GetFieldAsHexString(<name>)` and
  `GetFieldAsInteger(<name>)`

```python
id = feature.GetField('id')
id = feature.GetFieldAsHexString('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

```python
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()
# script to count features

# import modules
import ogr, os, sys

# set the working directory
os.chdir('f:/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()
Review: 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

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

• To close a file when done with it:
  ```python
  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')
Assignment 1a

• 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, one point per line
  • Hint: The two attribute fields in the shapefile are called "id" and "cover"
• Turn in your code and the output text file
Writing data

1. Get or create a writeable layer
2. Add fields if necessary
3. Create a feature
4. Populate the feature
5. Add the feature to the layer
6. Close the layer
Getting a writeable layer

• Open an existing DataSource for writing and get the layer out of it

```python
fn = 'f:/data/classes/python/data/sites.shp'
dataSource = driver.Open(fn, 1)
if dataSource is None:
    print 'Could not open ' + fn
    sys.exit(1)  #exit with an error code
layer = dataSource.GetLayer(0)
```
Creating a writeable layer

• Create a new DataSource and Layer

  1. CreateDataSource(<filename>) on a Driver object – the file cannot already exist!

  2. CreateLayer(<name>,
     geom_type=<OGRwkbGeometryType>, [srs])
     on a DataSource object

```python
 ds = driver.CreateDataSource('test.shp')
 layer = ds.CreateLayer('test',
          geom_type=ogr.wkbPoint)
```
Checking if a datasource exists

• Use the exists(<filename>) method in the os.path module
• Use DeleteDataSource(<filename>) on a Driver object to delete it (this causes an error if the file does not exist)

```python
import os
if os.path.exists('test.shp'):
    driver.DeleteDataSource('test.shp')
```
Adding fields

- Cannot add fields to non-empty shapefiles
- Shapefiles need at least one attribute field
- Need a FieldDefn object first
  - Copy one from an existing feature with
    
    ```python
    fieldDefn = feature.GetFieldDefnRef(0)
    fieldDefn = feature.GetFieldDefnRef('id')
    ```
• Or create a new FieldDefn with
  
  ```python
  FieldDefn(<field_name>, <OGRFieldType>),
  ```

  where the field name has a 12-character limit

  ```python
  fldDef = ogr.FieldDefn('id', ogr.OFTInteger)
  ```

• If it is a string field, set the width

  ```python
  fieldDefn = ogr.FieldDefn('id', ogr.OFTString)
  fieldDefn.SetWidth(4)
  ```
• Now create a field on the layer using the FieldDefn object and
  `CreateField(<FieldDefn>)`

  `layer.CreateField(fieldDefn)`
Creating new features

• Need a FeatureDefn object first
  • Get it from the layer after adding any fields
  ```python
  featureDefn = layer.GetLayerDefn()
  ```

• Now use the FeatureDefn object to create a new Feature object
  ```python
  feature = ogr.Feature(featureDefn)
  ```
• Set the geometry for the new feature
  `feature.SetGeometry(point)`

• Set the attributes with `SetField(<name>, <value>)`
  `feature.SetField('id', 23)`

• Write the feature to the layer
  `layer.CreateFeature(feature)`

• Make sure to close the DataSource with `Destroy()` at the end so things get written
# script to copy first 10 points in a shapefile
# import modules, set the working directory, and get the driver
import ogr, os, sys
os.chdir('f:/data/classes/python/data')
driver = ogr.GetDriverByName('ESRI Shapefile')

# open the input data source and get the layer
inDS = driver.Open('sites.shp', 0)
if inDS is None:
    print 'Could not open file'
sys.exit(1)
inLayer = inDS.GetLayer()

# create a new data source and layer
if os.path.exists('test.shp'):
    driver.DeleteDataSource('test.shp')
outDS = driver.CreateDataSource('test.shp')
if outDS is None:
    print 'Could not create file'
sys.exit(1)
outLayer = outDS.CreateLayer('test', geom_type=ogr.wkbPoint)

# use the input FieldDefn to add a field to the output
fieldDefn = inLayer.GetFeature(0).GetFieldDefnRef('id')
outLayer.CreateField(fieldDefn)
# get the FeatureDefn for the output layer
featureDefn = outLayer.GetLayerDefn()

# loop through the input features

cnt = 0
inFeature = inLayer.GetNextFeature()
while inFeature:

# create a new feature
outFeature = ogr.Feature(featureDefn)
outFeature.SetGeometry(inFeature.GetGeometryRef())
outFeature.SetField('id', inFeature.GetField('id'))

# add the feature to the output layer
outLayer.CreateFeature(outFeature)

# destroy the features
inFeature.Destroy()
outFeature.Destroy()

# increment cnt and if we have to do more then keep looping

cnt = cnt + 1
if cnt < 10: inFeature = inLayer.GetNextFeature()
else: break

# close the data sources
inDS.Destroy()
outDS.Destroy()
Assignment 1b

- Copy selected features from one shapefile to another
  - Create a new point shapefile and add an ID field
  - Loop through the points in sites.shp
    - If the cover attribute for a point is ‘trees’ then write that point out to the new shapefile
- Turn in your code and a screenshot of the new shapefile being displayed