Map Algebra and Writing Raster Data

Open Source RS/GIS Python
Week 5

Numeric & NumPy

- The FWTools version of the GDAL libraries uses the Numeric Python module
- Most other new versions of GDAL use NumPy instead
- Both are similar to IDL or Matlab in that you can easily process large multi-dimensional arrays of data
- Manuals are included with this week's data

Using Numeric & NumPy

```
>>> import Numeric #note the capital N
>>> import numpy #note the lower-case n

>>> x = Numeric.arange(20)
>>> x = numpy.arange(20)
>>> print x
[ 0  1  2  3  4  5  6  7  8  9 10 11 12 13
  14 15 16 17 18 19]

>>> print x[10]
10

>>> print x[:10] # start up to 10
[0 1 2 3 4 5 6 7 8 9]

>>> print x[10:] # 10 thru end
[10 11 12 13 14 15 16 17 18 19]
```

```
>>> print x
[ 0  1  2  3  4  5  6  7  8  9 10 11 12 13
  14 15 16 17 18 19]

>>> print x[5:15] # 5 up to 15
[ 5  6  7  8  9 10 11 12 13 14]

>>> print x[5:15:2] # 5 up to 15 by 2
[ 5  7  9 11 13]

>>> print x[::3] # start thru end by 3
[ 0  3  6  9 12 15 18]

>>> print x[::-2] # end thru start by -2
[19 17 15 13 11  9  7  5  3  1]

>>> print x[10::-2] # 10 thru start by -2
[10  8  6  4  2  0]
```

```
>>> print Numeric.zeros(5) # initialize to 0
[0 0 0 0 0]

>>> print Numeric.reshape(x, (2,10))
[[ 0  1  2  3  4  5  6  7  8  9]
 [10 11 12 13 14 15 16 17 18 19]]

>>> print Numeric.reshape(x, (10,2))
[[ 0  1]
 [ 2  3]
 [ 4  5]
 [ 6  7]
 [ 8  9]
 [10 11]
 [12 13]
 [14 15]
 [16 17]
 [18 19]]
```

reshape(<array>, (<rows>, <cols>))

For numpy, just replace **`Numeric`** with **`numpy`**

```
>>> a = Numeric.array([10, 2, 5])
>>> b = Numeric.array([4, 3, 7])

>>> print a+b
[14  5 12]

>>> print a-b
[ 6 -1 -2]

>>> print a*b
[40  6 35]

>>> print a/b
[2 0 0]

>>> print (a/b).astype(Numeric.Float16)
[ 2.  0.  0.]

>>> print a.astype(Numeric.Float16)/b
[ 2.5         0.66666667  0.71428571]
```

numpy: substitute **`numpy.float`** for **`Numeric.Float16`**

```
>>> a = Numeric.array([10, 2, 5])
>>> b = Numeric.array([4, 3, 7])

>>> print a/2
[5 1 2]

>>> print a/2.0
[ 5.   1.   2.5]

>>> print a % b # mod
[2 2 5]

>>> print a == b
[0 0 0]

>>> print a > b
[1 0 0]

>>> print a < b
[0 1 1]
```

```
>>> values = Numeric.array([100,500])
>>> print values
[100 500]

>>> input = Numeric.array([0,3,5,0,2])
>>> print input
[0 3 5 0 2]

>>> mask = Numeric.greater(input, 0)
>>> print mask
[0 1 1 0 1]

#chooses the <mask> value from <values>
>>> output = Numeric.choose(mask, values)
>>> print output
[100 500 500 100 500]
```

```
>>> print y
[2 6 4 9 1 5 3]

# if y>5 then 10 else 0
>>> print Numeric.where(y >= 5, 10, 0)
[ 0 10  0 10  0 10  0]

# if y>5 then 10 else y
>>> print Numeric.where(y >= 5, 10, y)
[ 2 10  4 10  1 10  3]

# clip y so min value is 3 and max is 6
>>> print Numeric.clip(y, 3, 6)
[3 6 4 6 3 5 3]
```

```
>>> z = Numeric.array([[1,2,3],[4,5,6],[7,8,9]])
>>> print z
[[1 2 3]
 [4 5 6]
 [7 8 9]]

>>> print z[0,0]
1

>>> print z[2,1]
8

>>> print z[2:]
[ [7 8 9]]

>>> print z[2]
[7 8 9]

>>> print z[2,:]
[7 8 9]
```

```
>>> print z
[[1 2 3]
 [4 5 6]
 [7 8 9]]

>>> print z[:,2]
[3 6 9]

>>> print z[:1,0]
[1]

>>> print z[:2,0]
[1 4]
```

Manipulating data

- Say we want to compute an NDVI (normalized difference vegetation index) on aster.img
- (NIR-RED)/(NIR+RED), where NIR is band 3 and RED is band 2
- Assume we have read data from band 3 into data3 and band 2 into data2

```
ndvi = (data3 – data2) / (data3 + data2)
```

```
ndvi = (data3 – data2) / (data3 + data2)
```

- What happens if data3 and data2 are both 0 (NODATA since this image uses 0 as the NODATA value)?
- *Division by Zero* error (almost as bad as the Blue Screen of Death!)

- Have to cast data to floating point before choose() will work correctly
- We want output to be floating point anyway (-1.0 – 1.0)

```
data2 = band2.ReadAsArray(0, 0, cols,
  rows).astype(Numeric.Float16)
data3 = band3.ReadAsArray(0, 0, cols,
  rows).astype(Numeric.Float16)
mask = Numeric.greater(data3 + data2, 0)
ndvi = Numeric.choose(mask, (-99, (data3 -
  data2) / (data3 + data2)))
```

Mac problem

- I ran into a problem using numpy on a Mac (it worked on a Windows box)
- To avoid a division error, I had to change the last line on the previous slide to this:

```
ndvi = Numeric.choose(mask, (-99, (data3 -
  data2) / (data3 + data2 + 0.00000000001)))
```

Creating a new data set

- We probably want to write the NDVI out to a file
- We need a Driver object that will create the type of file we want
- Can get the Driver that the input file uses like this:

```
driver = inDataset.GetDriver()
```

- **`Create(<filename>, <xsize>, <ysize>, [<bands>], [<GDALDataType>])`**
  - bands is optional and defaults to 1
  - GDALDataType is optional and defaults to GDT_Byte

```
outDataset = driver.Create(filename, cols,
  rows, 1, GDT_Float32)
```

- Space on disk is allocated immediately

Writing to a raster data set

- First we need to get the band to write to

```
outBand = outDataset.GetRasterBand(1)
```

- Band objects have a **`WriteArray(array, xoff, yoff)`** method that we can use to write a Numeric array into the Band
- Assuming we computed NDVI for the entire image:

```
outBand.WriteArray(ndvi, 0, 0)
```

Reading & writing by block

```
xBlockSize = 64
yBlockSize = 64
for i in range(0, rows, yBlockSize):
  if i + yBlockSize < rows:
      numRows = yBlockSize
  else:
      numRows = rows - i
  for j in range(0, cols, xBlockSize):
      if j + xBlockSize < cols:
            numCols = xBlockSize
      else:
            numCols = cols - j
      data = band.ReadAsArray(j, i, numCols, numRows)
      # do calculations here to create outData array
      outBand.WriteArray(outData, j, i)
```

Setting a NoData value

- Use **`SetNoDataValue(<value>)`** on a Band object to set its NoData value

```
outBand.SetNoDataValue(-99)
```

- Can get the NoData value of an existing band with **`GetNoDataValue()`**
  - Returns **`None`** if there isn't one set (like for aster.img)

Calculating band statistics

- Flush data to disk with **`FlushCache()`**
- Use the **`GetStatistics(<approx_ok>, <force>)`** method on the band
- If approx_ok=1 then stats might be computed based on pyramids
- If force=0 then stats will not be computed if the entire image needs to be re-read

```
outBand.FlushCache()
outBand.GetStatistics(0, 1)
```

Georeferencing a new image

- We probably want our NDVI image to be georeferenced – easy if it is the same as the input image

```
geoTransform = inDataset.GetGeoTransform()
outDataset.SetGeoTransform(geoTransform )
```

- We can do the same with projection information

```
proj = inDataset.GetProjection()
outDataset.SetProjection(proj)
```

Building pyramids

- Force Imagine-style pyramid file (.rrd)

```
gdal.SetConfigOption('HFA_USE_RRD', 'YES')
```

- To actually build the pyramids

```
outDataset.BuildOverviews(overviewlist=[2,4,
  8,16,32,64,128])
```

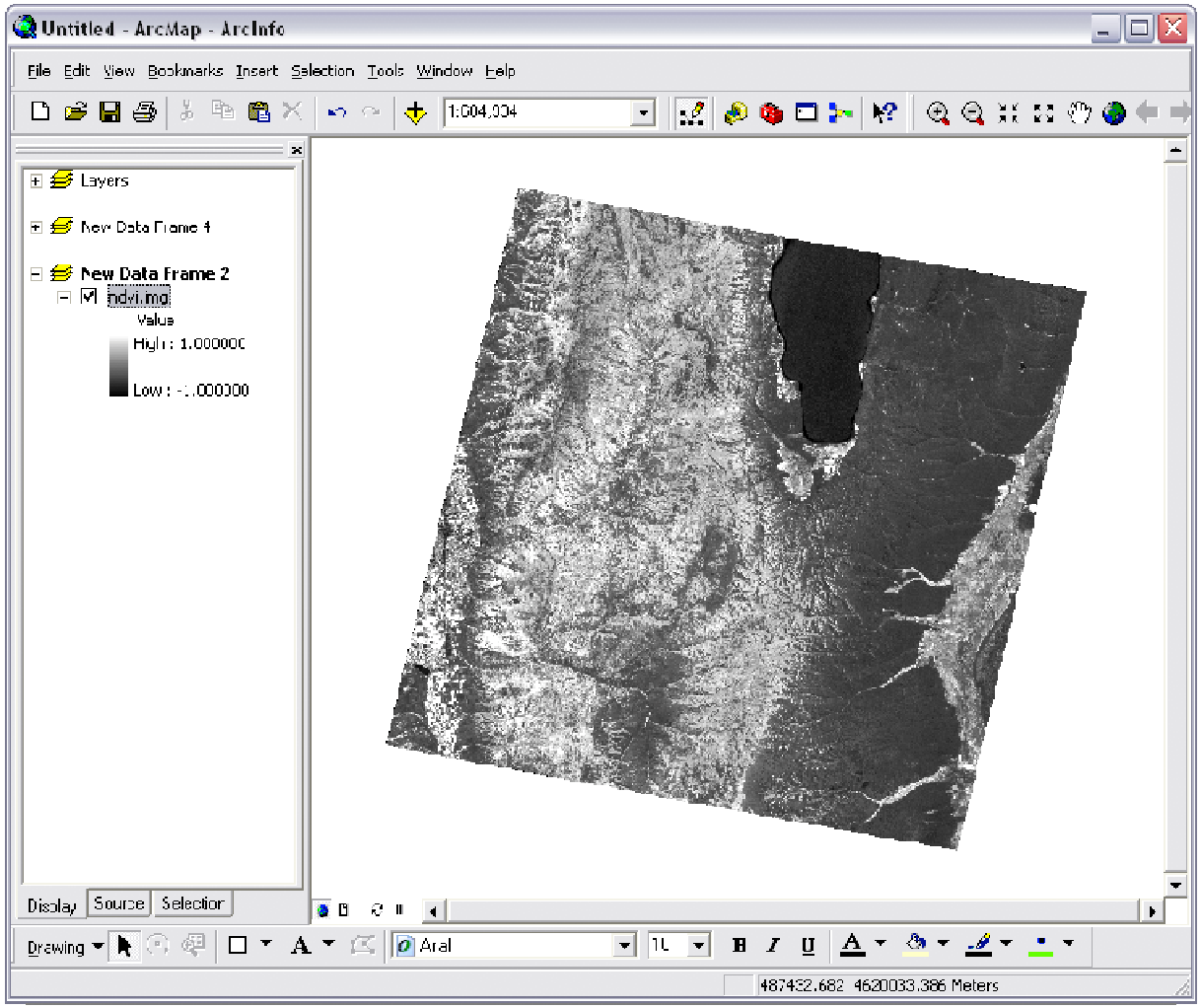
- A pyramid level of 4 on an 5665x5033 image will be a 1417x1259 tile

```
5665/4 = 1417
5033/4 = 1259
```

Assignment 5a

- Create an NDVI image
  - Read in data from aster.img
  - Create an NDVI image
  - Write out NDVI to new file
  - Can do entire image at once or block by block
  - Don't forget to calculate statistics, set projection and georeferencing information, and build pyramids

- It should look like this:



How would we mosaic images?

1. For each image:
   - Get the number of rows and columns
   - Get origin x,y (minX, maxY) from the geotransform
   - Get pixel width and pixel height from the geotransform
   - Compute maxX and minY
     - maxX1 = minX1 + (cols1 * pixelWidth)
     - minY1 = maxY1 + (rows1 * pixelHeight) [remember pixel height is negative]

2. Get minX, maxX, minY, maxY for the output image
   - minX = min(minX1, minX2, …)
   - maxX = max(maxX1, maxX2, …)
   - Do the same for minY and maxY
3. Compute the number of rows and columns for the output image
   - cols = int((maxX – minX) / pixelWidth)
   - rows = int((maxY – minY) / abs(pixelHeight)

4. Create the output image
5. For each image:
   - Compute the offset of that image's minX and maxY based on the size of the output image
     - xOffset1 = int((minX1 - minX) / pixelWidth)
     - yOffset1 = int((maxY1 - maxY) / pixelHeight)
   - Read in the data for that image (we'll do the whole thing at once to make it easy)
   - Write out the data to the output image using the computed offsets

6. For the output image:
   - Compute the statistics
   - Set the geotransform
     - [minX, pixelWidth, 0, maxY, 0, pixelHeight]
   - Set the projection
   - Build the pyramids

Assignment 5b

- Mosaic doq1.img and doq2.img together
  - The pixel sizes are the same for both images
  - Read in each image all at once – that will make it easier
  - If you display it in ArcMap, change the symbology so it doesn't stretch the data and it will look better
  - Because it has different pyramid levels than the originals it might look offset when zoomed out, but zoom in and you'll see no difference