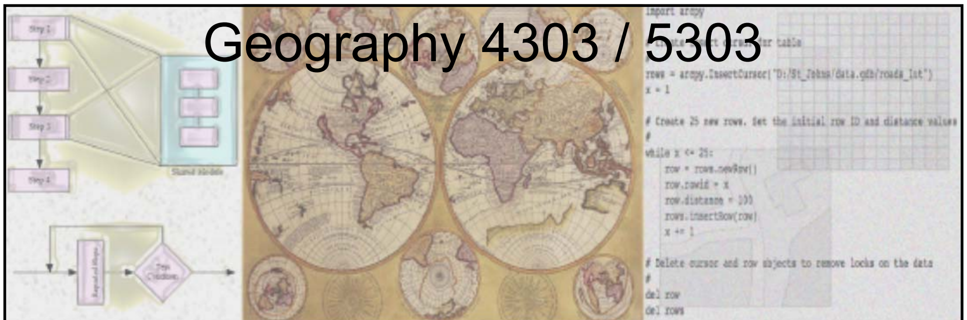


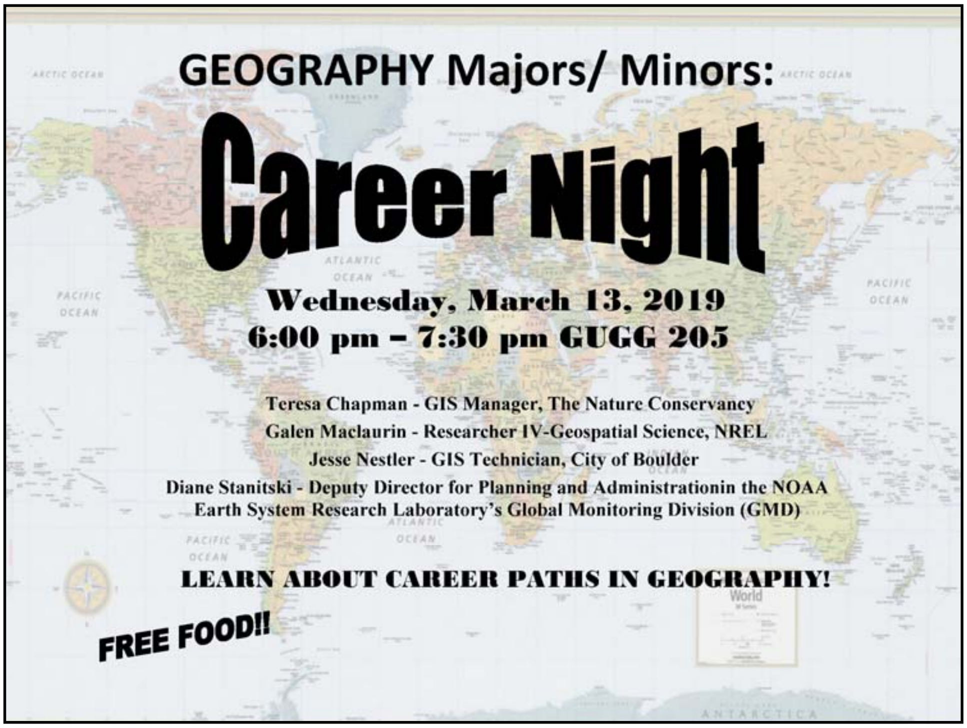
Geography 4303 / 5303

GIS Programming for Spatial Analysis

Class 09: Raster Analysis, Modeling in Numpy & GDAL

Some Updates

- Lab 5 work and submission
- Groups for final projects
- Schedule for project work to be discussed later



Last Lecture / Last Week

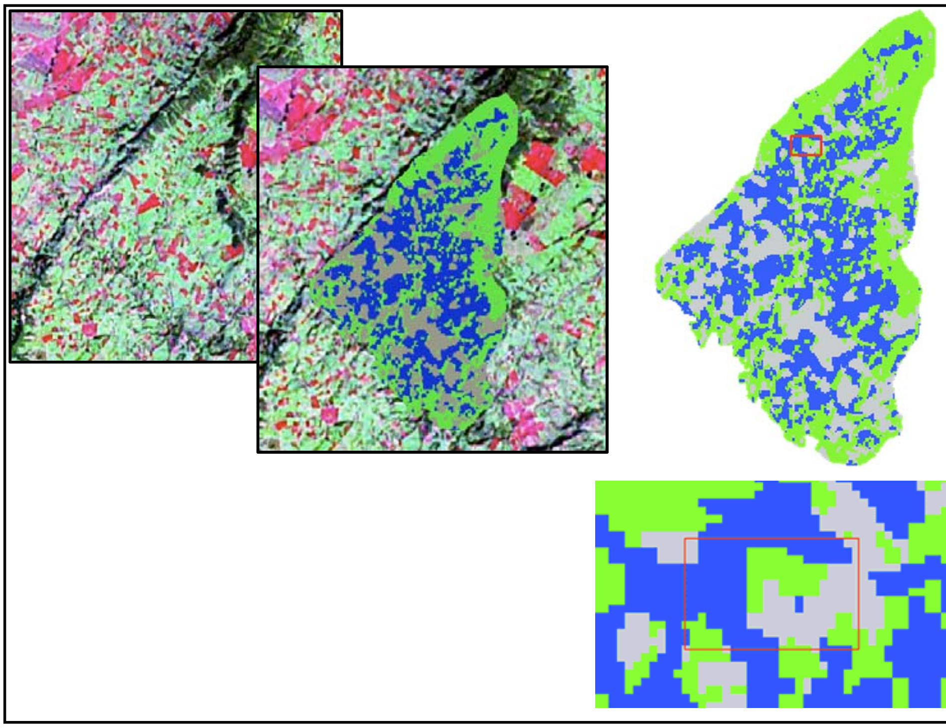
- Raster map algebra
- Local functions
- Focal functions: filtering
- Using Numpy multiarrays and Raster objects (arcpy)
- Exercise class 08?

Today ‘s Outline

- More on focal functions and directionality measures (example of exposure assessment)
- Concepts of zonal analysis in raster data
- Concepts of global functions, focus on distance calculations
- Exercise: Compute distance grids

More focal analysis: Detection

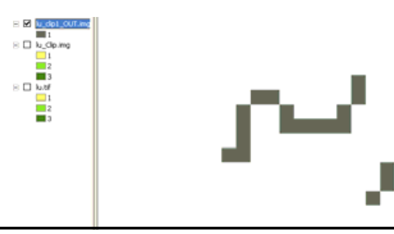
- Identify pixels that fulfill certain boundary conditions
  - What is nearby?
  - Where are classes of interest, relative to the forest edge?
- E.g., we would like to find forest pixels that have agric. area in their direct neighborhood
- These locations may be affected by and exposed to pesticide or insecticide applications



Conditions in focal neighborhoods

- Shape of neighborhood
- Moving window
- Classes in focal environment
  - 1-Agriculture
  - 2-Other
  - 3-Forest

```
def computeBoundary(myArray):
    print "Executing computeBoundary()..."
    myArrayBound = numpy.zeros(myArray.shape).astype(int)
    for i in range (1,numpy.size(myArray,1)-1):
        for j in range (1,numpy.size(myArray,0)-1):
            if myArray[j][i] == 42:
                for ii in range(i-1,i+2):
                    for jj in range(j-1,j+2):
                        if myArray[jj][ii] == 81 or myArray[jj][ii] == 82:
                            myArrayBound[j][i] = 1
    return myArrayBound
```

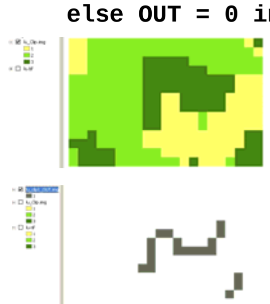


Pseudo-codes

- Good time to talk about **pseudo-codes**
- They help you to write down the **implementation idea** (rationale) in **natural language** but guided by computing logic
- Helps you to recognize the most important points, places for **conditional constructs** and needs for **iterations** (nested and simple)
- Way of **schematic thinking** packed into your **own language** easily transferred into programming slang

Pseudo-code forest boundaries

```
For each pixel p in image I look up the pixel value v
  If v = forest (42) search the focal environment of p
  For each neighboring pixel pp adjacent to p
    Test if pp has a value vv = agriculture (81 or 82)
      If this is true increment a variable COUNT
  If COUNT is greater than or equal to 2
    p is given the value OUT = 1 in the output image;
  else OUT = 0 in the output image
```

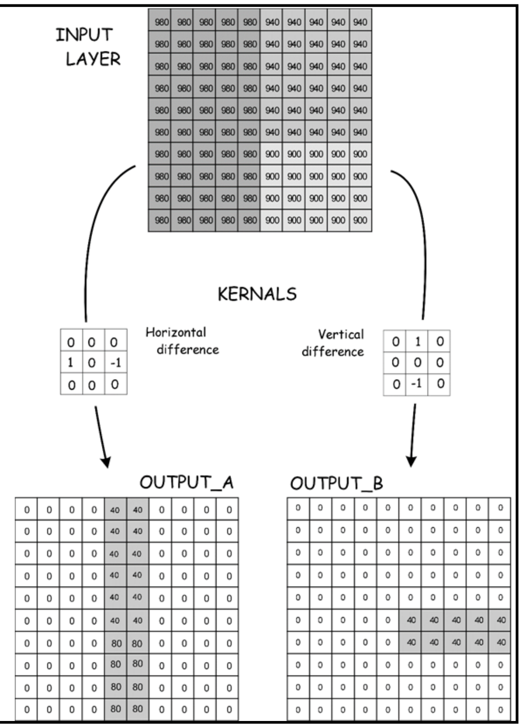


```
def forestAgricBounds(myArrayList):
    myArray = myArrayList[0]
    ySize = myArray.shape[0]
    xSize = myArray.shape[1]
    myArray2 = Numeric.zeros((ySize, xSize))
    #myArray[rows(Ydown)][columns(Xright)]
    for i in range(1,xSize - 1):
        for j in range(1,ySize - 1):
            count1 = 0
            if myArray[j][i] == 3:
                for ii in range (i-1,i+2):
                    for jj in range (j-1,j+2):
                        if myArray[jj][ii] == 1: count1 += 1

            if count1 >= 2: myArray2[j][i] = 1
            else: myArray2[j][i] = 0

    myArrayList[0] = myArray2
    return myArrayList
```

class07b_np_arc_focal – boundary

Edge Detection Using Kernels

- Concentration changes; elevation changes,…
- Discovering **contrasts** / **differences** within the focal neighborhood



More focal analysis: Patch definition

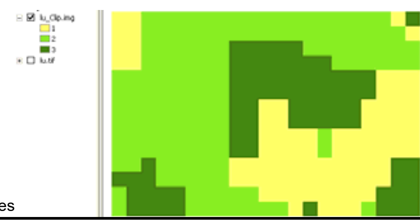
- We are interested in identifying pixels that fulfill certain boundary and contiguity conditions
- Agricultural pixels in contiguous groups (patches) defined agricultural area in their direct neighborhood
- Patch boundaries (those Ag pixels that constitute the borderline pixels to other classes) – here we have to work with two grids

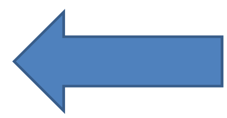
class07b_np_arc_focal – patch boundaries

More focal analysis: Directionality

- An example for forest exposure to insecticide application in grassland
- East wind direction:
  - If forest has an adjacent grassland pixel (71) East of it the forest pixel is Potentially "exposed"
  - otherwise not (East wind assumed)

- Classes
  - 1-Agriculture
  - 2-Other
  - 3-Forest





class07b_np_arc_focal – patch boundaries

Exercise Class 09a: Impacts of Pesticide Applications on Forest Ecosystems

Using land cover data you are supposed to carry out an assessment of potential **pesticide impact** on **forest ecosystems**. Wind speed and wind direction are very critical parameters in pesticide exposure assessments. Here you carry out an assessment under the constraint of **main wind direction only.**

-Assume that the main wind direction is ***North-West*** in your study area (which could change of course).

-You are estimating the exposure of forest pixels (42) from pesticide application done in agricultural pixels (81 or 82).

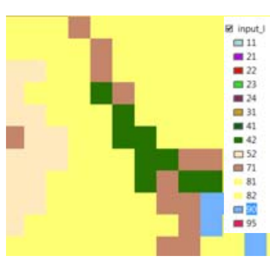
-To be exposed there should be direct adjacency (simplified) between forest and agriculture land use.

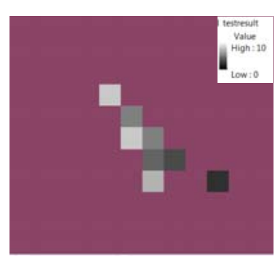
-You are using a very simplified scale to quantify potential exposure. Forest pixels located ***South-East*** of an agricultural pixel get the value of 3 (exposure), forest locations ***South and East*** of an agricultural pixel get a value of 2 and all other directions get a value of 1. Each pixel will be assigned with the cumulative sum of these values.

Write a pseudo-code for solving this problem and try to be very clear about working steps, tests you have to perform and looping techniques you will need to apply.

The new image should answer the question: To what degree is each forest location potentially impacted by **agricultural** pesticide application or in other words: each forest pixel should get a value between 0 and ???

If you are done you can think about more advanced exposure assessments that would include wind speed and thus the distance decay functions for potential exposure.



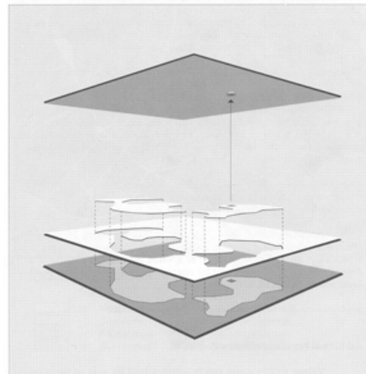


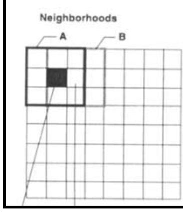
Zonal & Block Functions

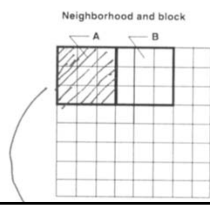
- Zones of **another** layer for evaluation of target cell
- Zones = **geographic areas**
- Internal **attribute homogeneity**
- Statistics are written **into each cell of a zone**

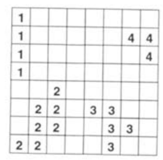
```
meangrid = zonalmean(zonalgrid,valuegrid)
```

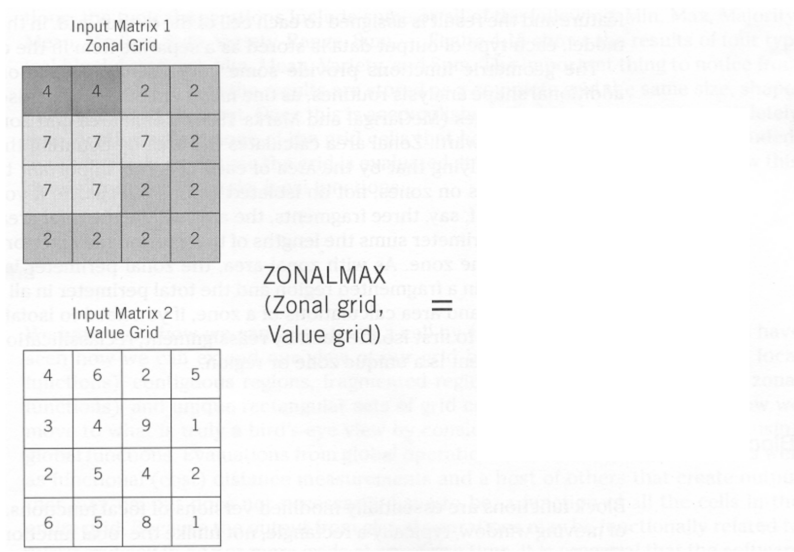
- **Blocks** similar but predefined “roving window”



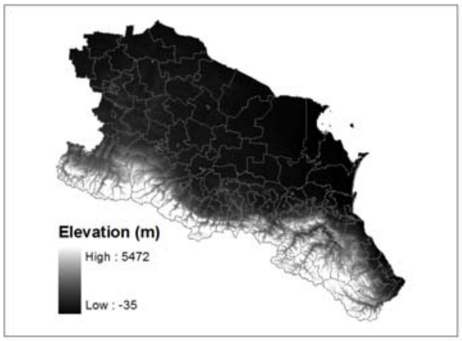


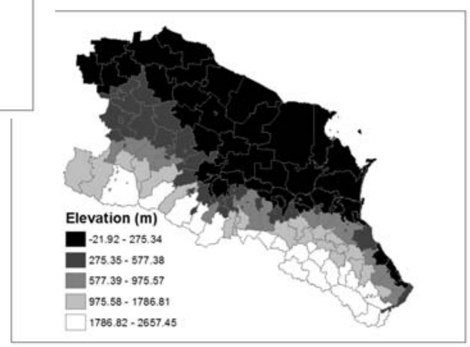






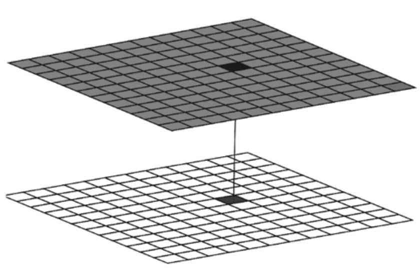
Average elevation by district?

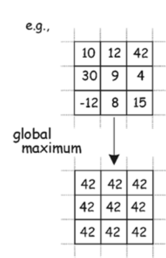




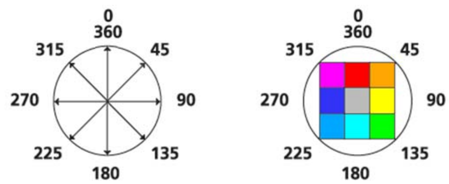
Global Functions

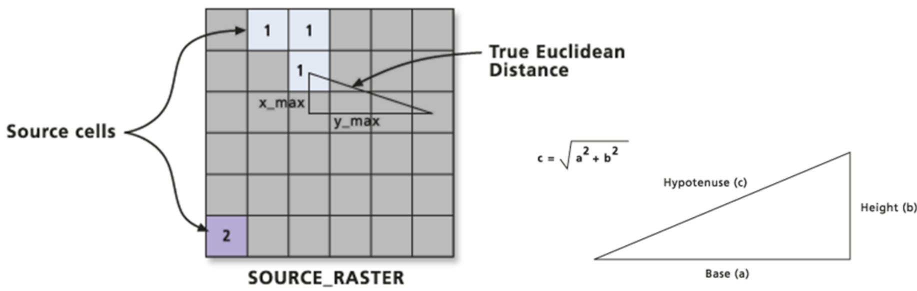
- Operate on the **entire grid** at once
- Each cell is a **function** of all input cells
- Distance functions, Hydrological, functions, Viewshed, Max,…





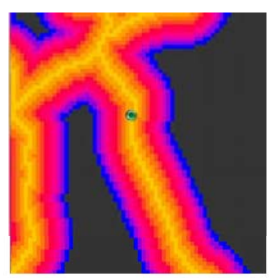
Distance Calculation in Raster Data

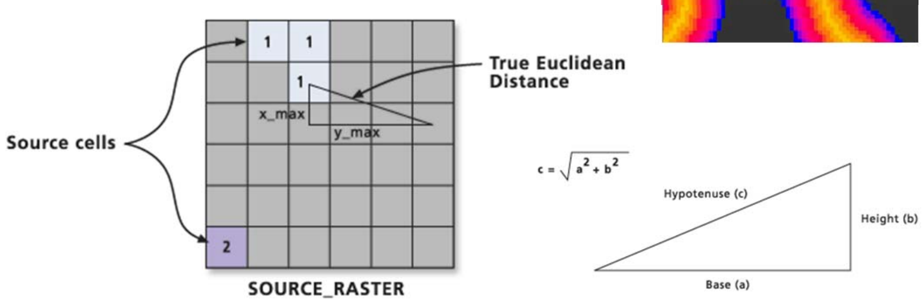


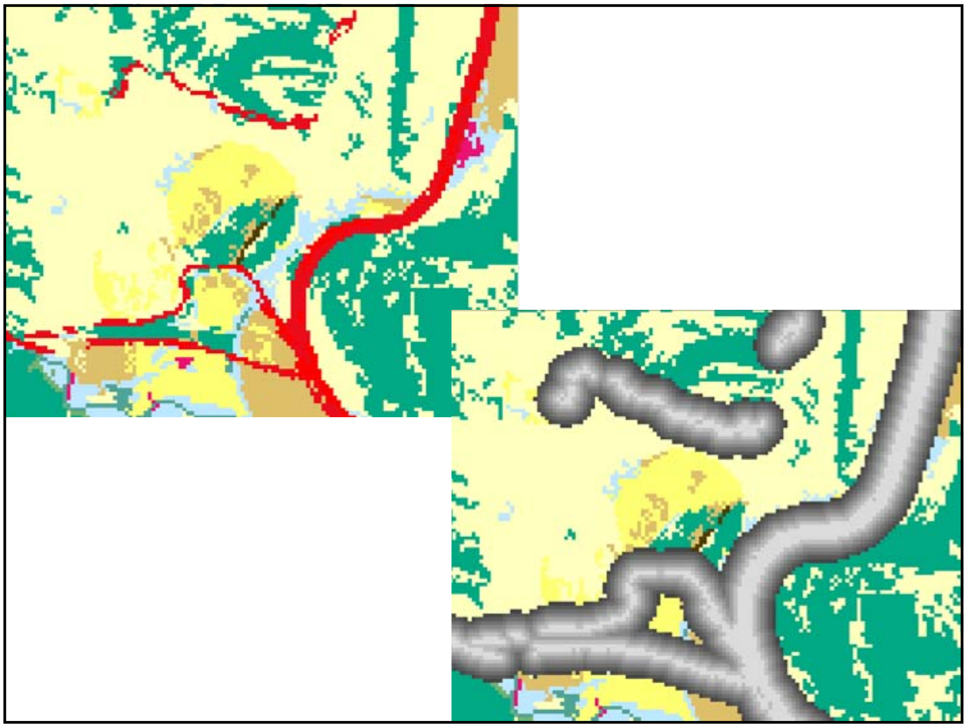


Distance Calculation in Raster Data

- Euclidean Distance
- Path Distance/ Terrain Distance
- Cost Distance







Class 09b Exercise: Calculating Distance Grids

Your task is to calculate a distance raster. The cell values of this raster dataset will represent the **distance (Euclidean) of each cell to the nearest developed cell** (NLCD classes 21, 22 or 23). We apply a cut-off value (for efficiency) for distances of **180m** which means that distance values in any direction that are greater than 180m will be ignored. Obviously, we are only interested in values less than that cut-off value which may be the case in models of **suitability for residential land** or similar applications.
The distances should be calculated for all pixels within a certain **distance band** around occurrences of developed land. Think about a strategy that you could apply in order to: **(1)** identify all pixels that are developed, **(2)** calculate distances of all values that are not developed nearby, and **(3)** how to limit the distance values to a maximum of 180m. You will have to think about how to assign the minimum distance value of a cell that has two or more developed land pixels nearby. You will also have to think about how to deal with no-data values.
Use the NLCD subset "input_l" in the \\data folder of your GIS3 folder. The figures below show the resulting distance grid (right panel) on top of the original NLCD (left panel).