Exercise 4: Working with the ArcPy Spatial Analyst module

Your task in this exercise is to use the ArcPy Spatial Analyst module and map algebra functions to find potential gold deposits in Lander County, Nebraska. Most of Nevada's current gold production is from Carlin-type deposits. The following characteristics will be used in attempting to identify Carlin-type gold deposits:

- High PPM (parts per million) of arsenic or antimony in sediment samples
- Low PPM of potassium in sediment samples
- Found mainly in Paleozoic and Mesozoic geologic formations

Fuzzy logic tools will be used to identify areas within Lander County that meet these criteria. The use of fuzzy logic will enable our identification to be **probability** based, as opposed to based on simple binary logic where an area is either suitable or unsuitable.

You will complete a Python script that performs multiple steps including interpolation, fuzzy overlay, map algebra, and conditional logic, all using the ArcPy Spatial Analyst module.

Examine data

- Navigate to folder …Python_ArcGIS\exercises\exercise4_sa
- Double-click GoldInThemHills.mxd to open it in ArcMap, and examine the data layers that are available in the table of contents.
- The layer 'Sediment Samples' contains sample values for mineral composition and the 'Geology' layer is a raster representing geologic formations.
- The data needed to complete this exercise are stored at …Python_ArcGIS\exercises\exercise4_sa\data\Carlin.gdb
- Close ArcMap

Setting up the script

- Open PythonWin
- File>Open to the folder …Python_ArcGIS\exercises\exercise4_sa\scripts
- Open the Python script GoldInThemHills.py
- The script file starts the normal way: some comments and then imports of arcpy, os, and all ArcPy Spatial Analyst module functions

```
#import system modules
import arcpy
from arcpy.sa import *
import os
```

- Also completed in this script is the setting of a number of important geoprocessing environments like workspace, scratch workspace, processing extent, mask, and cell size.
    - Setting these environments is especially important when working with the ArcPy Spatial Analyst module because the workspace and scratch workspace environments determine where permanent and temporary rasters are stored, and the extent and mask environments control the geographic area that will be included in the analysis and output rasters.

```
#set geoprocessing environments
arcpy.env.overwriteOutput = True
arcpy.env.workspace = os.path.join(os.path.dirname(os.getcwd()), "data", "Carlin.gdb")
arcpy.env.scratchWorkspace = os.path.join(os.path.dirname(os.getcwd()), "data", "scratch")
arcpy.env.extent = "LanderCounty"
arcpy.env.mask = "LanderCounty"
arcpy.env.cellSize = "Geology"
```

- To use any Spatial Analyst functions, the extension must also be checked out.

```
arcpy.CheckOutExtension("Spatial")
```

Perform raster analysis

In addition to the Desktop Help **Geoprocessing Tool Reference** book, a good way to get help on how to use a tool is to use the built-in Python help() function. In the PythonWin interactive window, type ***import arcpy*** then press enter. Next, type ***from arcpy.sa import ****, then press enter. Finally, call the help function by putting the name of the function inside help(); for example, ***help(FuzzyOverlay)***. You can use this same technique with other tools not in the Spatial Analyst module by following this standard: arcpy.toolname_toolbox, for example, ***help(arcpy.Buffer_analysis)***.

- The first step is to create interpolated surfaces representing the arsenic, antimony, and potassium composition at the sediment sample locations. Use the NaturalNeighbor function to perform the interpolation.
    - The first parameter of the NaturalNeighbor function is an input point feature class, and the second parameter is the attribute field containing values to interpolate

```
samples = "SedimentSamples"

#create surfaces
antimony = NaturalNeighbor(samples, "NAA_SB")
arsenic = NaturalNeighbor(samples, "NAA_AS")
potassium = NaturalNeighbor(samples, "NAA_K")
```

- With these interpolated surfaces you can begin preparing for the fuzzy overlay operations by transforming the rasters into fuzzy membership distributions where values range from 0 to 1. Use the FuzzyMembership function to perform these operations.
    - The first parameter of the FuzzyMembership function is an input raster, and the second parameter is the algorithm to use in the fuzzy transformation
    - Since we are looking for areas with high antimony and arsenic samples, a FuzzyMSLarge algorithm is used for the antimony and arsenic fuzzy membership rasters. The FuzzyMSLarge algorithm calculates the transformed values based on the mean and standard deviation of the input raster values, with **larger values having transformed values closer to 1**.
    - Contrarily, we are looking for areas with low potassium samples, so a FuzzyMSSmall algorithm is used for the potassium fuzzy membership raster. The FuzzyMSSmall algorithm also calculates the transformed values based on the mean and standard deviation of the input raster values, but **smaller values will have transformed values closer to 1**.

```
#calculate fuzzy membership
antimonyFMS = FuzzyMembership(antimony, FuzzyMSLarge(1, 1))
arsenicFMS = FuzzyMembership(arsenic, FuzzyMSLarge(1, 1))
potassiumFMS = FuzzyMembership(potassium, FuzzyMSSmall(1, 1))
```

- Since we are looking for areas that are **high** in arsenic or antimony, use the FuzzyOverlay function to combine the antimony and arsenic fuzzy membership rasters. The output of this overlay will have high values where **at least one** of the input rasters have a high value (with the "OR" option, the output raster takes the maximum value of the input rasters).

```
#combine antimony and arsenic fuzzy members
mineralfactor = FuzzyOverlay([antimonyFMS, arsenicFMS], "OR")
```

- Next, use the FuzzyOverlay function to find areas that are **high** in either arsenic or antimony and **low** in potassium. The output of this overlay will have high values where **ALL** of the input rasters have a high value, and low values where **ANY** of the input rasters have a low value (with the "AND" option, the output raster takes the minimum value of the input rasters).

```
#combine mineral factor with potassium membership
lithadjmineralfactor = FuzzyOverlay([mineralfactor, potassiumFMS], "AND")
```

- ALTERNATIVE - Instead of running these two overlays on multiple lines, they can be nested into a single line that performs both operations. Nesting refers to using a call to a function (and the resulting output) as the input to another function. These fuzzy overlay operations could be nested as follows:

```
#ALTERNATE way to combine antimony and arsenic fuzzy membersship with potassium membership
lithadjmineralfactor = FuzzyOverlay([FuzzyOverlay([antimonyFMS, arsenicFMS], "OR"), potassiumFMS], "AND")
```

The remaining criterion needed for the model relates to the underlying geologic formations (Paleozoic and Mesozoic geologic formations are most suitable; other formations are various degrees of suitability). The Geology raster dataset is a categorical raster – it has zones that define areas that share a geologic formation. To combine this kind of categorical information with our fuzzy membership information on mineral composition, the categorical Geology raster must be reclassified.

- Use the Reclassify function and a list of formation codes and their corresponding suitability score to reclassify the geology raster.
    - Copy and paste this line into your script:

      remap = [['TPC', 50], ['TRPE', 60], ['TMF', 50], ['TMV', 50], ['UPZ', 90], ['LMZ', 40], ['Q', 50], ['LPZ', 90], ['QV', 50], ['TPV', 50], ['TPF', 50], ['KG', 0], ['C', 85], ['UPZE', 90], ['P', 60], ['TI', 0], ['LTV', 50], ['LPZE', 90], ['JG', 0], ['KC', 0], ['UPZC', 75], ['LMZV', 55], ['TRG', 0], ['KG2', 0], ['JMI', 0]]

```
#reclassify the geologies into continuous values (close to 100=good, close to 0=bad)
remap = [['TPC', 50], ['TRPE', 60], ['TMF', 50], ['TMV', 50], ['UPZ', 90], ['LMZ', 40], ['Q', 50],
georeclass = Reclassify("Geology", "CODE", RemapValue(remap))
```

- Even though the geology raster is reclassified to have numeric values representing formation suitability, to work with the fuzzy logic functions we need to transform the raster into a floating point raster with values between 0 and 1. Use a map algebra division operator / and divide by a number with a decimal to perform this transformation.

```
#the reclassified geology must be converted to a floating point raster, and values divided by 100
geofactor = georeclass / 100.0
```

- Combine the mineral factor (areas high in arsenic or antimony and low in potassium) with the geologic factor (areas with a suitable geologic formation) using the FuzzyOverlay function, and save the output raster class.

```
#combine the geology factor and mineral factor with Fuzzy overlay
goldpossibility = FuzzyOverlay([lithadjmineralfactor, geofactor], "GAMMA", 0.9)
goldpossibility.save("GoldPossibility")
```

- Find the areas where there is a high possibility (0.70) of finding gold, given the defined mineral and geologic characteristics, and save the output raster.

```
#extract the areas that have a fuzzy possibility of > 0.70
bestchance = SetNull(goldpossibility < 0.70, 1)
bestchance.save("BestChance")
```

- Press the Check button to check the script for syntax errors like indentation mistakes

- Press the Run button and run the full script

Results

The ArcPy Spatial Analyst module provides powerful and efficient functions for performing raster analysis, including interpolation, fuzzy overlay, map algebra, and conditional logic. In addition to providing access to this analytical functionality, being able to script workflows that can be easily shared and re-run can make Python scripting in ArcGIS an invaluable tool.

The results of the analysis reveal areas in which the mineral and geologic characteristics may lead us to where gold deposits can be found. Maybe there is gold in them hills!