SUPPLEMENTARY MATERIAL – Python Scripts

Scripts

Script 1 Script that was used to automatically assign scores for community income levels

```
##Name of script: Income2001.py
##Purpose: Automated allocation of scores to communities for the 2001 income levels
data
#Importing system modules
import arcpy
#Input data and input fields and their lengths
table = "E:/GIS/RayMhlaba.mdb/Income_2001"
fields = ["No_income","R1_R9600","R9601_R19200","R19201_R38400","R38401_more"]
#Adding fields to input table to store maximum and field names
maxfield = "HIGHEST"
maxname = "HIGHEST_name"
scores = "In2001_score"
arcpy.AddField_management(table, maxfield, "TEXT")
arcpy.AddField_management(table, maxname, "TEXT")
arcpy.AddField_management(table, scores, "SHORT")
#Adding created fields to the array
fields2 = fields[:] # Shallow copy
fields2.extend([maxfield, maxname, scores])
#Checking and updating of fields
with arcpy.da.UpdateCursor(table, fields2) as cursor:
    for row in cursor:
        arrayVals = [row[0], row[1], row[2], row[3], row[4]]
        highest = max(arrayVals)
        row[5] = highest
        row[6] = fields[arrayVals.index(highest)]
        if row[6] == "No_income":
            row[7] = 0
            cursor.updateRow(row)
        elif row[6] == "R1_R9600":
            row[7] = 1
            cursor.updateRow(row)
        elif row[6] == "R9601_R19200":
            row[7] = 2
            cursor.updateRow(row)
        elif row[6] == "R19201_R38400":
            row[7] = 3
            cursor.updateRow(row)
        else:
            row[7]= 4
            cursor.updateRow(row)
```

Script 2 Script that was used for automated creation of a new shapefile and attribute table and joining of Age_Profile scores to the IndicatorScores table

```
##Name of script: ScoreInput2001.py
##Purpose: Automated creation of a new shapefile and attribute table for saving an
integration of the four indicator scores and automated joining of the remaining 3
indicator scores to the IndicatorScores table using MP_NAME as the linking field.

#Importing system modules
import arcpy
from arcpy import env
#Setting the environment
env.workspace = "E:/GIS/"
# Specifying the input feature class, output location and feature classes
inFeatures = "E:/GIS/RayMhlaba.mdb/Age_2001"
outLocation = "E:/GIS/RayMhlaba.mdb"
outFeatureClass = "Ind_Scores2001"
```

```
# Listing fields to be retained
myfields = ["MP_NAME", "A2001_score"]
# Creating an empty field mapping object
mapS = arcpy.FieldMappings()
# Creating an individual field map each field, and adding it to the field mapping
object
for field in myfields :
    map = arcpy.FieldMap()
    map.addInputField(inFeatures, field)
    mapS.addFieldMap(map)
    # Copying the feature class using the fields
    arcpy.FeatureClassToFeatureClass_conversion (inFeatures, outLocation,
outFeatureClass, field_mapping=mapS)
#Joining the remaining 3 indicator scores for different fields into one table
arcpy.JoinField_management("Ind_Scores2001", "MP_NAME", "Income_2001", "MP_NAME",
"In2001_score")
arcpy.JoinField_management("Ind_Scores2001", "MP_NAME", "Literacy_2001", "MP_NAME",
"Lit2001_score")
arcpy.JoinField_management("Ind_Scores2001", "MP_NAME", "Water_2001", "MP_NAME",
"W2001_score")
```

Script 3: Script that was used for joining of the remaining 3 indicator scores to the IndicatorsScores table using MP_NAME as the linking field

```
## Name of script: AD2001.py
## Purpose: Automated summation and ranking of the four indicator scores for year
2001
#Importing system modules
import arcpy, math
#Importing scores from indicator attribute tables into the Adaptive capacity
shapefile
table = "E:/GIS/RayMhlaba.mdb/Ind_Scores2001"
fields = ["A2001_score","In2001_score","Lit2001_score", "W2001_score"]
# Adding fields to input table to store maximum and field name
total = "AD2001_Score"
rating = "ACRating_2001"
arcpy.AddField_management(table, total, "SHORT")
arcpy.AddField_management(table, rating, "TEXT")
#Adding created fields to the array
fields2 = fields[:]
fields2.extend([total, rating])
#Classifying community-level adaptive capacity scores
with arcpy.da.UpdateCursor(table, fields2) as cursor:
    for row in cursor:
        arrayVals = [row[0], row[1], row[2], row[3]]
#Calculating adaptive capacity for each community by summing the 4 indicator scores
        summation = sum(arrayVals)
        row[4] = summation
        #Allocating the adaptive capacity rating
        if row[4] <=5:
            row[5] = 'LOW'
            cursor.updateRow(row)
        elif row[4]> 5 and row[4]<=10:
            row[5] = 'MEDIUM'
            cursor.updateRow(row)
        else:
            row[5]='HIGH'
            cursor.updateRow(row)
```

Script 4: Script that was used for joining of the remaining 3 indicator scores to the IndicatorsScores table using MP_NAME as the linking field

```
## Name of script: AD_Change.py
## Purpose: Automated creation of a new shapefile and saving adaptive capacities
for years 2001 and 2011 into one attribute table
#Importing system modules
import arcpy
from arcpy import env
#Setting the environment
env.workspace = "E:/GIS/"
# Specifying of input feature class, output location and feature classes
inFeatures = "E:/GIS/RayMhlaba.mdb/Ind_Scores2011"
outLocation = "E:/GIS/RayMhlaba.mdb"
outFeatureClass = "AD_Diff"
# Listing of fields to be retained
myfields = ["MP_NAME", "AD2011_Score"]
# Creating an empty field mapping object
mapS = arcpy.FieldMappings()
# Creating an individual field map for each field and adding it to the field
mapping object
for field in myfields :
    map = arcpy.FieldMap()
    map.addInputField(inFeatures, field)
    mapS.addFieldMap(map)
    # Copying the feature class using the fields
    arcpy.FeatureClassToFeatureClass_conversion(inFeatures, outLocation,
outFeatureClass, field_mapping=mapS)
#Joining of Adaptive capacity 2001 field to the Adaptive capacity 2011 field to
create one table
arcpy.JoinField_management ("AD_Diff", "MP_NAME", "Ind_Scores2001", "MP_NAME",
"AD2001_Score")
```

Script 5 Script that was used for calculating changes in adaptive capacities between 2001 and 2011

```
## Name of script: Diff.py
## Purpose: Calculating changes in adaptive capacities between years 2001 to 2011
#Importing system modules
import arcpy, math
# Selecting fields of interest from attribute table in shapefile
table = "E:/GIS/RayMhlaba.mdb/AD_Diff"
fields = ["AD2011_Score","AD2001_Score"]
# Adding new fields to table to store calculated differences in adaptive capacity
difference = "AC_Resultant"
change = "AC_Change"
arcpy.AddField_management(table, difference, "SHORT")
arcpy.AddField_management(table, change, "TEXT")
#Adding created fields to the array
fields2 = fields[:]
fields2.extend([difference, change])
#Classifying the changes in adaptive capacity
with arcpy.da.UpdateCursor(table, fields2) as cursor:
    for row in cursor:
        arrayVals = [row[0], row[1]]
        #Subtracting the 2001 adaptive capacity from the 2011 adaptive capacity
        row[2] = row[0] - row[1]
        #Allocating the adaptive capacity change
        if row[2] <= -1:
            row[3] = 'DECREASE'
            cursor.updateRow(row)
        elif row[2] == 0:
            row[3] = 'NO CHANGE'
            cursor.updateRow(row)
        else:
            row[3]='INCREASE'
            cursor.updateRow(row)
```

Script 6 Script that was used for calculating the final adaptive capacity using average soil moisture, arable lands and socio-economic data

```
## Name of script: Final_AD.py
## Purpose: Summation and ranking of the three indicators for assessing adaptive
capacity
#Importing system modules
import arcpy, math
#Importing scores from indicator attribute tables into the Adaptive capacity
#shapefile
table = " C:/RM/RM.mdb/RM6_Intersect"
fields = ["gridcode","AD_Score","ARL_Score"]
# Adding fields to input table to store maximum and field name
total = " Final_Score"
rating = " Final_Rating"
arcpy.AddField_management(table, total, "SHORT")
arcpy.AddField_management(table, rating, "TEXT")
#Adding created fields to the array
fields2 = fields[:]
fields2.extend([total, rating])
#Classifying community-level indicator scores
with arcpy.da.UpdateCursor(table, fields2) as cursor:
    for row in cursor:
        arrayVals = [row[0], row[1], row[2]]
#Calculating adaptive capacity for each community by summing the 3 indicator scores
        summation = sum(arrayVals)
        row[3] = summation
        #Allocating the final adaptive capacity rating
        if row[3] <=4:
            row[4] = 'LOW'
            cursor.updateRow(row)
        elif row[3]> 4 and row[3]<=6:
            row[4] = 'MEDIUM'
            cursor.updateRow(row)
        else:
            row[4]='HIGH'
            cursor.updateRow(row)
```