**Appendix D. Scripts**

1. Join SSURGO

import pandas as pd

import numpy as np

import os

try:

import arcpy

path2collectionOfssurgoFolders = arcpy.GetParameterAsText(0)

lookupTable = arcpy.GetParameterAsText(1)

except Exception, e:

print e

if path2collectionOfssurgoFolders =="":

# Input a folder that has all the folders of names similar to UT012, Ut027 etc.

path2collectionOfssurgoFolders = r"E:\Research Data\del\_ssro"

if lookupTable == "":

lookupTable = os.path.join(os.getcwd(), "GREENAMPT\_LOOKUPTABLE.csv")

def step3\_merge\_ssurgo(path2collectionOfssurgoFolders ,path2lookupTable=lookupTable ):

"""

:param path2collectionOfssurgoFolders: The path to a folder containing the collection of SSURGO (or Statsgo) folders

:param path2lookupTable: The greenampt csv lookup table that with soil properties for each soil texture classes

:return: a csv file in each ssurgo folders, that has soil properties calculated for each map units

"""

lookupTable = pd.read\_csv(path2lookupTable , sep=',', skiprows = 0)

# create a list of folders only

folderList = []

[folderList.append(folders) for folders in os.listdir(path2collectionOfssurgoFolders)

if os.path.isdir(os.path.join(path2collectionOfssurgoFolders, folders))]

for folder in folderList:

path2ssurgo= os.path.join(path2collectionOfssurgoFolders , folder)

path2tabular = os.path.join(path2ssurgo, "tabular")

path2Spatial= os.path.join(path2ssurgo, "spatial")

# Make changes here! The values that we need to average

valuesToAvg = ['ksat\_r','Ks','dbthirdbar\_r','dbfifteenbar\_r', 'ResidualWaterContent', 'Porosity',

'EffectivePorosity', 'BubblingPressure\_Geometric', 'PoreSizeDistribution\_geometric']

# fileNameColNoListHeaders [ [filename, [column numbers for fields to pull up], [col headers to be assigned]], ]

# the number start from 0, so 1 is actually the second column/field

fileNameColNoListHeaders = [ ["comp",[1,5,107,108],["ComponentPercent","MajorComponent", "MUKEY","COKEY"]],

["muaggatt",[10,17,39],["AvaWaterCon","HydroGrp","MUKEY"]],

["chorizon",[6,9,12,81,72,75,169,170],["TopDepth","BottomDepth", "HorizonDepth","ksat\_r","dbthirdbar\_r","dbfifteenbar\_r","COKEY","CHKEY"]],

["chtextur",[0,2,3],["textureName","CHtxtgrpKEY","CHTXTKEY"]],

["chtexgrp",[4,5],["CHKEY","CHtxtgrpKEY"]]

]

def STEP1\_rawToRefined( fileName\_ColNoList\_Headers, path=path2tabular):

"""

:param fileName\_ColNoList\_Headers: the list (filename, col numbers, names to the col)

:param path: path2tabular

:return: file in the memory, as panda dataframe

"""

for afileColHdr in fileName\_ColNoList\_Headers:

txtFilename= afileColHdr[0]

colNo = afileColHdr[1]

header = afileColHdr[2]

txtFile = os.path.join(path, txtFilename + ".txt") #RETURNS FULL ADDRESS

csvFileData = pd.read\_csv(txtFile, sep = "|", header=None, comment='#')

reqdData = csvFileData.iloc[:,colNo]

reqdData.columns = header

reqdData.to\_csv(os.path.join(path , txtFilename + ".csv"), index=False)

return reqdData

# Merges the CSV files read earlier

def STEP2\_mergeCSV(path=path2tabular):

muaggatt = pd.read\_csv(path+"/muaggatt.csv") ; print "/muaggatt.csv", len(muaggatt.index)

component = pd.read\_csv(path+"/comp.csv") ; print "/comp.csv", len(component.index)

chorizon = pd.read\_csv(path+"/chorizon.csv") ; print "/chorizon.csv", len(chorizon.index)

chtextur = pd.read\_csv(path+"/chtextur.csv") ; print "/chtextur.csv", len(chtextur.index)

chtexgrp = pd.read\_csv(path+"/chtexgrp.csv") ; print "/chtexgrp.csv", len(chtexgrp.index)

component\_Muaggatt = pd.merge(muaggatt , component, on='MUKEY')

chorizon\_Component\_Muaggatt = pd.merge(component\_Muaggatt , chorizon, on='COKEY')

chTxt\_chTxtGrp = pd.merge(chtextur , chtexgrp, on='CHtxtgrpKEY')

merged = pd.merge(chTxt\_chTxtGrp , chorizon\_Component\_Muaggatt, on='CHKEY')

# print chorizonWithComponent

merged.to\_csv(path + "/MERGED.csv", index=False)

return merged

# \_\_main\_\_

try:

STEP1\_rawToRefined(fileNameColNoListHeaders) ; print "Headers applied to raw txts"

mergdf = STEP2\_mergeCSV() ; print "Merging completed"

mergeWithLookUp = pd.merge(mergdf, lookupTable, on= 'textureName') #>result: OverallMergedWithTexture.csv

mergeWithLookUp.to\_csv(os.path.join(path2tabular, "OverallMergedWithTexture.csv"), index=False)

print "Merging with texture lookup table completed"

except Exception, e:

print e

# Calculations part

try:

# STEP4 Take i)Height Weighted Average ii)Component % weighted average --------> result MUKEY-Vs-Values.csv

merged = pd.read\_csv(os.path.join(path2tabular, "OverallMergedWithTexture.csv"))

# Caclulation of weighted average

HorizonDepth2 = merged['BottomDepth'] - merged['TopDepth'] ; merged.loc[:,'HorizonDepth2']= HorizonDepth2

# the values whose weighted average we want, needs to be given in the list below

# -------> MUKEY Vs Value (just one) MUKEY-Value.csv

for valueName in valuesToAvg: # add those values to merged

VxD = merged['HorizonDepth2']\* merged[valueName] ; merged.loc[:,valueName+"xD\_sum"]= VxD

chorizonCalc = merged.groupby('COKEY').agg({valueName+"xD\_sum":np.sum , 'HorizonDepth2':np.sum,'ComponentPercent':np.max,'COKEY':np.max,'MUKEY':np.max })

chorizonCalc=chorizonCalc.rename(columns = {'HorizonDepth2':'HorizonDepth2\_sum'}) # because grouping by cokey, the column name doesnt match its data

VxD\_by\_sum = chorizonCalc[valueName+"xD\_sum"].astype('float').div(chorizonCalc['HorizonDepth2\_sum'].astype('float'))

chorizonCalc.loc[:,valueName+"\_avgH"]= VxD\_by\_sum

# percentage weightage

compPerc\_X\_Havg = chorizonCalc['ComponentPercent'].astype('float')/100. \* chorizonCalc[valueName+"\_avgH"]

chorizonCalc.loc[:,valueName+"\_WtAvg"] = compPerc\_X\_Havg

# now Group it by MUKEY, and done!

componentPercentageCalc = chorizonCalc.groupby('MUKEY').agg({'MUKEY':np.max, valueName+"\_WtAvg":np.sum })

componentPercentageCalc.to\_csv(os.path.join(path2tabular, "MUKEY-"+ valueName +".csv"), index=False)

# now, function to use the 'valuesToAvg' list above, and merge them against MUKEY

mukeyValues = componentPercentageCalc.MUKEY

except Exception, e:

print e

try:

# STEP5: Merge all the MUKEY Vs Values csv --------> result MUKEY-Vs-Values.csv

lastValueFile = pd.read\_csv(path2tabular+"\\MUKEY-"+ valuesToAvg[-1] +".csv")

for valueName in valuesToAvg:

# if valueName == valuesToAvg[-1] : break

fl = pd.read\_csv(path2tabular+"\\MUKEY-"+ valueName +".csv")

print path2tabular+"\\MUKEY-"+ valueName +".csv"

lastValueFile = pd.merge(lastValueFile, fl, on="MUKEY")

# Print mukeyValuesAllMerged

lastValueFile.to\_csv(path2ssurgo+"\\MUKEY-Vs-Values.csv", index=False)

print 'All values table written down in the ssurgo folder'

# Create a schema.ini so that arcGIS can understand the MUKEY field

schema = open(path2ssurgo+"\\schema.ini", "w")

schema.write("[MUKEY-Vs-Values.csv]"+ "\n" + "Col2=MUKEY Text") #may not always be column 1 though

schema.close()

except Exception, e:

print e

try:

## adding Soil group to the final table

final\_table = pd.read\_csv(os.path.join(path2collectionOfssurgoFolders, folder, "MUKEY-Vs-Values.csv"))

muaggat = pd.read\_csv(os.path.join(path2collectionOfssurgoFolders, folder,"tabular", "muaggatt.csv"))

# remove duplicate Soil group elements

# may be replacement is not advisable. Need to double check on this

muaggat= muaggat.replace("A/B", "A")

muaggat=muaggat.replace("A/C", "A")

muaggat=muaggat.replace("B/D", "D")

muaggat=muaggat.replace("B/C", "B")

muaggat=muaggat.replace("B/D", "B")

muaggat=muaggat.replace("C/D", "C")

muaggat.to\_csv(os.path.join(path2collectionOfssurgoFolders, folder,"tabular", "muaggatt\_Removed\_HydrGRP.csv"), index=False)

merge\_soilGRP\_final = pd.merge(final\_table, muaggat, on= 'MUKEY')

merge\_soilGRP\_final.to\_csv(os.path.join(path2collectionOfssurgoFolders, folder , "MUKEY-Vs-Values.csv"), index=False)

print "Merging Soil Hydrologic Group Complete"

# delete all the csv files made so far, except the MUKEY-Vs-Values.csv

filelist = [ f for f in os.listdir(path2tabular) if f.endswith(".csv") ]

for f in filelist:

os.remove(os.path.join(path2tabular, f))

except Exception,e :

print "Merging the Hydrologic Soil Group failed with the error %s"%e

if \_\_name\_\_ == "\_\_main\_\_":

step3\_merge\_ssurgo( path2collectionOfssurgoFolders, lookupTable)

**STEP3: Subbasin\_properties\_table**

import arcpy

import os

subbasin= arcpy.GetParameterAsText(0)

raster\_folder = arcpy.GetParameterAsText(1)

outGDB= arcpy.GetParameterAsText(2)

if subbasin == "":

subbasin = r"C:\Users\WIN10-HOME\Dropbox\CLASSES\CEE6450 Hydrological Modeling\Project\_cee6450\hechms\_Prasanna\_Sal\_RBC\HECHMS\_Projects\hecgeohms.gdb\Catchment"

raster\_folder = r"E:\Research Data\00 Red Butte Creek\RBC\_3\tif"

outGDB = r"E:\Research Data\00 Red Butte Creek\RBC\_3\New File Geodatabase (2).gdb"

# create a GDB if user does not specify

if outGDB == "":

try:

arcpy.CreateFileGDB\_management(raster\_folder, "TABLES.gdb")

outGDB = os.path.join(raster\_folder, "TABLES.gdb")

except Exception, e:

arcpy.AddMessage(outGDB)

arcpy.env.overwriteOutput = True

arcpy.env.workspace = arcpy.env.scratchWorkspace = outGDB

list\_of\_rasters = [os.path.join(raster\_folder, file) for file in os.listdir(raster\_folder) if file.endswith(".tif")]

mxd = arcpy.mapping.MapDocument("CURRENT") # get the map document

df = arcpy.mapping.ListDataFrames(mxd,"\*")[0] #first dataframe in the document

for aRaster\_fullname in list\_of\_rasters:

raster\_layer = arcpy.mapping.Layer(aRaster\_fullname ) # create a new layer

arcpy.mapping.AddLayer(df, raster\_layer ,"TOP")

table\_name = os.path.basename(aRaster\_fullname).split(".")[0]

arcpy.AddMessage("%s and %s"%(raster\_layer,table\_name) )

arcpy.gp.ZonalStatisticsAsTable\_sa(subbasin, "GridID", raster\_layer, table\_name , "DATA", "MEAN")

arcpy.AddField\_management(in\_table=table\_name, field\_name=table\_name, field\_type="DOUBLE",

field\_precision="", field\_scale="", field\_length="", field\_alias="",

field\_is\_nullable="NULLABLE", field\_is\_required="NON\_REQUIRED", field\_domain="")

arcpy.CalculateField\_management(in\_table=table\_name, field=table\_name,

expression="[MEAN]", expression\_type="VB", code\_block="")

for aRaster\_fullname in list\_of\_rasters:

table\_name = os.path.basename(aRaster\_fullname).split(".")[0]

arcpy.AddJoin\_management(in\_layer\_or\_view=subbasin,

in\_field="GridID",

join\_table=table\_name,

join\_field="GridID", join\_type="KEEP\_ALL")

arcpy.TableToExcel\_conversion(Input\_Table=subbasin,

Output\_Excel\_File=os.path.join(raster\_folder, "Subbasin\_characteristics.xls"),

Use\_field\_alias\_as\_column\_header="NAME",

Use\_domain\_and\_subtype\_description="DESCRIPTION")