GIS Analysis with Application of Python Programming of Small-Scale Channel Widening Experiment

Mariel Jumawan

PGEO6050

Fall 2020

INTRODUCTION

In the National Sedimentation Laboratory, a group of scientists investigated the occurrence of small-scale channel widening in the presence of a less erodible base layer. To ensure the morphological change is through lateral expansion, the channel was constricted to a vertical position. The initial and final time for the experiment were, consecutively, 0 seconds and 2090 seconds, with images collected in 10-second intervals. The objective of this report is to analyze the given datasets with the application of ArcMAP and python programming and calculate the following deliverables: generate horizontal profiles, generate channel width for a selected time step, calculate descriptive statistics for a selected time step, and generate the average channel width for all time steps.

METHODS

Description of datasets

Geo-referenced images: **T10geo.tif and T2100geo.tif**

- What information is provided in the attribute table? No attribute table.
- What is the ArcGIS and Python data types for each field? No attribute table.
- What is the geometry type of this dataset? No geometry type.
- What are the boundary conditions of this problem (time and space)? X and Y values.

Polygon with channel edges: **P2016-07-19.shp**

- What information is provided in the attribute table? FID, Shape, Time.
- What is the ArcGIS and Python data types for each field?

	ArcGIS	Python
FID	Object ID	Integers
Shape	Geometry	Strings
Time	Long	Integers

- What is the geometry type of this dataset? Geometry Type – Polygon.
- What are the boundary conditions of this problem? X and Y values.

<u>List of geoprocessing tools to generate your final output</u>

(Images of tools and its outputs are displayed within the code for a better understanding of the process of analysis)

- **Create Feature Class**
- **Select Layer By Attribute**
- **Intersect Analysis**

<u>Pseudo code</u>

GENERATING HORIZONTAL PROFILES

1. Create feature class with ArcMap tool
2. Establish x and y coordinates
3. Use for loop
    a. Use insert cursor
    b. Add equation to calculate channel width: A-J use 2 cm
    c. Delete cursor

GENERATING CHANNEL WIDTH FOR A SELECTED TIME SETUP

1. Select layer with ArcMap tool
2. Intersect layers
3. Write csv file
    a. Open and write file
    b. Name file headers
    c. Use search cursor
    d. Use for loop
        i. Create empty list for row 1 and row 2
        ii. Convert list to strings
        iii. Write
    e. Close file
    f. Delete cursor

CALCULATING DESCRIPTIVE STATISTICS FOR A SELECTED TIME SETUP

1. Import libraries
2. Create open list
3. Load csv file created in previous section
4. Open new csv file
    a. Open and write file
    b. Name file headers
    c. Use for loop
5. Write code for mean, max value, min value, mode, median, and standard deviation.
6. Write code to csv file
7. Close file

GENERATING AVERAGE CHANNEL WIDTH FOR ALL TIME SETUPS

- Import libraries
- Load new file
- Create two empty lists
- Plot table
  - Determine x and y values
  - Label x axis
  - Label y axis
  - Set x and y axis limits
- Create regression model
  - Determine x and y and labels
  - Add legend and gridlines
- Create plot as figure
  - Set figure size
  - Name figure output
  - Close figure

Generating horizontal profiles

```
#create featureclass
import arcpy
arcpy.CreateFeatureclass_management("D:/Graduate/2020Fall_PythonProgramming/te
rmProject","Lines1.shp","POLYLINE")
cursor = arcpy.da.InsertCursor("Lines1",['SHAPE@'])

#generate information (X and Y coordinates)
Xright = 100.875
Xleft = 99.95
Ymin = 100.325
Ymax = 101.81

#generate horizontal profiles
for i in range(200):
    array = arcpy.Array([arcpy.Point(Xleft,Ymin),arcpy.Point(Xright,Ymin)])
    multiline = arcpy.Polyline(array)
    cursor.insertRow([multiline])
    Ymin = Ymin + 0.02
    if (Ymin > Ymax):
        break
del cursor
```

Figure 1. Screenshot of the ArcMap tool ‘Create Feature Class’ to begin the creation of horizontal profiles.

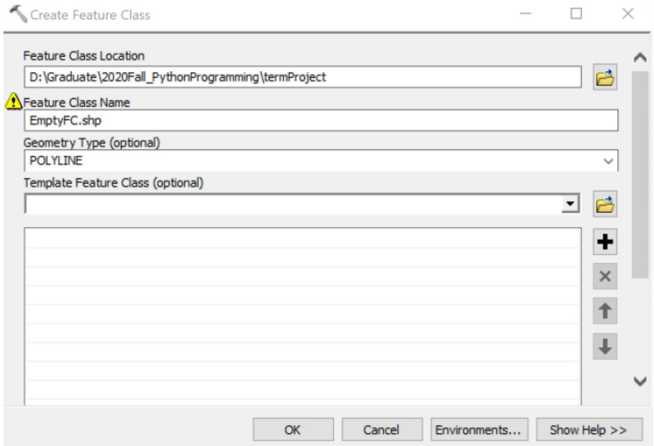
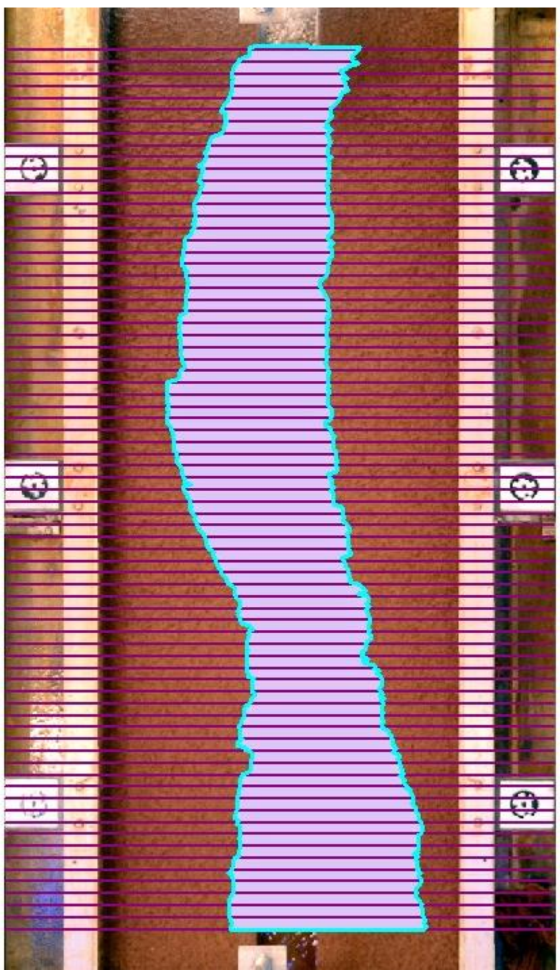


Figure 2. Output of the ‘Create Feature Class’ tool. File name: Lines1.



Generating channel width for a selected time step

```
#select by attribute
# Replace a layer/table view name with a path to a dataset (which can be a layer file) or
create the layer/table view within the script
# The following inputs are layers or table views: "P2016JUL19"
arcpy.SelectLayerByAttribute_management("P2016JUL19",
"NEW_SELECTION",'"TIME" =1500')

#intersect analysis
arcpy.Intersect_analysis(["P2016JUL19,Lines1"],
"D:/Graduate/2020Fall_PythonProgramming/termProject/time_1500.shp","ALL","-1
Unknown","LINE")

#writing to csv file
f = open(r'D:\Graduate\2020Fall_PythonProgramming\termProject\time_1500.csv','w')
f.write("CHANNEL WIDTH, ELAPSED TIME\n")
cursor = arcpy.da.SearchCursor("time_1500",["SHAPE@LENGTH","TIME"])
for row in cursor:
    x = row[0]
    y = row[1]
    s = str(x)+','+str(y)+'\n'
    f.write(s)
f.close()
del cursor
```

Figure 3. Screenshot of ‘Select by Attribute’ ArcMap tool to initiate step to generate channel width for a selected time step.

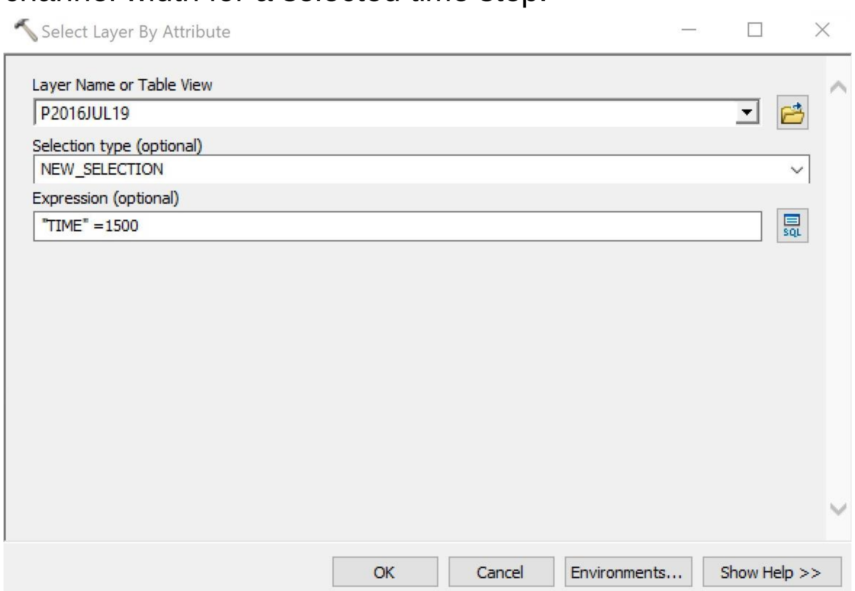


Figure 4. Screenshot of 'Intersect Analysis' Arcmap tool.

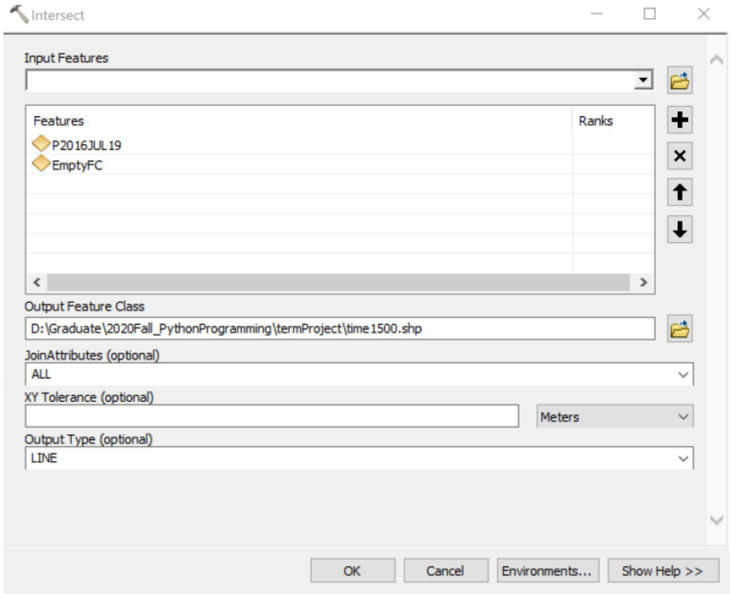


Figure 5. Output of Intersect Analysis tool. Output name: time_1500.

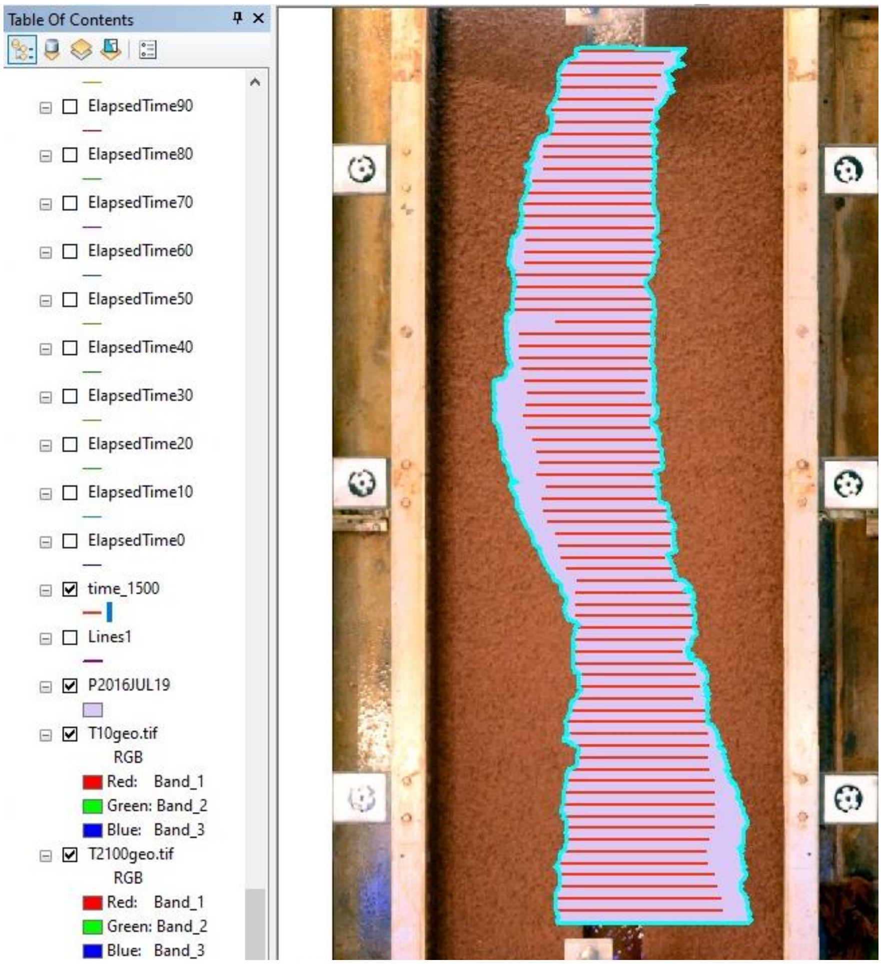
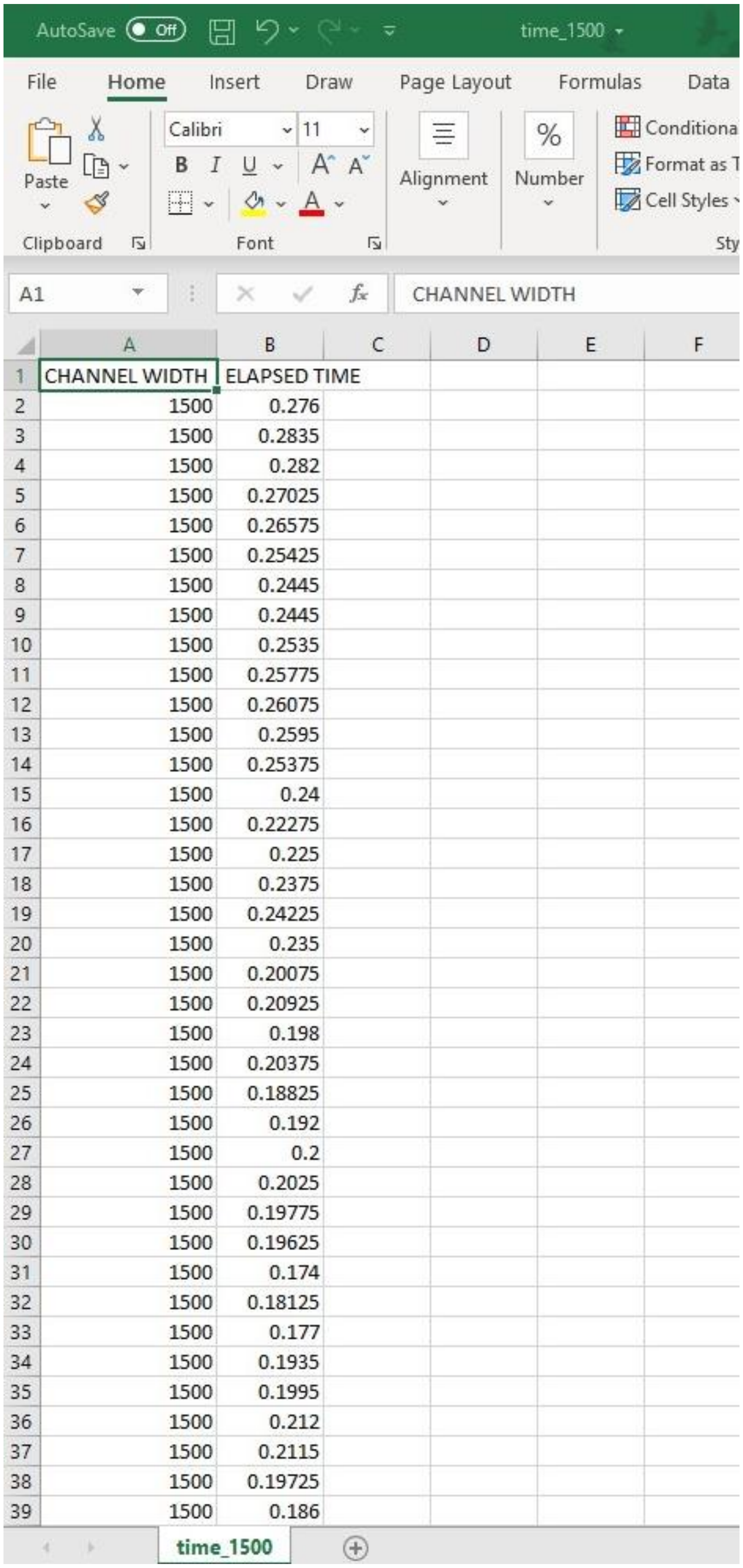


Figure 6. Screenshot of CSV file, result from python script code to generate channel width over time for a selected time step = 1500.

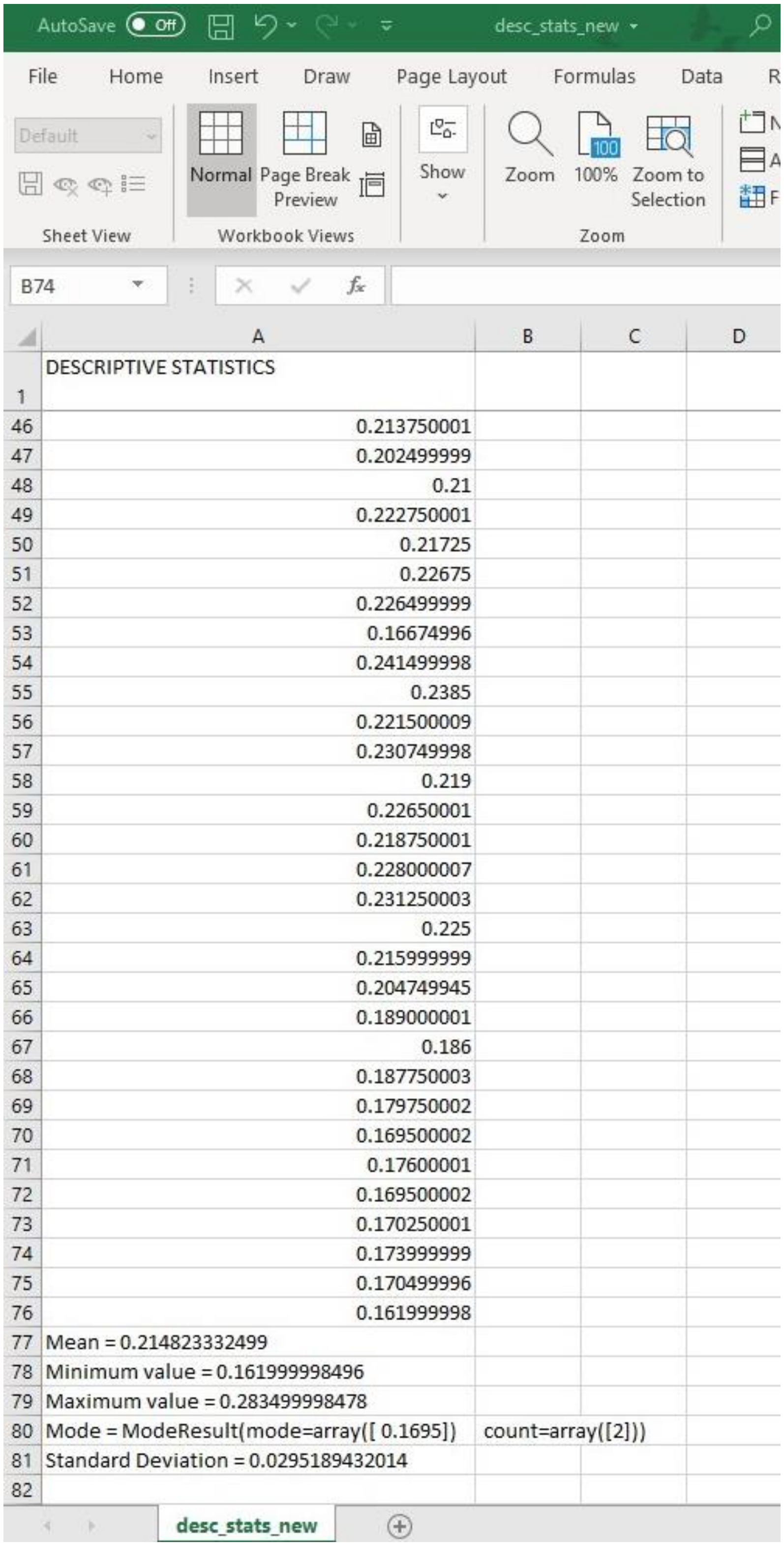


CHANNEL WIDTH	ELAPSED TIME
1500	0.276
1500	0.2835
1500	0.282
1500	0.27025
1500	0.26575
1500	0.25425
1500	0.2445
1500	0.2445
1500	0.2535
1500	0.25775
1500	0.26075
1500	0.2595
1500	0.25375
1500	0.24
1500	0.22275
1500	0.225
1500	0.2375
1500	0.24225
1500	0.235
1500	0.20075
1500	0.20925
1500	0.198
1500	0.20375
1500	0.18825
1500	0.192
1500	0.2
1500	0.2025
1500	0.19775
1500	0.19625
1500	0.174
1500	0.18125
1500	0.177
1500	0.1935
1500	0.1995
1500	0.212
1500	0.2115
1500	0.19725
1500	0.186

Calculating descriptive statistics for a selected time step

```
import numpy as np
import csv
from scipy import stats
J = []
f1 = open(r'D:\Graduate\2020Fall_PythonProgramming\termProject\time_1500.csv', 'r')
c = 0
for j in f1:
    if (c>0):
        jStr = j.strip()
        jLst = jStr.split(',')
        J.append(float(jLst[1]))
    c = c + 1
Mean = np.mean(J)
Maximum = max(J)
Minimum = min(J)
Mode = stats.mode(J)
StandardDeviation = np.std(J)
J.insert(0,'DESCRIPTIVE STATISTICS\n')
J.append("Mean = " + str(Mean))
J.append("Minimum value = " + str(Minimum))
J.append("Maximum value = " + str(Maximum))
J.append("Mode = " + str(Mode))
J.append("Standard Deviation = " + str(StandardDeviation))

File =
open(r'D:\Graduate\2020Fall_PythonProgramming\termProject\desc_stats_new.csv','w')
read = csv.writer(File, delimiter = '\n')
read.writerow(J)
f1.close()
```

Figure 7. Screenshot csv created with descriptive statistics information for a selected time step. Result from python script code to calculate statistics with mean, mode, minimum value, maximum value, and standard deviation.



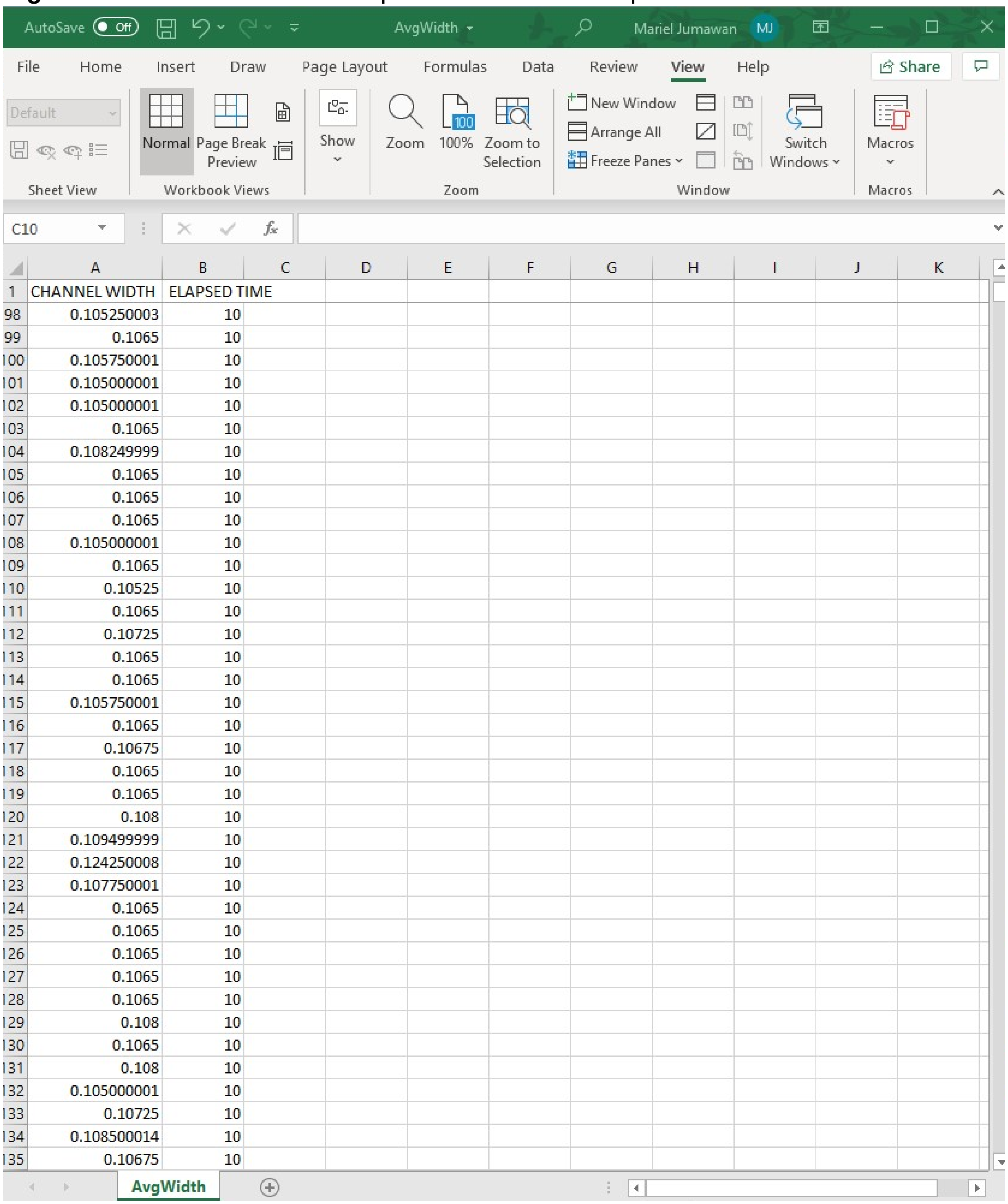
	A	B	C	D
1	DESCRIPTIVE STATISTICS			
46	0.213750001			
47	0.202499999			
48	0.21			
49	0.222750001			
50	0.21725			
51	0.22675			
52	0.226499999			
53	0.16674996			
54	0.241499998			
55	0.2385			
56	0.221500009			
57	0.230749998			
58	0.219			
59	0.22650001			
60	0.218750001			
61	0.228000007			
62	0.231250003			
63	0.225			
64	0.215999999			
65	0.204749945			
66	0.189000001			
67	0.186			
68	0.187750003			
69	0.179750002			
70	0.169500002			
71	0.17600001			
72	0.169500002			
73	0.170250001			
74	0.173999999			
75	0.170499996			
76	0.161999998			
77	Mean = 0.214823332499			
78	Minimum value = 0.161999998496			
79	Maximum value = 0.283499998478			
80	Mode = ModeResult(mode=array([0.1695])	count=array([2]))		
81	Standard Deviation = 0.0295189432014			
82				

Generating average channel width for all time steps

```
import numpy as np
M = open(
r'D:\Graduate\2020Fall_PythonProgramming\termProject\New_CH\Mean_CWD.csv','w')
M.write("Elapsed Time, Mean\n")
L = np.loadtxt('AvgWidth.csv', delimiter = ',', skiprows = 1)
ELT = L[:,1]
CWD = L[:,0]
for row in range(0,2110,10):
    N = []
    for i in range(len(CWD)):
        if ELT[i] == row:
            N.append(CWD[i])
    mean = np.mean(N)
    n = str(row) + ',' + str(mean) + '\n'
    M.write(n)
M.close()

# plotting the graph
import numpy as np
import matplotlib.pyplot as plt

F =
np.loadtxt(r'D:\Graduate\2020Fall_PythonProgramming\termProject\New_CH\Mean_C
WD.csv', delimiter = ',', skiprows = 1)

X = F[:,0]
Y = F[:,1]

plt.plot(X, Y, 'b', label = "2016-07-19 Data")
plt.xlabel("Elapsed Time, in seconds", fontsize = 14)
plt.ylabel("Average Channel Width, in meters", fontsize = 14)
plt.axis([0,2000, 0.0, 0.5])
##plt.show()
Xestimate = np.asarray(range(0,5000,10), dtype = np.float)
model = np.polyfit(X, Y, 2)
Yestimate = np.polyval(model,Xestimate)
plt.plot(Xestimate, Yestimate, 'g--', label =  "Regression Line = 1")
# adding legend and gridlines
plt.grid(True)
plt.legend(loc= 'upper right', fontsize = 12)
##plt.show()
```

```
fig = plt.figure(1, figsize=(10,5), dpi = 250)
fig.savefig("Mariel_plot3.png")
plt.close(fig)
```

Figure 8. Screenshot of one shapefile out of 2110 shapefiles.



	A	B
1	CHANNEL WIDTH	ELAPSED TIME
98	0.105250003	10
99	0.1065	10
100	0.105750001	10
101	0.105000001	10
102	0.105000001	10
103	0.1065	10
104	0.108249999	10
105	0.1065	10
106	0.1065	10
107	0.1065	10
108	0.105000001	10
109	0.1065	10
110	0.10525	10
111	0.1065	10
112	0.10725	10
113	0.1065	10
114	0.1065	10
115	0.105750001	10
116	0.1065	10
117	0.10675	10
118	0.1065	10
119	0.1065	10
120	0.108	10
121	0.109499999	10
122	0.124250008	10
123	0.107750001	10
124	0.1065	10
125	0.1065	10
126	0.1065	10
127	0.1065	10
128	0.1065	10
129	0.108	10
130	0.1065	10
131	0.108	10
132	0.105000001	10
133	0.10725	10
134	0.108500014	10
135	0.10675	10

Figure 9. Screenshot of one shapefile out of 2110 shapefiles.

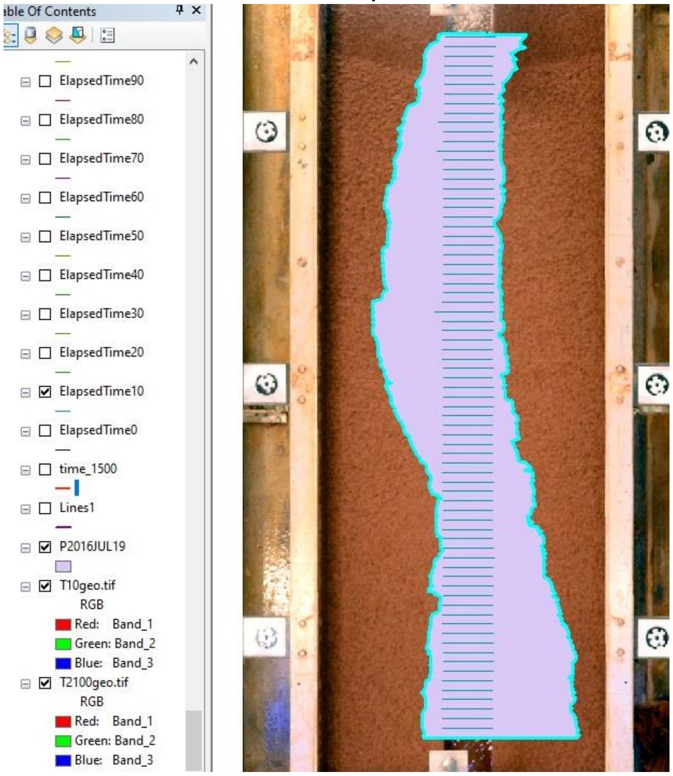
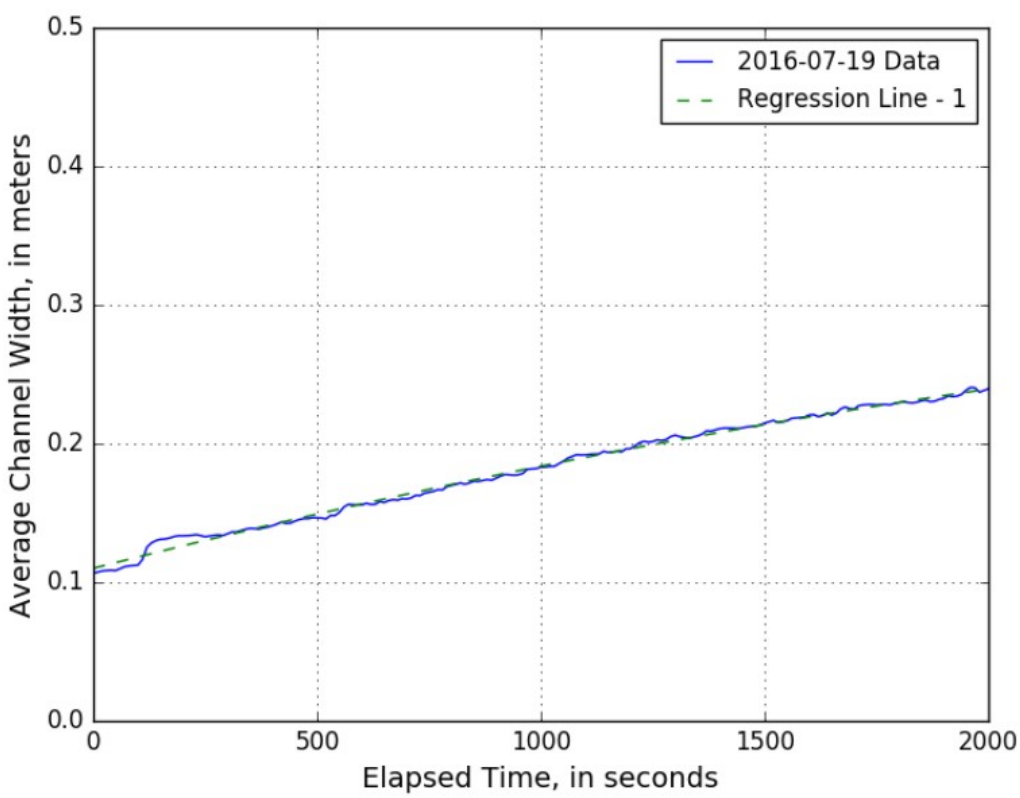


Figure 10. Image of figure created using plot feature on python.



CONCLUSION

The final figure created in this report shows a strong positive linear relationship in the 2016-07-19 data between the average channel width and the elapsed time. This shows that as time increases so does the average channel width. Furthermore, this report really allows the statistician to fully immerse themselves in the application of GIS and python programming in scientific literature.