

Project Report for Guided Project 2: Google Earth Engine for Elevation Extraction

1. Introduction

Project Overview:

The aim of this project is to automate the extraction of elevation data for specific geographical points using Google Earth Engine (GEE). These points are defined by their coordinates in a CSV file, derived from a georeferenced raster file, flood_2class.tif. The task is to leverage GEE's USGS 3DEP dataset, extract elevation data at these coordinates, and save the results into a shapefile format. In addition, the project includes developing an ArcGIS Python Toolbox to simplify the use of this process within ArcGIS Pro, enabling users to execute the tool with minimal interaction with the command line.

Key Objectives:

1. Convert a CSV file containing X and Y coordinates into a point shapefile.
 2. Extract elevation data from Google Earth Engine using the USGS 3DEP 10m elevation dataset.
 3. Write the elevation data into the shapefile.
 4. Create a user-friendly ArcGIS Python Toolbox for use within ArcGIS Pro.
 5. Visualize the results within ArcGIS Pro, adjust symbology, and create a report-ready layout.
-

2. Data Format and Description

The input data consists of a CSV file with the following columns:

- **row**: Row index of the point in the water boundary grid.
- **column**: Column index of the point in the water boundary grid.
- **X**: The longitude (X-coordinate) of the point.
- **Y**: The latitude (Y-coordinate) of the point.

Each coordinate pair (X, Y) represents a geographical location from which elevation data will be extracted using the USGS 3DEP 10m DEM (Digital Elevation Model) available on Google Earth Engine.

Google Earth Engine Dataset:

- **Dataset**: USGS 3DEP 10m National Map Seamless Elevation Data
- **Purpose**: Provides elevation values at a 10m resolution across the United States.

The project utilizes Google Earth Engine to access this dataset and extract elevation values at the given coordinates in the CSV file.

3. Code Explanation and Walkthrough

This section explains the logic behind the code and provides a step-by-step breakdown.

1. Google Earth Engine Authentication:

Before using Google Earth Engine (GEE), the script ensures that the Earth Engine API is authenticated and initialized. If it is not already authenticated, it prompts the user to authenticate.

```
python
Copy code
def main():
    import sys

    try:
        ee.Initialize(project='ee-shmirheidarian')
    except:
        ee.Authenticate()
        ee.Initialize(project='ee-shmirheidarian')
```

2. Reading the CSV and Extracting Coordinates:

The CSV file, containing the coordinates of the points, is read into a pandas DataFrame. For each row, a Point geometry is created using the X and Y coordinates.

```
python
Copy code
def getGeeElevation(workspace, csv_file, outfc_name, epsg=4326):
    csv_file = os.path.join(workspace, csv_file)
    data = pd.read_csv(csv_file)
    geometry = [ee.Geometry.Point([x, y], f'EPSG:{epsg}')] for x, y in
zip(data['X'], data['Y'])
    fc = ee.FeatureCollection(geometry)
```

3. Extracting Elevation Data from Google Earth Engine:

The script then queries the USGS 3DEP DEM dataset on Google Earth Engine and extracts elevation data for each point using the sampleRegions function. The elevation values are then attached to each point's properties.

```
python
Copy code
dem = ee.Image('USGS/3DEP/10m')
sampled_fc = dem.sampleRegions(collection=fc, scale=10, geometries=True)
sampled_info = sampled_fc.getInfo()
```

```
for ind, item in enumerate(fc.getInfo()['features']):
    item['properties'] = sampled_info['features'][ind]['properties']
```

4. Writing to Shapefile:

Next, the script creates a new shapefile to store the points and their corresponding elevation values. The elevation values are written to a new field, elevation.

```
python
Copy code
fname = os.path.join(workspace, outfc_name)

if arcpy.Exists(fname):
    arcpy.management.Delete(fname)

arcpy.management.CreateFeatureclass(workspace, outfc_name,
geometry_type='POINT', spatial_reference=epsg)
arcpy.management.AddField(fname, field_name='elevation', field_type='FLOAT')

with arcpy.da.InsertCursor(fname, ['SHAPE@', 'elevation']) as cursor:
    for feat in fc.getInfo()['features']:
        coords = feat['geometry']['coordinates']
        pnt = arcpy.PointGeometry(arcpy.Point(coords[0], coords[1]),
spatial_reference=epsg)
        elev = feat['properties']['elevation']
        cursor.insertRow((pnt, elev))
```

- The script creates a point shapefile with the EPSG code defined in the input arguments.
- The elevation field is added to the shapefile to store the elevation data.
- The InsertCursor is used to write the points along with their elevation values.

4. ArcGIS Toolbox Integration

To make the script accessible within ArcGIS Pro, a Python Toolbox is created. The toolbox allows users to interact with the script via a graphical user interface (GUI) within ArcGIS Pro.

- **User Inputs:**
 - Input CSV file: User can select a CSV file containing the coordinates.
 - Output Shapefile: User specifies the name for the output shapefile.
 - Spatial Reference (EPSG code): User can define the EPSG code for the projection of the shapefile.
- **Toolbox Design:** The toolbox provides a simple interface for users to run the elevation extraction process without needing to manually edit the script or provide command-line arguments.

5. Data Visualization and Layout in ArcGIS Pro

Once the shapefile is created with the elevation data, users can visualize it in ArcGIS Pro. The following steps guide users on how to visualize the points and modify symbology:

1. Add the Shapefile to ArcGIS Pro:

- Open ArcGIS Pro and create a new map or use an existing map.
- Add the output shapefile to the map.

2. Modify Symbology:

- Right-click on the shapefile layer in the table of contents and select "Symbology".
- Use a color gradient or classification based on the elevation field to represent the elevation values visually.

3. Create Layout for Printing:

- Add a new layout in ArcGIS Pro.
- Insert elements such as the title, legend, north arrow, and scale bar.
- Add a map frame to the layout and display the map with the symbology applied.

4. Export Layout to PDF:

- Once the layout is complete, export it to a PDF to share or print the map.

6. GitHub Repository

The following link is the GitHub repository where the project files, code, and documentation are hosted:

- **GitHub** **Repository:**
https://github.com/shabnamm7/geog4057_shabnam/tree/main/FinalProject/Project2

The repository contains the following:

- Python scripts for elevation extraction using Google Earth Engine.
 - A sample CSV file with X, Y coordinates.
 - Instructions on how to run the code.
 - A Python Toolbox for ArcGIS Pro to simplify the process.
 - Documentation including this report and setup instructions.
-

7. Conclusion

Summary of Work:

This project successfully implements a Python script for extracting elevation data from Google Earth Engine using coordinates from a CSV file. The data is written into a shapefile, allowing for easy visualization and analysis in ArcGIS Pro. The project also includes a Python Toolbox that facilitates the use of the script within ArcGIS Pro, making it accessible to users with minimal technical expertise.

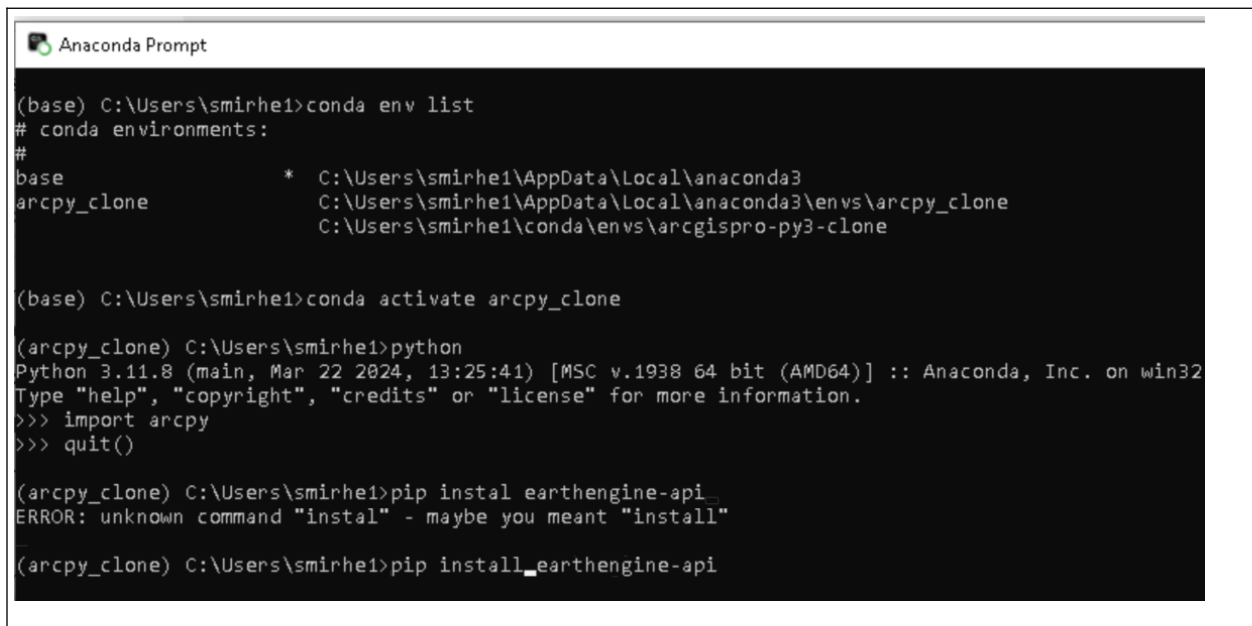
Instructions for Use:

1. Ensure the Google Earth Engine API is installed and authenticated.
2. Download or clone the repository containing the project files.
3. Open ArcGIS Pro and use the provided Python Toolbox to run the tool.
4. Provide the required input CSV file, output shapefile name, and EPSG code.
5. Visualize the output in ArcGIS Pro, modify symbology, and generate a layout for reporting.

Future Enhancements:

- Add more advanced error handling and validation for the CSV file format and data consistency.
- Incorporate additional environmental datasets from Google Earth Engine for broader analysis, such as land cover or vegetation data.
- Enable multi-threading or batch processing to handle large datasets more efficiently.

Screenshots of the project:



```
Anaconda Prompt

(base) C:\Users\smirhe1>conda env list
# conda environments:
#
base                  *  C:\Users\smirhe1\AppData\Local\anaconda3
arcpy_clone            C:\Users\smirhe1\AppData\Local\anaconda3\envs\arcpy_clone
                       C:\Users\smirhe1\conda\envs\arcgispro-py3-clone

(base) C:\Users\smirhe1>conda activate arcpy_clone

(arcpy_clone) C:\Users\smirhe1>python
Python 3.11.8 (main, Mar 22 2024, 13:25:41) [MSC v.1938 64 bit (AMD64)] :: Anaconda, Inc. on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> import arcpy
>>> quit()

(arcpy_clone) C:\Users\smirhe1>pip instal earthengine-api_
ERROR: unknown command "instal" - maybe you meant "install"

(arcpy_clone) C:\Users\smirhe1>pip install_earthengine-api
```

```
Anaconda Prompt - python

(base) C:\Users\smirhe1>conda activate arcpy_clone

(arcpy_clone) C:\Users\smirhe1>import ee
'import' is not recognized as an internal or external command,
operable program or batch file.

(arcpy_clone) C:\Users\smirhe1>python
Python 3.11.8 (main, Mar 22 2024, 13:25:41) [MSC v.1938 64 bit (AMD64)] :: Anaconda, Inc. on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> import ee
>>> ee.Authenticate()
To authorize access needed by Earth Engine, open the following URL in a web browser and follow the instructions. If the
web browser does not start automatically, please manually browse the URL below.

https://accounts.google.com/o/oauth2/auth?client_id=517222506229-vsmmajv00ul0bs7p89v5m89qs8eb9359.apps.googleusercontent.com&scope=https%3A%2F%2Fwww.googleapis.com%2Fauth%2Fearthengine+https%3A%2F%2Fwww.googleapis.com%2Fauth%2Fcloud-platform+https%3A%2F%2Fwww.googleapis.com%2Fauth%2Fdevstorage.full_control&redirect_uri=http%3A%2F%2Flocalhost%3A8085&response_type=code&code_challenge=nXlmPck0qyCA14_MreTzfI9gZiuzznQVbvw3SDdf8rs&code_challenge_method=S256

Waiting for successful authorization from web browser ...

Successfully saved authorization token.
>>> _
```

```
ee.ee_exception.EEException: Not signed up for Earth Engine or project is not registered. Visit https://developers.google.com/earth-engine/guides/access
>>> ee.Initialize(project='ee-smirheidarian')
>>> _
```

