# Buncombe County, North Carolina Flood Risk

Amanda Riggs

October 26, 2024

GEOG-6460

East Carolina University

Software used in this project includes ArcGIS Pro (Esri), Python (with ArcPy), and NumPy. Data sources include publicly available elevation, hydrography, and land use datasets.

This document and associated materials are shared for educational and portfolio purposes only.

# Introduction

Hydrological analysis in Geographic Information Systems (GIS) uses Digital Elevation Models (DEMs) to understand how water flows across terrain. Such techniques help model real-world water systems by simulating flow paths, watersheds, and stream networks in flood-prone areas. The ability to perform hydrological analysis is especially important in high-relief areas like Buncombe County, North Carolina, which experienced significant flooding from rainfall during Hurricane Helene in September of 2024. This project's objective is to use raw DEM data to perform a hydrological analysis and create a flood risk map using an automated workflow created through Python scripting in ArcGIS Pro to streamline the process.

## Study Area

Buncombe County, North Carolina (NC), is situated in the mountainous region of Western North Carolina and covers approximately 1,709 square kilometers (1,709,000 meters (m)). It includes parts of the Blue Ridge Mountains, and a section of the Eastern Continental Divide passes through it. Its landscape includes natural features like the French Broad River (Mazzocchi, 2006). Because Buncombe County is in a mountainous region, it has various elevations. According to Topographic Map (n.d.), the county's average elevation is 2,753 feet (ft) (839.1144 m), the minimum elevation is 846 ft (257.861 m), and the maximum elevation is 6,578 ft (2004.974 m). On September 27, 2024, Hurricane Helene's destructive rains that followed two days of saturating rain produced catastrophic flooding that claimed towns and lives. Ashville, North Carolina's (location shown in Figure 8) regional airport recorded almost 14 inches of rain before it stopped working on September 27, leading to large volumes of runoff that contributed

to record heights of rivers, with the French Broad River in Asheville reaching 24.7 ft (7.52856 m), a foot higher than its previous record (26.1 ft) (NASA Earth Observatory, 2024).

## Data

I acquired the raw DEM data for this project from NC OneMap who cited the data from the NC. Center for Geographic Information and Analysis, NC Department of Public Safety (2019). The Floodplain Mapping Program derived the DEM data from LiDAR, which the NC Floodplain Mapping Program and the Harris Corporation (Harris) collected. Then, the NC Department of Public Safety – Division of Emergency Management processed the DEM. The DEM is specific to the 2017 Phase Five project that consists of 21 NC counties. Harris collected both the LiDAR data, and the validation site collected by a single aerial vendor on the ESP team. The sensors used to collect the LiDAR data were three GmAPD (Geiger-mode_ sensors) set at a 0.35-meter nominal post spacing. The LiDAR data was collected for Buncombe County between February and April of 2017 during leaf-off conditions, and the ground survey for the project included a GPS base station and a collection of control points for calibration purposes. All the data is in tiles based on a 2,500-ft by 2,500-ft tilting scheme and in the LASer (LAS) version of 1.4 standard format.

NC. Center for Geographic Information and Analysis, NC Department of Public Safety (2019) wrote that the accuracy was required to be 18.13 cm for Non-Vegetated Accuracy (NVA) at a 95% confidence level and 30 cm for Vegetated Vertical Accuracy (VVAS) at the 95th percentile based on RMSE (z) of 9.25 cm in the bare earth and urban land cover classes. Testing of the NVA consisted of 1464 bare earth and urban checkpoints. In contrast, the testing of VVAS used 1735 checkpoints in various vegetation types distributed throughout the project area and the GPS techniques used for the survey. Independent contractors of the North Carolina Geodetic Survey

Comparison provided elevations and lidar surface to the elevation values of the surveyed control points for comparison. Calculation of the NVA of the DEM files used Triangulated Irregular Networks (TIN) derived from the final calibrated and controlled data that used the 1464 checkpoints I mentioned earlier. Accuracy Z was evaluated at 0.157 m to meet 18.13 cm or better non-vegetated vertical accuracy (NVVA) at 95% confidence using RMSE (z) x 1.9600, and to meet 30 cm or better vegetated vertical accuracy (VVA) was computed at a 94thpercentile. Also, the calculation of the VVA evaluated at 0.228 meters used the 1735 checkpoints located in various vegetation types that I mentioned earlier, at the 95th percentile. It was compared with and calculated by surveyed elevations also provided by independent North Carolina Geodetic Survey contractors.

The Buncombe County, NC DEM I downloaded for analysis was hydro-flattened using ground LiDAR points in conjunction with hydro brakes to create a 3.125-ft hydro-flattened Raster DEM (NC Center for Geographic Information and Analysis, NC Department of Public Safety, 2019). A Tagged Image File Format (TIFF) file was created and reviewed for each tile using ESP Analyst to check for any surface anomalies or incorrect elevations. Since the DEM I downloaded is already hydro-flattened, it will save much time during analysis. Water bodies must be hydro-flattened not only for aesthetic purposes to enhance visualization and interpretation but also for the representation of accurate water flow to ensure water bodies show consistent flow within flat surfaces. Hydro-flattening also protects against errors in flow direction and accumulation calculations to increase reliable watershed delineation. Hydro-flattening also helps to keep consistent elevation data across water bodies, which may show unrealistic elevations and skew analysis.

The NC Center for Geographic Information and Analysis, NC Department of Public Safety (2019) states that the DEM has a 3.125-ft resolution with a grid cell size of three ft, which is a new development, taking the DEMs from a 20-ft cell size to a higher resolution of three ft. A three-ft cell size has a much better resolution than a larger cell size, creating more detailed terrain features, but the tradeoff is that a smaller cell size covers a much smaller area. Larger cell sizes are great for covering larger areas, but the tradeoff is less detailed terrain information. The projected Coordinate System is NAD_1983_2011_StatePlane_North_Carolina_FIPS_3200_Ft_US with a horizontal datum of NAD83 (2011) and a vertical datum of NAVD88 (Geoid 12B) in the US survey feet. The Projection is Lambert_Conformal_Conic. The Geographic Coordinate System is GCS_NAD_1983_2011. The WKID is 6318, and the authority is EPSG. The Datum the DEM uses are D_NAD_1983_2011. I had to project the DEM into NAD_1983_2011_StatePlane_North_Carolina_FIPS_3200_(meters) and use the Raster Calculator tool to convert the DEM's cell units into meters rather than US Ft. It is essential to convert a DEM used for analysis into meters because meters are a standard unit of measurement in most scientific fields. The design of GIS and hydrological modeling tools frequently collaborates with units in meters; therefore, to ensure compatibility, it is best practice to convert DEM units into meters. Also, the use of the metric system is global, and international projects use meters to share and collaborate for varied reasons.

Other data includes state and boundary shapefiles to use for clipping and location mapping purposes from TIGER/Line Shapefiles, developed by the US Census Bureau, and the Watershed Boundary Dataset (WBD) in the early 1990s using various methods based on state. I acquired the dataset through NC OneMap. The US Census Bureau (2022) sourced the shapefile from the

USDA Geospatial Data Gateway, initially created by the US Geological Survey. The dataset underwent detailed quality reviews to ensure accuracy and compliance with federal standards for delineating hydrologic unit boundaries (Census Bureau, 2022). The dataset is delivered at a scale of 1:24,000 and includes attributes for hydrologic unit codes and areas in acres.

For validation, I included FEMA's 2024 flood insurance hazard area map that identifies special flood hazard areas as part of the National Flood Insurance Program. The data's original coordinate system is a Web Mercator Auxiliary Sphere with a scale limited to 1:1,000,000 and larger, and its resolution is 0.0001 m. The layer was derived from a June 277, 2024, extract of the National Flood Hazard Layer feature class with the original extent consisting of Contiguous United States, Alaska, Hawaii, Puerto Rico, Guam, US Virgin Islands, Northern Marianas, Islands, and American Samoa.

# Methods

To create an automated workflow for the hydrological analysis of Buncombe County, I initially attempted to construct a model in ArcPro GIS ModelBuilder. However, the extensive processing led me to develop a Python script utilizing ArcPy, a site package for ArcGIS Pro, with permission to utilize artificial intelligence (AI) to automate the workflow and expedite processing times. AI is a useful tool for learning the art of programming by helping break down error codes and explain each step in detail. Each tool utilized in the following scripts is available in the ArcGIS Pro software, which I referenced all from ESRI's various tool documentation. I provided a hyperlink within each tool referenced throughout the method section for easy access to each tool's documentation. I also referred to ESRI's (n.d.) hydrologic analysis and surface tool sample workflows to ensure the correct sequence of hydrologic tools, Esri's (n.d.) surface

analysis examples, as well as Esri's (2020) tutorial on how to create a threshold raster to be used as an input for the spatial analyst hydrology tools to be sure I generated rasters that would enhance the quality of the final flood risk map. With the help of OpenAI (2024), together we created Python scripts for semi-automation in separate sections to break up the workflow into manageable parts.

## Raster Preparation

The first function is responsible for preparing the DEM for analysis by setting the workspace, scratch workspace, projecting the spatial coordinate system, filling the raw DEM, and clipping the DEM to boundary extent. The script begins by using the *Project Raster* and *Raster Calculator* tools. As seen in Figure 1, the beginning script projects the input raster to the NAD83 UTM Zone 20N coordinate system and saves it. Next, the raster calculator multiplies the raster values by 0.3048 to convert from feet to meters, and the *Fill* tool is applied to remove depressions within the raster. Then the script utilizes the *Clip Raster* tool to clip the DEM to the study area boundary using boundary shapefile of the county.

```python
import arcpy

import os


def chunk1(input_raster, clip_feature, output_folder):
    print("Starting Chunk 1 processing...")
    projected_raster = os.path.join(output_folder, 'projected_raster.tif')
    out_coor_system = arcpy.SpatialReference(26920)

    # Delete existing output if it exists
    delete_if_exists(projected_raster)

    print(f"Projecting raster to {projected_raster}...")
    arcpy.management.ProjectRaster(input_raster, projected_raster, out_coor_system)

    # Project the clip feature
    projected_clip_feature = os.path.join(output_folder, 'projected_clip.shp')
    delete_if_exists(projected_clip_feature)

    print(f"Projecting clip feature to {projected_clip_feature}...")
    arcpy.management.Project(clip_feature, projected_clip_feature, out_coor_system)

    calculated_raster = os.path.join(output_folder, 'calculated_raster.tif')
    projected_raster_obj = arcpy.Raster(projected_raster)
    calculated_raster_result = projected_raster_obj * 0.3048

    # Delete existing output if it exists
    delete_if_exists(calculated_raster)

    print(f"Saving calculated raster to {calculated_raster}...")
    calculated_raster_result.save(calculated_raster)

    filled_raster = os.path.join(output_folder, 'filled_raster.tif')

    # Delete existing output if it exists
    delete_if_exists(filled_raster)

    print(f"Filling raster and saving to {filled_raster}...")
    filled_raster_result = arcpy.sa.Fill(calculated_raster)
    filled_raster_result.save(filled_raster)

    # Clip the filled raster using the projected clip feature
    clipped_raster = clip_raster(filled_raster, projected_clip_feature, output_folder)

    print("Chunk 1 processing complete!")
    return clipped_raster
```

*Figure 1. Raster preparation including spatial coordinate projection, unit conversion, depression filling, and boundary clipping.*

## Hydrological Analysis

The next three sections of the script work on hydrological analysis. It begins by computing the flow direction for each cell in the filled raster, generating a flow direction raster using the *Flow Direction* tool and then the *Flow Accumulation* tool to create a raster that calculates the accumulated flow to each cell as seen in Figure 2. The next section of the script is shown in Figure 3, which generates a conditional raster representing the stream network using the *Con* tool that identifies the cell with the maximum flow accumulation value and then converts. The threshold is the most important parameter during this part of the workflow. To find the threshold to delineate the cells with the highest flow, I classified the flow accumulation raster into five classes. I used the fourth-class end value of (420997289) to use in the con tool because after trying various thresholds, the fourth class output the best result. According to Esri (2020), I should be able to use the *Stream Link* tool to create a pour point raster. However, creating a pour point raster part of the workflow needed to utilize other tools because the watershed tool kept failing when using the output from the stream link tool. Therefore, I added the *Raster to Point* tool and used the flow accumulation output to create a pour point feature layer so I could append the specific values to each point generated in the pour point feature layer. I preferred to include the *Snap Pour Point* tool to ensure the precision of the pour points and output a snapped pour point shapefile. However, due to time constraints, I had to remove the snap pour point tool from the workflow because of the long processing times that led the script's to repeated failures. To append the elevation values to the generated pour point feature, I used the *Extract Values to Points* tool to extract the cell values of the flow accumulation raster to the pour point feature layer. Finally, as seen in Figure 4, I used the pour point feature with the extracted values as a parameter input to the *Watershed* tool, which uses the outputs from the flow direction output and the pour point feature outputs to delineate watersheds based on the flow direction and the pour

point values, which then outputs a watershed raster. Next, I included the *Raster to Polygon* tool to convert the watersheds into a shapefile to use for future analysis. I also included two stream delineation tools. The con tool, in combination with the flow accumulation output, generates a stream network raster by applying a threshold of 2000 to extract information about drainage patterns, such as upstream and downstream connectivity. I chose this threshold based on trial and error. The higher the threshold resulted in higher detail. I also used the *Stream Order* tool to assign a numeric order to the branches of linear streams. According to Esri (n.d.), the default Strahler method assigns all links without tributaries in an order of 1 (first order), which is the most common stream ordering method. However, because this method only increases in order at intersections of the same order, it does not account for all links, and the method is sensitive to the addition or removal of links. I would have liked to use the Shreve method because it accounts for all links in the network; however, when attempting to utilize the Shreve method, processing times significantly slowed, hindering my ability to complete the project in a timely manner.

```python
import arcpy
import os

def chunk2(filled_raster, output_folder):
    print("Starting Chunk 2 processing...")
    flow_direction = os.path.join(output_folder, 'flow_direction.tif')

    # Delete existing output if it exists
    delete_if_exists(flow_direction)

    print(f"Calculating flow direction and saving to {flow_direction}...")
    flow_direction_result = arcpy.sa.FlowDirection(filled_raster)
    flow_direction_result.save(flow_direction)

    flow_accumulation = os.path.join(output_folder, 'flow_accumulation.tif')

    # Delete existing output if it exists
    delete_if_exists(flow_accumulation)

    print(f"Calculating flow accumulation and saving to {flow_accumulation}...")
    flow_accumulation_result = arcpy.sa.FlowAccumulation(flow_direction)
    flow_accumulation_result.save(flow_accumulation)
```

*Figure 2. Python script using the Flow Direction tool and Flow Accumulation tool.*

```python
def main():
    print("Starting the main process...")

    # Enable overwriting of existing datasets
    arcpy.env.overwriteOutput = True

    # Define paths
    output_folder = r'D:\DigiTerrain\DTA_week9\Output'
    clipped_flow_accumulation = r'D:\DigiTerrain\DTA_week9\Output\clipped_flow_accumulation.tif'
    clipped_flow_direction = r'D:\DigiTerrain\DTA_week9\Output\clipped_flow_direction.tif'
    clip_feature = r'D:\DigiTerrain\DTA_week9\Buncombe_Bound.shp'  # Update this path as needed

    # Load the clipped flow accumulation raster
    flow_accumulation_raster = arcpy.Raster(clipped_flow_accumulation)

    # Create pour points raster
    try:
        print("Creating new pour points raster with threshold: 258685081")
        threshold = 420997289  # Adjust threshold if necessary

        # Create the pour points raster
        pour_points_raster = arcpy.sa.Con(flow_accumulation_raster > threshold, 1)
        pour_points_raster_path = os.path.join(output_folder, 'new_pour_points_raster.tif')
        print(f"Saving new pour points raster to: {pour_points_raster_path}")
        pour_points_raster.save(pour_points_raster_path)
        print("Pour points raster created successfully!")
    except Exception as e:
        print(f"Error in pour point calculation: {e}")

    # Convert pour points raster to point feature
    pour_points_feature = os.path.join(output_folder, 'pour_points.shp')
    try:
        print(f"Converting pour points raster to point feature: {pour_points_feature}...")
        arcpy.conversion.RasterToPoint(pour_points_raster_path, pour_points_feature, "VALUE")  # Use "VALUE" instead of "RASTERVALU"
        print("Pour points feature created successfully!")

        # Add an ID field to the pour points feature
        arcpy.management.AddField(pour_points_feature, "ID", "LONG")
        arcpy.management.CalculateField(pour_points_feature, "ID", "!FID!", "PYTHON3")
        print("ID field added and populated successfully!")
    except Exception as e:
        print(f"Error converting pour points raster to feature: {e}")

    # Extract values from flow accumulation raster to pour points feature
    try:
        print(f"Extracting values from flow accumulation raster to pour points feature...")
        arcpy.sa.ExtractValuesToPoints(pour_points_feature, flow_accumulation_raster,
                                       os.path.join(output_folder, 'pour_points_with_values.shp'),
                                       interpolate_values=True)
        print("Values extracted successfully!")
    except Exception as e:
        print(f"Error extracting values: {e}")
```

*Figure 3. Python script using the Con, Raster to Point, Add Field, and Extract Values to Points tools.*

```
pour_point_field = "RASTERVALU"  # Use the correct field name

# Check if pour points feature exists
if not arcpy.Exists(pour_points_feature):
    print(f"Error: Pour points feature does not exist at {pour_points_feature}")
    return
else:
    print(f"Pour points feature exists at {pour_points_feature}")

# Check if the output folder exists
if not os.path.exists(output_folder):
    print(f"Error: Output folder does not exist at {output_folder}")
    return
else:
    print(f"Output folder exists at {output_folder}")

# Watershed delineation
try:
    print("Starting watershed delineation...")
    print(f"Using flow direction raster: {clipped_flow_direction}")
    print(f"Using pour points feature: {pour_points_feature}")

    watershed = arcpy.sa.Watershed(clipped_flow_direction, pour_points_feature, pour_point_field)
    print("Watershed calculation in progress...")  # Add this line for monitoring
    watershed.save(watershed_output)
    print(f"Watersheds delineated successfully and saved to: {watershed_output}")

    # Convert the watershed raster to a shapefile
    print("Converting watershed raster to shapefile...")
    arcpy.conversion.RasterToPolygon(watershed_output, watershed_shapefile, "NO_SIMPLIFY", "Value")
    print(f"Watershed shapefile created successfully: {watershed_shapefile}")

except Exception as e:
    print(f"Error in watershed delineation or conversion: {e}")

# Stream delineation
try:
    print("Starting stream delineation...")
    # Set the threshold for defining streams
    threshold = 2000  # Going with a higher stream to find more detailed streams
    print(f"Stream threshold set to: {threshold}")

    # Create the stream network based on the flow accumulation raster
    print(f"Using flow accumulation raster: {clipped_flow_accumulation}")
    flow_accumulation = arcpy.Raster(clipped_flow_accumulation)
    print("Creating stream network...")
    stream_network = arcpy.sa.Con(flow_accumulation > threshold, 1)
    stream_network.save(stream_network_output)
    print(f"Stream network created successfully and saved to: {stream_network_output}")

    # Calculate stream order
    print("Calculating stream order...")
    stream_order = arcpy.sa.StreamOrder(clipped_flow_direction, stream_network, "STRAHLER")
    stream_order.save(stream_order_output)
    print(f"Stream order calculated successfully and saved to: {stream_order_output}")

except Exception as e:
    print(f"Error in stream delineation: {e}")
```

*Figure 4. Python script using the Watershed, Raster to Polygon, Con, and Stream Order tools.*

## Terrain Analysis

The next section of the python script consists of surface analysis, which calculates terrain attributes from the filled raster using ESRIs (n.d.) *Slope*, *Curvature*, *Aspect*, and *Hillshade* surface tools as shown in Figure 5. Each analysis generates a raster that the script saves in the output folder. The slope tool identifies the steepness from each cell of the raster; the curvature tool calculates the curvature of the raster's surface to collect information about the type and amount of the surface curvature, which helps to describe the physical characteristics of the surface. The aspect tool identifies the compass direction from each cell of the raster's surface, which helps identify flat areas and slopes. The hillshade tool creates a shaded relief from the surface raster by considering the illumination source shadows, which is great for adding depth and dimensions to maps. Finally, the *main* function directs the entire workflow by initializing the workspace, defining the coordinate system, and calling the chunk functions to process the raster data.

```python
import os

def calculate_terrain_attributes(filled_raster, output_folder):
    """Calculate slope, curvature, aspect, and hillshade from a filled raster."""
    # Define output raster paths
    slope_raster = os.path.join(output_folder, 'slope.tif')
    curvature_raster = os.path.join(output_folder, 'curvature.tif')
    aspect_raster = os.path.join(output_folder, 'aspect.tif')
    hillshade_raster = os.path.join(output_folder, 'hillshade.tif')

    # Ensure output folder exists
    if not os.path.exists(output_folder):
        os.makedirs(output_folder)

    # Calculate slope
    try:
        slope_result = arcpy.sa.Slope(filled_raster, "DEGREE")
        slope_result.save(slope_raster)
        print("Slope raster created successfully.")
    except Exception as e:
        print(f"Error calculating slope: {e}")

    # Calculate curvature
    try:
        curvature_result = arcpy.sa.Curvature(filled_raster, 1)
        curvature_result.save(curvature_raster)
        print("Curvature raster created successfully.")
    except Exception as e:
        print(f"Error calculating curvature: {e}")

    # Calculate aspect
    try:
        aspect_result = arcpy.sa.Aspect(filled_raster, method="PLANAR")
        aspect_result.save(aspect_raster)
        print("Aspect raster created successfully.")
    except Exception as e:
        print(f"Error calculating aspect: {e}")

    # Calculate hillshade
    try:
        hillshade_result = arcpy.sa.Hillshade(filled_raster, azimuth=315, altitude=45)
        hillshade_result.save(hillshade_raster)
        print("Hillshade raster created successfully.")
    except Exception as e:
        print(f"Error calculating hillshade: {e}")

    # Check if output rasters were created successfully
    for raster in [slope_raster, curvature_raster, aspect_raster, hillshade_raster]:
        if arcpy.Exists(raster):
            print(f"{raster} exists and was created successfully.")
        else:
            print(f"Warning: {raster} was not created.")

    print("Terrain attributes processing complete!")

def main():
    # Define file paths
    output_folder = r'D:\DigiTerrain\DTA_week9\Output'
    filled_raster = os.path.join(output_folder, 'filled_raster.tif')
```

*Figure 5. Python script using the Slope, Curvature, Aspect, and Hillshade tools.*

## Statistics

The next section of the script calculates statistics using the *Zonal Statistics* tool to summarize the values of the raster within the zones of the other datasets to calculate and display statistical information about the zones in order to compare zones, monitor trends, extract spatial information, and analyze data as seen in Figure 6. For inputs for zonal statistics, I used the watershed, flow accumulation, flow direction, stream network, elevation, slope, and curvature to decide the weights for each layer to include in the *Weighted Overlay* tool. The weighted overlay tool overlays multiple rasters using a common measurement scale and weights each according to its importance. However, the weighted overlay tool requires that all inputs be in integer format; therefore, I had to employ the *Reclassify* tool to do the conversion. Shown in Figure 7, I set the scales for each raster based on the properties retrieved from each raw raster using the *Get Raster Properties* function to normalize all rasters for use in the weighted overlay tool. After using the zonal statistics tool for each output, I joined the statistical outputs in a table to determine the weights to give each raster. I created a separate table that included the mean and standard deviations of the rasters that I wanted to use in the weighted overlay tool. The formulas I used to calculate and normalize the weights referred to the formula for the *coefficient of variation (CV)*, which Wikipedia (2024) defines as the ratio of the standard deviation to the mean. Table 1 depicts the calculated normalized weights. The formulas used for the calculations are as follows:

Weight Calculation:

If the standard deviation is greater than zero:

$$weight = mean/std$$

If the standard deviation is zero:

$$weight = mean$$

Weight Normalization:

Total weight calculation:

$$total\ weight = \sum weights$$

Normalized weight calculations:

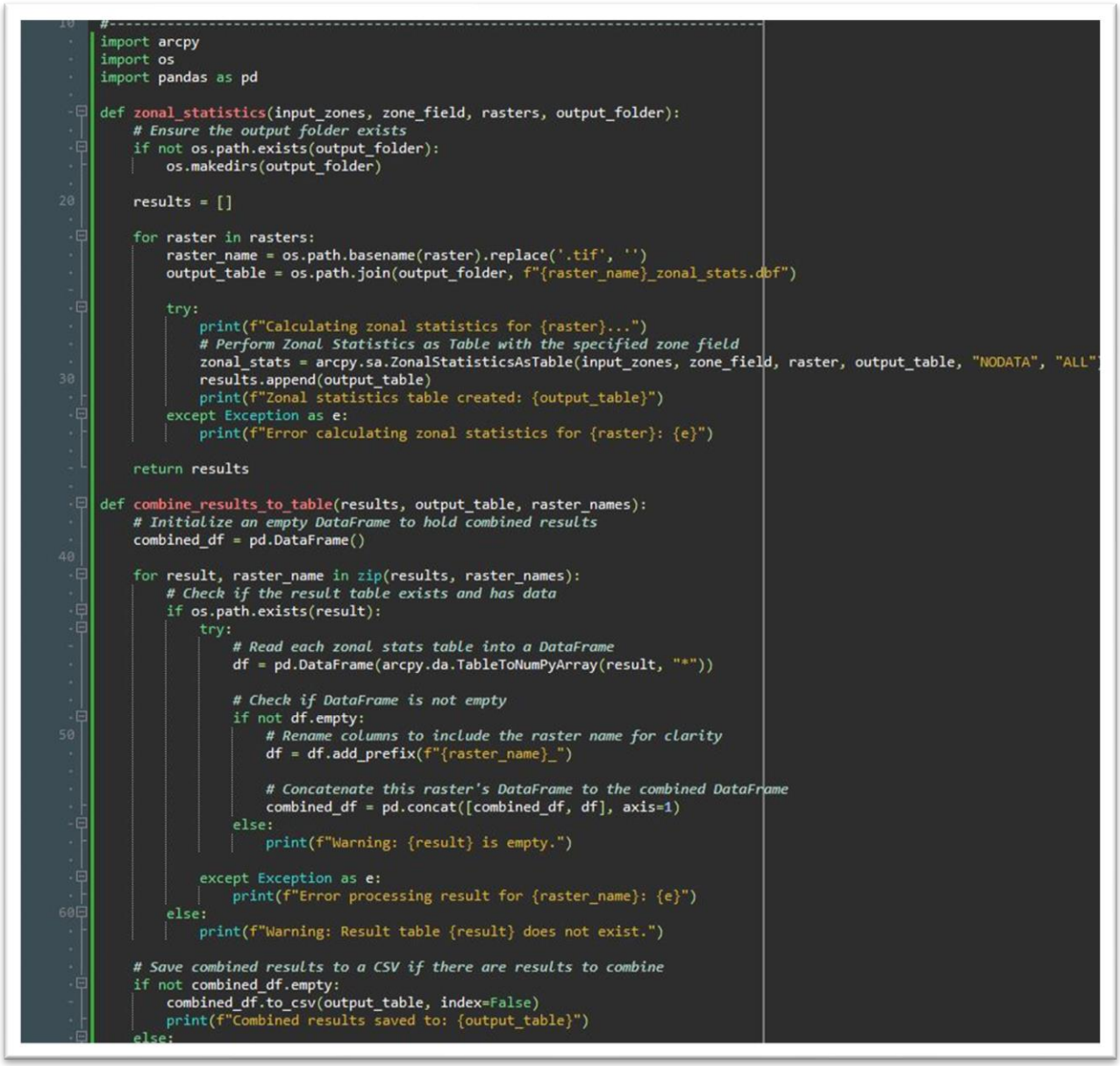
$$Normalized\ weight = weight/total\ weight$$

I was initially concerned about the results of the calculations, but considering the study area is in a mountainous area, it made sense that elevation (filled_raster = 0.95275) would contribute the most weight since water naturally flows downhill, making the lower areas prone to flood hazards, especially during periods of heavy rain. Therefore, I used the exact values derived from the formula for the weight parameters for the final analysis.

| | Metric | Weight |
|---|---|---|
| 1 | filled_raster | 0.95275 |
| 2 | slope | 0.003621 |
| 3 | curvature | -0.000005 |
| 4 | clipped_flow_accumul... | 0.025808 |
| 5 | clipped_flow_direction | 0.0102 |
| 6 | watersheds | 0 |
| 7 | stream_network | 0.003813 |
| 8 | stream_order | 0.003813 |

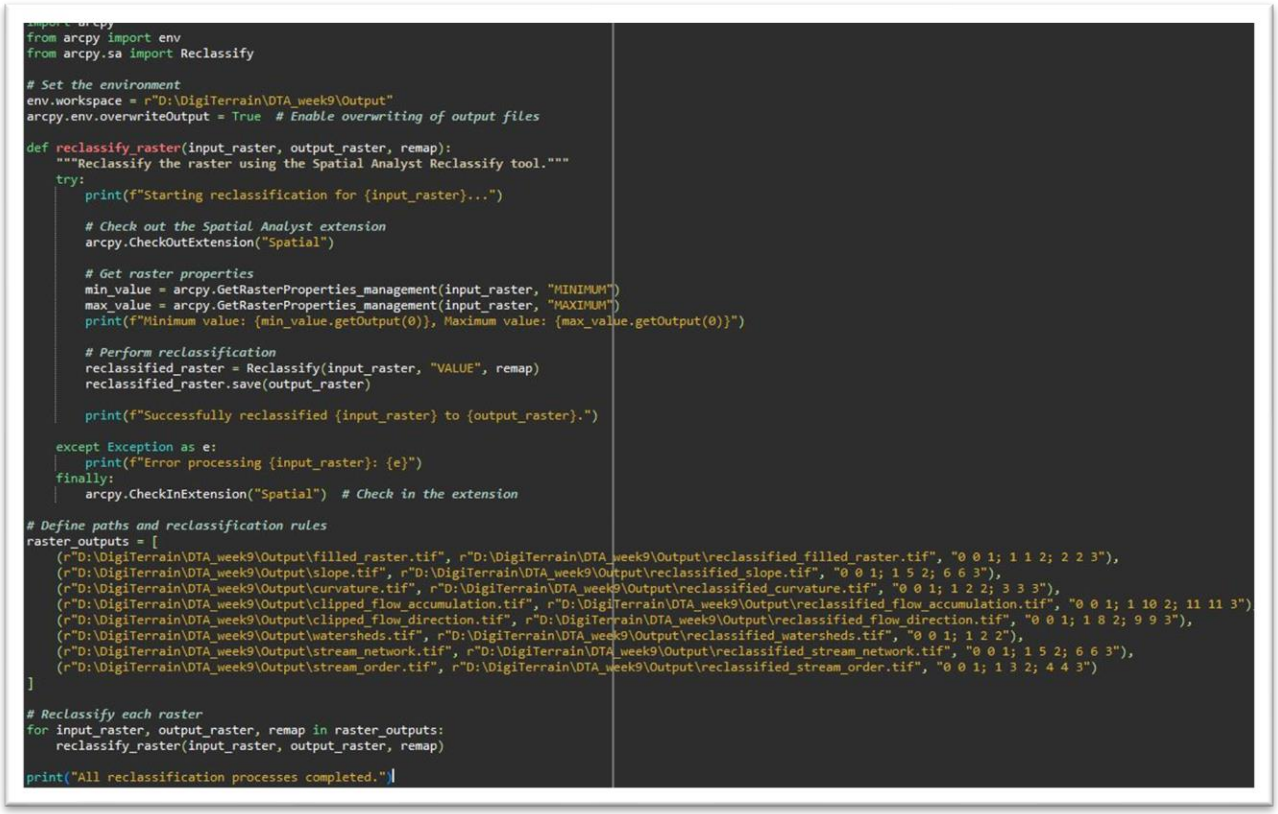
*Table 1 Calculated weights for the elevation, slope, curvature, flow accumulation, flow direction, watersheds, stream network, and stream order rasters.*

Unfortunately, the weighted overlay tool would not allow me to assign weights to any of my reclassified rasters, and the tool kept hanging up while attempting to assign weights. Therefore, I remembered the *Weighted Sum* tool also overlays several rasters, multiplying each by its given weight and summing them together, so I used it instead of the weighted overlay output. I decided to use the stand-alone tool for this final part of the workflow since it was the last tool, and I wanted to be sure the resulting output was correct. It worked immediately, resulting in a meaningful output.



```python
#-------------------------------------------------------------------------------
import arcpy
import os
import pandas as pd

def zonal_statistics(input_zones, zone_field, rasters, output_folder):
    # Ensure the output folder exists
    if not os.path.exists(output_folder):
        os.makedirs(output_folder)

    results = []

    for raster in rasters:
        raster_name = os.path.basename(raster).replace('.tif', '')
        output_table = os.path.join(output_folder, f"{raster_name}_zonal_stats.dbf")

        try:
            print(f"Calculating zonal statistics for {raster}...")
            # Perform Zonal Statistics as Table with the specified zone field
            zonal_stats = arcpy.sa.ZonalStatisticsAsTable(input_zones, zone_field, raster, output_table, "NODATA", "ALL")
            results.append(output_table)
            print(f"Zonal statistics table created: {output_table}")
        except Exception as e:
            print(f"Error calculating zonal statistics for {raster}: {e}")

    return results

def combine_results_to_table(results, output_table, raster_names):
    # Initialize an empty DataFrame to hold combined results
    combined_df = pd.DataFrame()

    for result, raster_name in zip(results, raster_names):
        # Check if the result table exists and has data
        if os.path.exists(result):
            try:
                # Read each zonal stats table into a DataFrame
                df = pd.DataFrame(arcpy.da.TableToNumPyArray(result, "*"))

                # Check if DataFrame is not empty
                if not df.empty:
                    # Rename columns to include the raster name for clarity
                    df = df.add_prefix(f"{raster_name}_")

                    # Concatenate this raster's DataFrame to the combined DataFrame
                    combined_df = pd.concat([combined_df, df], axis=1)
                else:
                    print(f"Warning: {result} is empty.")

            except Exception as e:
                print(f"Error processing result for {raster_name}: {e}")
        else:
            print(f"Warning: Result table {result} does not exist.")

    # Save combined results to a CSV if there are results to combine
    if not combined_df.empty:
        combined_df.to_csv(output_table, index=False)
        print(f"Combined results saved to: {output_table}")
    else:
```

*Figure 6. Python script using the Zonal Statistics tool as well as the Table to NumPy Array tool for conversion to a table.*



```
from arcpy import env
from arcpy.sa import Reclassify

# Set the environment
env.workspace = r"D:\DigiTerrain\DTA_week9\Output"
arcpy.env.overwriteOutput = True  # Enable overwriting of output files

def reclassify_raster(input_raster, output_raster, remap):
    """Reclassify the raster using the Spatial Analyst Reclassify tool."""
    try:
        print(f"Starting reclassification for {input_raster}...")

        # Check out the Spatial Analyst extension
        arcpy.CheckOutExtension("Spatial")

        # Get raster properties
        min_value = arcpy.GetRasterProperties_management(input_raster, "MINIMUM")
        max_value = arcpy.GetRasterProperties_management(input_raster, "MAXIMUM")
        print(f"Minimum value: {min_value.getOutput(0)}, Maximum value: {max_value.getOutput(0)}")

        # Perform reclassification
        reclassified_raster = Reclassify(input_raster, "VALUE", remap)
        reclassified_raster.save(output_raster)

        print(f"Successfully reclassified {input_raster} to {output_raster}.")

    except Exception as e:
        print(f"Error processing {input_raster}: {e}")
    finally:
        arcpy.CheckInExtension("Spatial")  # Check in the extension

# Define paths and reclassification rules
raster_outputs = [
    (r"D:\DigiTerrain\DTA_week9\Output\filled_raster.tif", r"D:\DigiTerrain\DTA_week9\Output\reclassified_filled_raster.tif", "0 0 1; 1 1 2; 2 2 3"),
    (r"D:\DigiTerrain\DTA_week9\Output\slope.tif", r"D:\DigiTerrain\DTA_week9\Output\reclassified_slope.tif", "0 0 1; 1 5 2; 6 6 3"),
    (r"D:\DigiTerrain\DTA_week9\Output\curvature.tif", r"D:\DigiTerrain\DTA_week9\Output\reclassified_curvature.tif", "0 0 1; 1 2 2; 3 3 3"),
    (r"D:\DigiTerrain\DTA_week9\Output\clipped_flow_accumulation.tif", r"D:\DigiTerrain\DTA_week9\Output\reclassified_flow_accumulation.tif", "0 0 1; 1 10 2; 11 11 3"),
    (r"D:\DigiTerrain\DTA_week9\Output\clipped_flow_direction.tif", r"D:\DigiTerrain\DTA_week9\Output\reclassified_flow_direction.tif", "0 0 1; 1 8 2; 9 9 3"),
    (r"D:\DigiTerrain\DTA_week9\Output\watersheds.tif", r"D:\DigiTerrain\DTA_week9\Output\reclassified_watersheds.tif", "0 0 1; 1 2 2"),
    (r"D:\DigiTerrain\DTA_week9\Output\stream_network.tif", r"D:\DigiTerrain\DTA_week9\Output\reclassified_stream_network.tif", "0 0 1; 1 5 2; 6 6 3"),
    (r"D:\DigiTerrain\DTA_week9\Output\stream_order.tif", r"D:\DigiTerrain\DTA_week9\Output\reclassified_stream_order.tif", "0 0 1; 1 3 2; 4 4 3")
]

# Reclassify each raster
for input_raster, output_raster, remap in raster_outputs:
    reclassify_raster(input_raster, output_raster, remap)

print("All reclassification processes completed.")
```

*Figure 7. Python script using the Reclassify tool with class intervals set based on the Get Raster Properties function.*

# Results

After different combinations of tools in Python scripts and model builder and several failed attempts, the resulting watershed output did not come out as expected. I expected to have sub-watersheds covering the entire county; however, each attempt outputs small watershed systems along the major river system. Therefore, I had to work with the watershed output that had the most visible watersheds along the river system. Even though the watershed result was not as I expected, after putting all the layers together and analyzing the output from the weighted sum tool, the final raster layer showed meaningful results of flood-prone areas surrounding the various tributaries located in Buncombe County. To validate the findings, I overlayed the final output with FEMA's recent 2024 flood hazard zone map. After using the boundary layer for

clipping, the flood zones lay agreeably over high to moderate risk areas of the final flood risk output, confirming the results as seen in Figure 8.

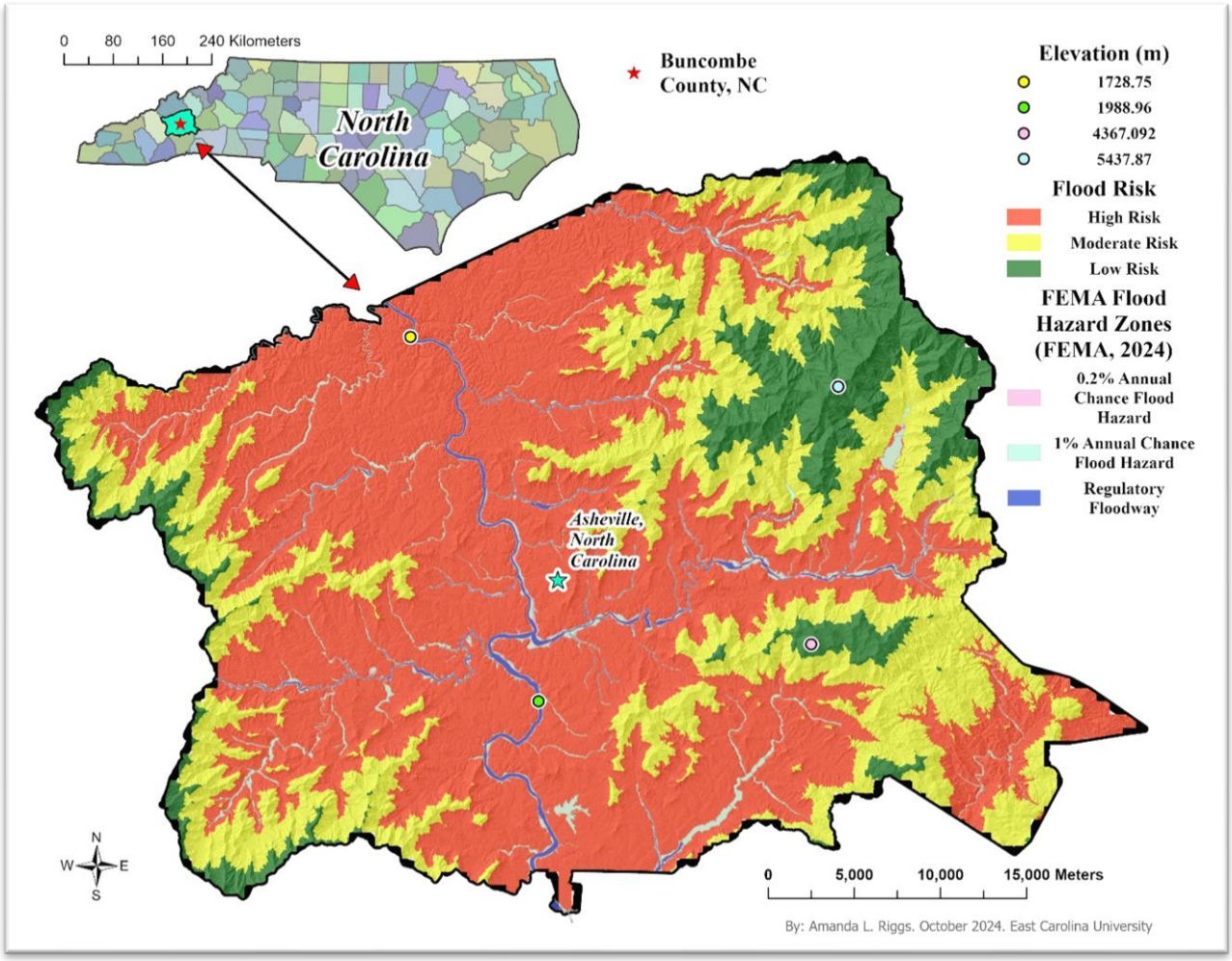



*Figure 8. The final flood risk map, including a reference map of North Carolina, elevation values in meters, and a FEMA flood hazard zone map overlay for validation of results.*

# Conclusion

All the tools worked as expected, with most tools producing meaningful results. However, some tools caused delays, such as the snap pour point tool and the weighted overlay tool. Future work for this project should include refining the Python script, including the tools that failed, and combining the Python script into one automated workflow for use in future projects.

# References


Census. (2022). TIGER/Line shapefile, 2022, nation, U.S., current County and equivalent national [Data]. *U.S. Department of Commerce, U.S. Census Bureau, Geography Division, Spatial Data Collection and Products Branch.* https://www2.census.gov/geo/tiger/TIGER2022/COUSUB/

Esri. (n.d.). Aspect function. *ArcGIS Pro 3.3* | *Documentation*. Aspect function—ArcGIS Pro | Documentation

Esri. (n.d.). Calculate statistics (data management). *ArcGIS Pro 3.3* | *Documentation.* https://pro.arcgis.com/en/pro-app/latest/tool-reference/data-management/calculate-statistics.htm

Esri. (n.d.). Clip raster (Data management). *ArcGIS Pro 3.3* | *Documentation*. Clip Raster (Data Management)—ArcGIS Pro | Documentation

Esri. (n.d.). Con (Spatial analyst). *ArcGIS Pro 3.3* | *Documentation.* *https://pro.arcgis.com/en/pro-app/latest/tool-reference/spatial-analyst/con-.htm*

Esri. (2023, July 27). Create a watershed model using hydrology in ArcGIS Pro. *ArcGIS Pro 3.3* | *Documentation*. https://support.esri.com/en-us/knowledge-base/how-to-create-a-watershed-model-using-hydrology-in-arcg-000023169

Esri. (2020, April 25) Create a threshold raster to be used as an input for the spatial analyst hydrology tools. *Esri* |*Technical Support*. How To: Create a Threshold Raster to Be Used as an Input for the Spatial Analyst Hydrology

Esri. (n.d.). Curvature function. *ArcGIS Pro 3.3* | *Documentation*. How Flow Accumulation Works—ArcGIS Pro | Documentation

Esri. (n.d.). Extract values to points (Spatial analyst). *ArcGIS Pro 3.3* | *Documentation.* https://pro.arcgis.com/en/pro-app/latest/tool-reference/spatial-analyst/extract-values-to-points.htm

Esri. (n.d.). Fill (Spatial analyst). *ArcGIS Pro 3.3* | *Documentation*. Fill (Spatial Analyst)—ArcGIS Pro | Documentation

Esri. (n.d.). Flow accumulation (Spatial analyst). *ArcGIS Pro 3.3* | *Documentation*. Flow Accumulation (Spatial Analyst)—ArcGIS Pro | Documentation

Esri. (n.d.). Flow direction (Spatial analyst). *ArcGIS Pro 3.3* | *Documentation*. Flow Direction (Spatial Analyst)—ArcGIS Pro | Documentation

Esri. (n.d.). Get raster properties (Data management). *ArcGIS Pro 3.3* | *Documentation*. https://pro.arcgis.com/en/pro-app/latest/tool-reference/data-management/get-raster-properties.htm

Esri. (n.d.). How flow accumulation works. *ArcGIS Pro 3.3* | *Documentation*. How Flow Accumulation Works—ArcGIS Pro | Documentation

Esri. (n.d.). How hillshade works. *ArcGIS Pro 3.3* | *Documentation*. How Hillshade works—ArcGIS Pro | Documentation

Esri. (n.d.). How stream order works. *ArcGIS Pro 3.3* | *Documentation*. https://pro.arcgis.com/en/pro-app/latest/tool-reference/spatial-analyst/how-stream-order-

works.htm#:~:text=The%20Strahler%20method%20is%20the%20most%20common%20stream%20ordering%20method.

Esri. (n.d.). Hydrologic analysis sample applications. *ArcGIS Pro | ArcGIS desktop*. Hydrologic analysis sample applications—ArcGIS Pro | Documentation

Esri. (n.d.). Project raster. *ArcGIS Pro 3.3 | Documentation. https://pro.arcgis.com/en/pro-app/latest/tool-reference/data-management/project-raster.htm*

Esri. (n.d.). Raster to point (Conversion). *ArcGIS Pro 3.3 | Documentation.* https://pro.arcgis.com/en/pro-app/latest/tool-reference/conversion/raster-to-point.htm

Esri. (n.d.). Raster to polygon (Conversion). *ArcGIS Pro 3.3 | Documentation.* https://pro.arcgis.com/en/pro-app/latest/tool-reference/conversion/raster-to-polygon.htm

Esri. (n.d.). Reclassify (Spatial analyst). *ArcGIS Pro 3.3 | Documentation.* Reclassify (Spatial Analyst)—ArcGIS Pro | Documentation

Esri. (n.d.). Slope (Spatial analyst). *ArcGIS Pro 3.3 | Documentation*. Slope (Spatial Analyst)—ArcGIS Pro | Documentation

Esri. (n.d.). Stream link (Spatial analyst). *ArcGIS Pro 3.3 | Documentation.* Stream Link (Spatial Analyst)—ArcGIS Pro | Documentation

Esri. (n.d.). Stream order (Spatial analyst). *ArcGIS Pro 3.3 | Documentation*. https://pro.arcgis.com/en/pro-app/latest/tool-reference/spatial-analyst/stream-order.htm

Esri. (n.d.). Surface analysis sample applications. *ArcGIS Pro 3.3 | Documentation*. https://pro.arcgis.com/en/pro-app/latest/tool-reference/3d-analyst/surface-analysis-sample-

applications.htm#:~:text=Classify%20landforms%20and%20inform%20understanding%20of%20geomorphologic%20processes.&text=You%20might%20want%20to%20know,being%20those%20most%20at%20risk).&text=You%20might%20be%20a%20farmer,area%20with%20a%20southerly%20aspect.&text=Determine%20the%20acceleration%20and%20convergence,basin%20to%20better%20understand%20erosion.&text=Contours%20can%20be%20useful%20for,overall%20gradation%20of%20the%20land.&text=You%20can%20create%20hillshade%20for,and%20raising%20the%20sun%20angle.&text=You%20may%20be%20leveling%20a,deposition%20in%20a%20river%20valley.

Esri. (n.d.). Weighted overlay (Spatial analysis). *ArcGIS Pro 3.3 | Documentation.* Weighted Overlay (Spatial Analyst)—ArcGIS Pro | Documentation

Esri. (n.d.). Weighted sum (Spatial analyst). *ArcGIS Pro 3.3 | Documentation. https://pro.arcgis.com/en/pro-app/latest/tool-reference/spatial-analyst/weighted-sum.htm*

Esri. (n.d.). Zonal statistics (Spatial analysis). *ArcGIS Pro 3.3 | Documentation*. Zonal Statistics (Spatial Analyst)—ArcGIS Pro | Documentation

FEMA. (2024, June 27). FEMA flood hazard areas [Data]. *Federal Emergency Management Agency*. https://www.arcgis.com/home/item.html?id=2b245b7f816044d7a779a61a5844be23

Mazzocchi, J. (2006). Buncombe County. Encyclopedia of North Carolina. *University of North Carolina Press. Retrieved from NCpedia*. https://www.ncpedia.org/geography/buncombe#:~:text=Buncombe%20County%2C%20nicknamed%20%22Land%20of,came%20to%20populate%20the%20region.

NASA Earth Observatory. (2024, October 5). Flash floods swamp North Carolina. *NASA*. https://earthobservatory.nasa.gov/images/153416/flash-floods-swamp-north-carolina#:~:text=After%20hitting%20Florida%2C%20Hurricane%20Helene,were%20at%20more%20normal%20levels.

NC. Center for Geographic Information and Analysis, NC Department of Public Safety (2019). Digital Elevation Model (3 fr. Grid Cells). Buncombe County [Data]. *North Carolina Department of Information Technology, Government Data Analytics Center, Center for Geographic*. https://www.nconemap.gov/#directdatadownloads

OpenAI. (2024). ChatGPT (v2) [AI language model]. Retrieved from https://www.openai.com/chatgpt

Topographic Map. (n.d.). Buncombe County. *Retrieved from Topographic Map*. https://en-us.topographic-map.com/map-p9kb3/Buncombe-County/?center=38.14698%2C-102.4

Wikipedia. (2024, May 17). Coefficient of variation. *The Free Encyclopedia.* Coefficient of variation - Wikipedia