lab3_1

November 24, 2023

1 Identify the cost path using Lidar data with varying weights

1.0.1 GIS 5571 Lab 3 part1

Tzu Yu Ma

November 28

[1]:
```
# information about Dory's prefer
start = "44.127985, -92.148796"
end = "44.05431509462441, -92.04443552409354" # North Picnic area
# Minnesota Digital Elevation Model - 30 Meter Resolution
dem_url = "https://resources.gisdata.mn.gov/pub/gdrs/data/pub/us_mn_state_dnr/
 ↪elev_30m_digital_elevation_model/fgdb_elev_30m_digital_elevation_model.zip"
# NLCD 2019 Land Cover, Minnesota
lc_url = "https://resources.gisdata.mn.gov/pub/gdrs/data/pub/us_mn_state_dnr/
 ↪biota_landcover_nlcd_mn_2019/tif_biota_landcover_nlcd_mn_2019.zip"
# water
water_url = 'https://resources.gisdata.mn.gov/pub/gdrs/data/pub/us_mn_state_dnr/
 ↪water_strahler_stream_order/shp_water_strahler_stream_order.zip'
# counties
county_url = "https://resources.gisdata.mn.gov/pub/gdrs/data/pub/
 ↪us_mn_state_dnr/bdry_counties_in_minnesota/shp_bdry_counties_in_minnesota.
 ↪zip"
```

[ ]:
```
import requests
import zipfile
import os
import arcpy
import numpy as np
from arcpy.sa import *
```

download data

[ ]:
```
# download DEM and extract
dem = requests.get(dem_url)

with open (r"D:\arc1\lab2\part2\fgdb_elev_30m_digital_elevation_model.zip",␣
 ↪"wb") as dem_df:
```

```
    dem_df.write(dem.content)

sub_dir = "fgdb_elev_30m_digital_elevation_model"
unzip_path = os.path.join(r"D:\arc1\lab2\part2", sub_dir)

os.makedirs(unzip_path, exist_ok = True)

with zipfile.ZipFile(r"D:\arc1\lab2\part2\fgdb_elev_30m_digital_elevation_model.
 ↪zip", "r") as dem_zip:
    dem_zip.extractall(unzip_path)

print(f"Unzipped DEM data to {unzip_path}")
```

[ ]:
```
# NLCD
lc = requests.get(lc_url)

with open (r"D:\arc1\lab2\part2\tif_biota_landcover_nlcd_mn_2019.zip", "wb") as␣
 ↪lc_df:
    lc_df.write(lc.content)

sub_dir = "tif_biota_landcover_nlcd_mn_2019"
unzip_path = os.path.join(r"D:\arc1\lab2\part2", sub_dir)

os.makedirs(unzip_path, exist_ok = True)

with zipfile.ZipFile(r"D:\arc1\lab2\part2\tif_biota_landcover_nlcd_mn_2019.
 ↪zip", "r") as lc_zip:
    lc_zip.extractall(unzip_path)

print(f"Unzipped NLCD data to {unzip_path}")
```

[ ]:
```
# water
water = requests.post(water_url)

with open (r"D:\arc1\lab2\part2\shp_water_strahler_stream_order.zip", "wb") as␣
 ↪w_df:
    w_df.write(water.content)

sub_dir = "shp_water_strahler_stream_order"
unzip_path = os.path.join(r"D:\arc1\lab2\part2", sub_dir)

os.makedirs(unzip_path, exist_ok = True)

with zipfile.ZipFile(r"D:\arc1\lab2\part2\shp_water_strahler_stream_order.zip",␣
 ↪"r") as w_zip:
    w_zip.extractall(unzip_path)
```

```
print(f"Unzipped water data to {unzip_path}")
```

[ ]:
```
# counties
c = requests.get(county_url)
with open (r"D:\arc1\lab2\part2\shp_bdry_counties_in_minnesota.zip", 'wb') as␣
↪c_zip:
    c_zip.write(c.content)

sub_dir = "shp_bdry_counties_in_minnesota"
unzip_path = os.path.join(r"D:\arc1\lab2\part2", sub_dir)

os.makedirs(unzip_path, exist_ok = True)

with zipfile.ZipFile(r"D:\arc1\lab2\part2\shp_bdry_counties_in_minnesota.zip",␣
↪"r") as c_ref:
    c_ref.extractall(unzip_path)

print(f"Unzipped county data to {unzip_path}")
```

prepare the study extent

[ ]:
```
# Create Study Extent (Feature Class to Feature Class)
arcpy.conversion.FeatureClassToFeatureClass(r"D:
↪\arc1\lab2\part2\shp_bdry_counties_in_minnesota\mn_county_boundaries.shp",␣
↪r"D:\arc1\lab2\part2\Lab2_2\Lab2_2.gdb", "StudyExtent", "CTY_NAME =␣
↪'Wabasha' Or CTY_NAME = 'Winona' Or CTY_NAME = 'Olmsted'", 'AREA "AREA" true␣
↪true false 19 Double 0 0,First,#,mn_county_boundaries,AREA,-1,-1;PERIMETER␣
↪"PERIMETER" true true false 19 Double 0␣
↪0,First,#,mn_county_boundaries,PERIMETER,-1,-1;CTYONLY_ "CTYONLY_" true true␣
↪false 19 Double 0 0,First,#,mn_county_boundaries,CTYONLY_,-1,-1;CTYONLY_ID␣
↪"CTYONLY_ID" true true false 19 Double 0␣
↪0,First,#,mn_county_boundaries,CTYONLY_ID,-1,-1;COUN "COUN" true true false␣
↪4 Short 0 4,First,#,mn_county_boundaries,COUN,-1,-1;CTY_NAME "CTY_NAME" true␣
↪true false 20 Text 0 0,First,#,mn_county_boundaries,CTY_NAME,0,20;CTY_ABBR␣
↪"CTY_ABBR" true true false 4 Text 0␣
↪0,First,#,mn_county_boundaries,CTY_ABBR,0,4;CTY_FIPS "CTY_FIPS" true true␣
↪false 4 Short 0 4,First,#,mn_county_boundaries,CTY_FIPS,-1,-1;Shape_Leng␣
↪"Shape_Leng" true true false 19 Double 0␣
↪0,First,#,mn_county_boundaries,Shape_Leng,-1,-1;Shape_Area "Shape_Area" true␣
↪true false 19 Double 0 0,First,#,mn_county_boundaries,Shape_Area,-1,-1', '')
```

[ ]:
```
# Dissolve County Boundaries
arcpy.management.Dissolve("StudyExtent", r"D:\arc1\lab2\part2\Lab2_2\Lab2_2.
↪gdb\StudyExtent_Dissolve", None, None, "MULTI_PART", "UNSPLIT_LINES")
```

prepare water stream

[ ]:
```
# Clip Streams
arcpy.analysis.Clip("streams_with_strahler_stream_order",␣
↪"StudyExtent_Dissolve", r"D:\arc1\lab2\part2\Lab2_2\Lab2_2.
↪gdb\waterstream_Clip", None)
```

[ ]:
```
# Feature to Raster
arcpy.conversion.FeatureToRaster("waterstream_Clip", "SO_VALUE", r"D:
↪\arc1\lab2\part2\Lab2_2\Lab2_2.gdb\Reclass_waterstream")
```

[ ]:
```
# Reclassify Streams
arcpy.ddd.Reclassify("Reclass_waterstream", "Value", "1 1;2 2;3 3;4 7;5 8;6 9;8␣
↪10", r"D:\arc1\lab2\part2\Lab2_2\Lab2_2.gdb\Reclass_Streams", "DATA")
```

prepar landcover

[ ]:
```
# Define the input raster dataset
farm_field = r"D:
↪\arc1\lab2\part2\tif_biota_landcover_nlcd_mn_2019\NLCD_2019_Land_Cover.tif"

# Define the output clipped raster
output_clip = r"D:\arc1\lab2\part2\Lab2_2\Lab2_2.gdb\LandCover_clip"

# Define the clipping feature
clip_feature = "StudyExtent_Dissolve"

# Perform the clip operation
arcpy.management.Clip(farm_field, clip_feature, output_clip)
```

[ ]:
```
# Reclassify Landcover
#farm_field_reclass = arcpy.sa.Reclassify("LandCover_clip.tif", "Value", "1 81␣
↪1;82 2;83 255 1", "DATA")

# Reclassify NLCD
arcpy.ddd.Reclassify("LandCover_clip", "NLCD_Land", "'Open Water' 10;
↪'Developed, Open Space' 2;'Developed, Low Intensity' 2;'Developed, Medium␣
↪Intensity' 2;'Developed, High Intensity' 2;'Barren Land' 5;'Deciduous␣
↪Forest' 7;'Evergreen Forest' 7;'Mixed Forest' 7;Shrub/Scrub 7;Herbaceous 7;
↪Hay/Pasture 9;'Cultivated Crops' 9;'Woody Wetlands' 10;'Emergent Herbaceous␣
↪Wetlands' 10", r"D:\arc1\lab2\part2\Lab2_2\Lab2_2.gdb\Reclass_NLCD", "DATA")
```

prepare slope

[ ]:
```
# Define the input raster dataset
DEM = r"D:
↪\arc1\lab2\part2\fgdb_elev_30m_digital_elevation_model\elev_30m_digital_elevation_model.
↪gdb\digital_elevation_model_30m"
```

```
# Define the output clipped raster
output_clip = r"D:\arc1\lab2\part2\Lab2_2\Lab2_2.gdb\DEM_clip"

# Define the clipping feature
clip_feature = "StudyExtent_Dissolve"

# Perform the clip operation
arcpy.management.Clip(DEM , clip_feature, output_clip)
```

[ ]:
```
# Calculate Slope
slope_name = r"D:\arc1\lab2\part2\Lab2_2\Lab2_2.gdb\DEM_clip\slope"

slope = arcpy.sa.Slope("DEM_clip", "DEGREE", 1, "PLANAR", "METER")

slope.save(slope_name)
```

[ ]:
```
# Reclassify Slope
Reclass_slope = arcpy.sa.Reclassify("slope", "VALUE", "0 5 5;5 10 4;10 15 3;15␣
↪20 2;20 90 1", "DATA")
```

create start & end points

[ ]:
```
# Create start point
start = arcpy.Point(44.127985, -92.148796)

# Create a point geometry
start_point = arcpy.PointGeometry(start, arcpy.SpatialReference(4326))
# Save the point geometry to a feature class
output_fc = r"D:\arc1\lab2\part2\Lab2_2\Lab2_2.gdb\Start_Point"

arcpy.CopyFeatures_management(start_point, output_fc)
```

[ ]:
```
# Create end point
end = arcpy.Point(44.05431509462441, -92.04443552409354)

# Create a point geometry
end_point = arcpy.PointGeometry(end, arcpy.SpatialReference(4326))
# Save the point geometry to a feature class
output_fc = r"D:\arc1\lab2\part2\Lab2_2\Lab2_2.gdb\End_Point"

arcpy.CopyFeatures_management(end_point, output_fc)
```

cost path functions

[1]:
```
# convert waterstream to raster
arcpy.conversion.FeatureToRaster(
    in_features="waterstream_Clip",
```

```
    field="SO_VALUE",
    out_raster=r"D:\fall2023\arc1\lab3\lab3_1\lab3_1.gdb\water_lab3",
    cell_size=276.583641600005
)
```

[ ]:
```
# reclassify waterstream raster (water:1, other:0)
arcpy.ddd.Reclassify(
    in_raster="water_lab3",
    reclass_field="Value",
    remap="1 1;2 1;3 1;4 1;5 1;6 1;8 1;NODATA 0",
    out_raster=r"D:\fall2023\arc1\lab3\lab3_1\lab3_1.gdb\Reclass_wate_Lab3",
    missing_values="DATA"
)
```

[ ]:
```
# reclassfy land use
arcpy.ddd.Reclassify(
    in_raster="LandCover_clip",
    reclass_field="NLCD_Land",
    remap="'Open Water' 2;'Developed, Open Space' 1;'Developed, Low Intensity'␣
↪1;'Developed, Medium Intensity' 1;'Developed, High Intensity' 1;'Barren␣
↪Land' 1;'Deciduous Forest' 1;'Evergreen Forest' 1;'Mixed Forest' 1;Shrub/
↪Scrub 1;Herbaceous 1;Hay/Pasture 1;'Cultivated Crops' 1;'Woody Wetlands' 2;
↪'Emergent Herbaceous Wetlands' 2",
    out_raster=r"D:\fall2023\arc1\lab3\lab3_1\lab3_1.gdb\Reclass_Land_Lab3",
    missing_values="DATA"
)
```

[ ]:
```
# reclassify slope
arcpy.ddd.Reclassify(
    in_raster="slope",
    reclass_field="VALUE",
    remap="0 4.046986 1;4.046986 8.716586 2;8.716586 13.697492 3;13.697492 19.
↪301012 4;19.301012 25.527145 5;25.527145 32.375891 6;32.375891 39.224637 7;
↪39.224637 46.073383 8;46.073383 52.610823 9;52.610823 58.836956 10;58.836956␣
↪79.383194 11",
    out_raster=r"D:\fall2023\arc1\lab3\lab3_1\lab3_1.gdb\Reclass_slop_Lab3",
    missing_values="DATA"
)
```

path 1: slope * 0.5, landuse * 0.3, water * 0.2

[1]:
```
output_raster = arcpy.ia.RasterCalculator(
    expression='("Reclass_slop_Lab3"*0.5)+("Reclass_Land_Lab3"*0.
↪3)+("Reclass_wate_Lab3"*0.2)'
)
output_raster.save(r"d:\fall2023\arc1\lab3\lab3_1\lab3_1.gdb\reclas_raste3")
```

[ ]:
```
out_distance_raster = arcpy.sa.CostDistance(
    in_source_data="start",
    in_cost_raster="reclas_raste",
    maximum_distance=None,
    out_backlink_raster=r"D:\fall2023\arc1\lab3\lab3_1\lab3_1.gdb\backlink1",
    source_cost_multiplier=None,
    source_start_cost=None,
    source_resistance_rate=None,
    source_capacity=None,
    source_direction=""
)
out_distance_raster.save(r"D:\fall2023\arc1\lab3\lab3_1\lab3_1.gdb\CostDis1")
```

[ ]:
```
out_raster = arcpy.sa.CostPath(
    in_destination_data="end",
    in_cost_distance_raster="CostDis1",
    in_cost_backlink_raster="backlink1",
    path_type="EACH_CELL",
    destination_field="OBJECTID",
    force_flow_direction_convention="INPUT_RANGE"
)
out_raster.save(r"D:\fall2023\arc1\lab3\lab3_1\lab3_1.gdb\CostPat1")
```

path2: slope * 0.8, landuse * 0.1, water * 0.1

[6]:
```
output_raster = arcpy.ia.RasterCalculator(
    expression='("Reclass_slop_Lab3"*0.8)+("Reclass_Land_Lab3"*0.
 ↪1)+("Reclass_wate_Lab3"*0.1)'
)
output_raster.save(r"d:\fall2023\arc1\lab3\lab3_1\lab3_1.gdb\reclas_raste2")
```

[ ]:
```
out_distance_raster = arcpy.sa.CostDistance(
    in_source_data="start",
    in_cost_raster="reclas_raste",
    maximum_distance=None,
    out_backlink_raster=r"D:\fall2023\arc1\lab3\lab3_1\lab3_1.gdb\backlink2",
    source_cost_multiplier=None,
    source_start_cost=None,
    source_resistance_rate=None,
    source_capacity=None,
    source_direction=""
)
out_distance_raster.save(r"D:\fall2023\arc1\lab3\lab3_1\lab3_1.gdb\CostDis2")
```

[ ]:
```
out_raster = arcpy.sa.CostPath(
    in_destination_data="end",
    in_cost_distance_raster="CostDis2",
```

```
    in_cost_backlink_raster="backlink2",
    path_type="EACH_CELL",
    destination_field="OBJECTID",
    force_flow_direction_convention="INPUT_RANGE"
)
out_raster.save(r"D:\fall2023\arc1\lab3\lab3_1\lab3_1.gdb\CostPat2")
```

path3: slope * 0.2, landuse * 0.65, water * 0.15

[ ]:
```
output_raster = arcpy.ia.RasterCalculator(
    expression='("Reclass_slop_Lab3"*0.2)+("Reclass_Land_Lab3"*0.
 ↪65)+("Reclass_wate_Lab3"*0.15)'
)
output_raster.save(r"d:\fall2023\arc1\lab3\lab3_1\lab3_1.gdb\reclas_raste3")
```

[ ]:
```
out_distance_raster = arcpy.sa.CostDistance(
    in_source_data="start",
    in_cost_raster="reclas_raste",
    maximum_distance=None,
    out_backlink_raster=r"D:\fall2023\arc1\lab3\lab3_1\lab3_1.gdb\backlink3",
    source_cost_multiplier=None,
    source_start_cost=None,
    source_resistance_rate=None,
    source_capacity=None,
    source_direction=""
)
out_distance_raster.save(r"D:\fall2023\arc1\lab3\lab3_1\lab3_1.gdb\CostDis3")
```

[ ]:
```
out_raster = arcpy.sa.CostPath(
    in_destination_data="end",
    in_cost_distance_raster="CostDis3",
    in_cost_backlink_raster="backlink3",
    path_type="EACH_CELL",
    destination_field="OBJECTID",
    force_flow_direction_convention="INPUT_RANGE"
)
out_raster.save(r"D:\fall2023\arc1\lab3\lab3_1\lab3_1.gdb\CostPat3")
```