

National Forests Trail Rating Application Procedures

Data Collection

Goal: Collect line features of trails inside the National Forests and National Forest boundary polygon features. Also collect detailed digital elevation model (DEM) and slope raster of contiguous United States.

1. Download National Forest trails geodatabase from the Forest Service's FSGeodata Clearinghouse website. On the home page click the link reading "Downloadable Data." The URL to that page is <https://data.fs.usda.gov/geodata/edw/datasets.php> for reference.
2. Once on this page, find the feature classes titled "Administrative Forest Boundaries" and "National Forest System Trails." It is important to download these as ESRI geodatabases and not shapefiles. Geodatabases allow for better data compression and for longer field names, which will be vital when defining difficulty classes.
3. Download DEM and slope rasters of contiguous United States. The best source to use is the EDNA dataset found on the USGS Earth Explorer web utility (<https://earthexplorer.usgs.gov/>).
 - a) Because the files that will be used are extremely large (around 60GB each once unzipped), USGS requires that an account be set up first for the site. They do this to better monitor where their large data sets are going and what they're being used for. To do this, click the "Register" link in the right hand corner of the Earth Explorer page and follow the instructions.
 - b) Once the account is created and the user is at the web utility, click the "Data Sets" tab, open the drop down menu labeled "Digital Elevation," select "EDNA" from subsequent list, and click "Results". (See Figure 1). In the resulting "Results" tab, advance to page three, and select the layer names "ORIG_DEM" and "SLOPE" from the list by clicking the footprint-shaped icon (See Figure 2). Then, click the "Download Options" icon (circled in Figure 2) and download.
4. Unzip all downloaded datasets.

Notes on Data Collection:

- ENDA was chosen for three reasons: (1) it is a complete DEM of the contiguous United States, which eases the elevation data extraction process, (2) the data set is detailed, with each pixel being 30x30 meters; (3) it had a slope raster as well, which eliminated the need for one to be calculated.

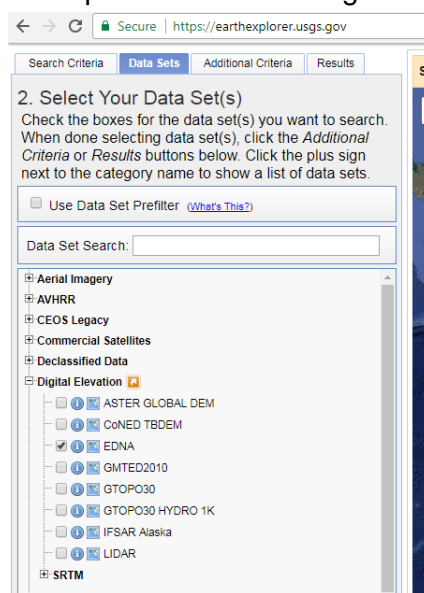


Figure 1

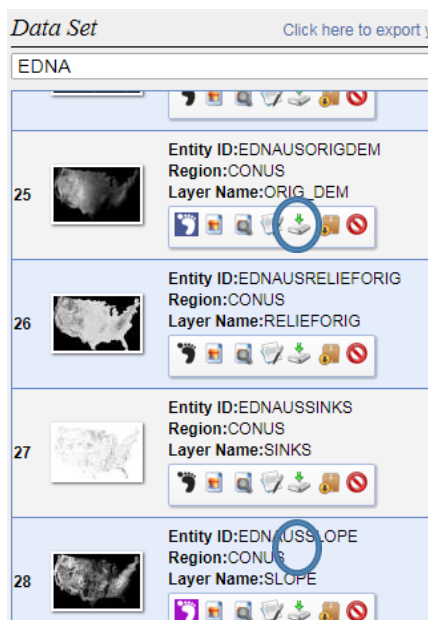


Figure 2

Data Splitting and Extraction

Goal: Create segments of each trail within the United States National Forests that include a field describing elevation and slope.

1. Open ArcCatalog and new ArcMap document.
2. In ArcCatalog, create three new file geodatabases in the appropriate folder. Name the first "Trails_Project" because this will be the geodatabase that will hold the trail feature classes before splitting and after processing. Name the second "Trails_Working" because this geodatabase will be where the split features will be stored before they are processed by the code which will be explained below. The third geodatabase should be named something to the effect of "Trails_Processed", which is where the processed trail features will be stored.
3. In the ArcMap window, click the "Add Data" icon and add the National Forest Trails feature class (titled TrailNFS_Publish), the AdministrativeForest feature class, and the EDNA DEM from their respective locations.
4. Right click the EDNA layer and select the "Properties" option from the drop-down menu. Click the "Source" tab and scroll until you find the spatial reference of the DEM. For this DEM it should be NAD_1983_Albers. Remember this spatial reference.
5. Open ArcToolbox. Under Data Management Tools open the "Projections and Transformations" tab and open the "Batch Project" tool.
6. In the input feature class box, insert both the TrailNFS_Publish and AdministrativeForest feature classes. Next, select the output workspace to be "Trails_Project" made earlier. Then select the output coordinate system to be NAD_1983_Albers. If the result resembles Figure 3, then run the tool.

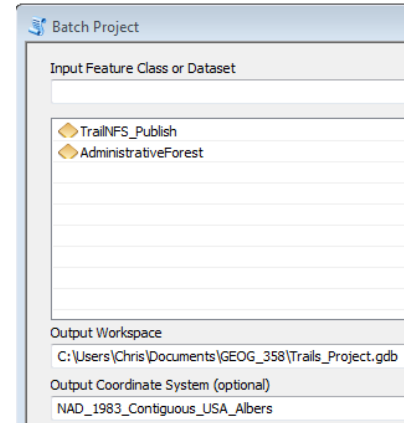


Figure 3

- a. These feature classes need to be projected to ensure the units of the extents of these feature classes match the extents of the DEM.

7. Because the projections were batched, you will need to manually add the projected features back into the ArcMap document.
8. Once added, remove the original trail and forest feature classes from the map document to avoid confusion.

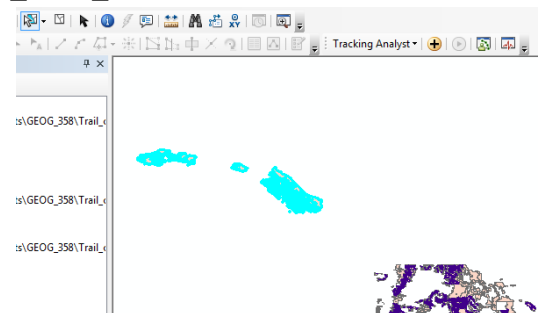


Figure 4

9. Because the DEM does not include elevation data for the state of Alaska, we will delete the trail and forest features located in Alaska.
10. Click the "Select Features" tool on the Tools toolbar, and then select the clump of features not located in the contiguous 48 states. (See Figure 4)
11. Open the Delete Features tool from within ArcToolbox, and then run the tool twice: deleting the selected features of trails and National Forests.
12. Open the Spatial Join tool from within ArcToolbox.
13. Set the arguments of the tool in a way that resembles

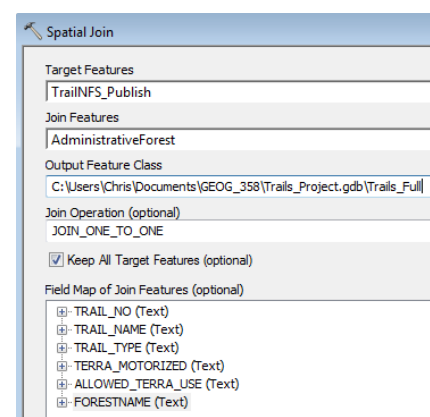


Figure 5

Figure 5. Our goal is to join to each trail the name of the National Forest it resides in. I also took the opportunity to remove all the unnecessary fields from the resulting feature class. This leaves only the trail number, trail name, trail type, whether motor vehicles can use it, what specifically can be done on the trail, and the forest name. Each of these fields will play an important role in sorting these trails later on.

14. Once the Spatial Join is complete, open the “Split by Attributes” tool within ArcToolbox. This tool splits a feature class into multiple feature classes based on a common field value.
 - a. This tool is only available with ArcMap 10.5 or newer.

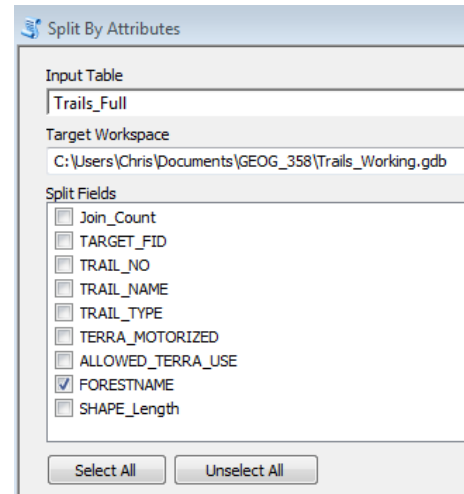


Figure 6

15. Select the “Trails_Full” feature as the input feature, select the output features to be saved to the “Trails_Working” geodatabase, and choose the split to be made by the “FORESTNAME” field. The window should resemble Figure 6.
 - a. The result of this tool will be a feature class in the “Trails_Working” geodatabase for each unique National Forest name value in the FORESTNAME field.
16. When the tool is complete, switch to ArcCatalog and open the “Trails_Working” geodatabase. Inside the folder, find the feature class simply titled “None” and delete it.
 - a. Within this feature class are all the trails that were not in the boundaries of a National Forest. Therefore, this feature class will have lines from all across the country and will cause the subsequent processing script to take many hours longer to run.
17. Close the ArcCatalog and ArcMap windows. You may save the ArcMap document as a .mxd file, but it isn’t required.
18. Open the Python IDLE located in the Python 2.7 subfolder of the ArcGIS program folder. A Python script will be needed to do the next step with any sort of efficiency.
19. Once the IDLE is open, click the “Run” drop-down menu and select “Python Shell” from the options. This will cause the Python Shell window to open up. Enter the commands found in Figure 7 to initialize the ArcPy module, the os module, both the Spatial Analyst tools and Environmental submodules within ArcPy, and to indicate to ArcPy that it is allowed to overwrite an existing file during this session. Then complete the “arcpy.env.workspace = “ argument with the string to the “Trails_Working” geodatabase in quotation marks.

```
import os
import arcpy

from arcpy import env
from arcpy.sa import *
env.overwriteOutput = True
arcpy.CheckOutExtension('Spatial')

arcpy.env.workspace =
```

Figure 7

20. Enter in the code in Figure 8, adjusting the paths of the clipped DEM and location of “Trails_Processed” to match your computer’s paths.
 - a. The reason we went through all the trouble to split up the feature classes and create this script is to maximize the efficiency of the Extract Values to Points tool. If all the features were processed at once, the Extract Values to Points tool takes hours to run given the size of the DEM being used. But if the features are restricted to a manageable area, like the size of a National Forest, the DEM can be clipped to a fraction of its original size and Extract Values to Points will run much, much faster. I estimate that processing the trail features by splitting them

up and running this script is around seven to nine hours faster than processing the features all at once.

Explanation of script:

- First line creates a list of all of the feature classes within the “Trails_Working” geodatabase. Each of the features in the geodatabase will be processed through the subsequent for-loop.
- The first three lines in the loop determine the extent of the feature class being processed. This extent will be the extent of the clipped raster.
- A section of the main DEM is clipped based on the extent of the feature class, and saved to a specified location. This clipped file will be overwritten every new iteration of the loop.
- Next, the scratch workspace is established. The four temporary files created while processing will be saved to the scratch workspace while they’re needed. Every computer that runs ArcGIS has a designated scratch workspace that files can be saved to that are only needed during the workflow of a task. These files will be overwritten during the next iteration, since they are not needed once processing is complete.
- The trail feature class is diced into segments using the Dice tool. The lines are segmented every 20 vertices.
- The lines are then converted to points using the Feature Vertices to Points tool. The ‘BOTH_ENDS’ argument specifies that a point is created of each end point of each line segment. The issue with this is that there will be an identical point describing the end of one line segment and the start of another. To address this, the Delete Identical tool is used to delete identical features that share the same location. The result of this is a single point at the meeting point of line segments.
- After the identical features have been deleted, the Extract Values to Points tool is

```
featureClasses = arcpy.ListFeatureClasses()

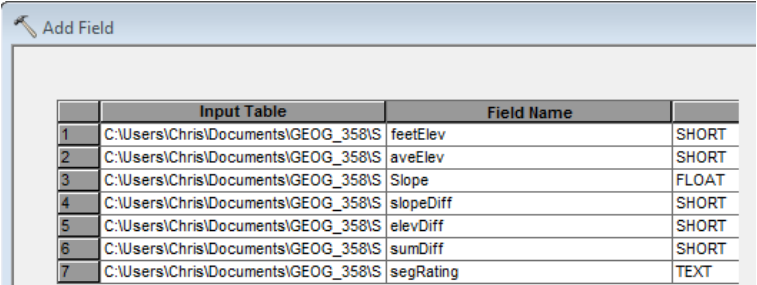
for fc in featureClasses:
    desc = arcpy.Describe(fc)
    extent = desc.extent
    extStr = "{} {} {} {}".format(extent.XMin, extent.YMin, extent.XMax, extent.YMax)
    DEMClip = r"C:\Users\Chris\Documents\GEOG_358\DEMClip.tif"
    arcpy.Clip_management(DEM, extStr, DEMClip, fc, "#", "#")
    scratcher = arcpy.env.scratchGDB
    pts = os.path.join(os.path.abspath(scratcher), "pts")
    ptsExt = os.path.join(os.path.abspath(scratcher), "ptsEXT")
    temp = os.path.join(os.path.abspath(scratcher), "temp")
    dice = os.path.join(os.path.abspath(scratcher), "Dice")
    arcpy.Dice_management(fc, dice, 20)
    arcpy.FeatureVerticesToPoints_management(dice, pts, "BOTH_ENDS")
    arcpy.DeleteIdentical_management(pts, "Shape")
    arcpy.sa.ExtractValuesToPoints(pts, DEMClip, ptsExt)
    arcpy.AlterField_management(ptsExt, "RASTERVALU", "startElev")
    fieldmappings = arcpy.FieldMappings()
    fieldmappings.addTable(dice)
    fieldmappings.addTable(ptsExt)
    for field in fieldmappings.fields:
        b=fieldmappings.getFieldMap(fieldmappings.findFieldMapIndex(field.name))
        b.mergeRule = 'First'
        fieldmappings.replaceFieldMap(fieldmappings.findFieldMapIndex(field.name),b)
    arcpy.SpatialJoin_analysis(dice, ptsExt, temp, "JOIN_ONE_TO_ONE", "#", fieldmappings)
    fieldmappings.removeAll()
    arcpy.AlterField_management(ptsExt, "startElev", "endElev")
    fieldmappings = arcpy.FieldMappings()
    fieldmappings.addTable(temp)
    fieldmappings.addTable(ptsExt)
    for field in fieldmappings.fields:
        b=fieldmappings.getFieldMap(fieldmappings.findFieldMapIndex(field.name))
        b.mergeRule = 'Last'
        fieldmappings.replaceFieldMap(fieldmappings.findFieldMapIndex(field.name),b)
    arcpy.SpatialJoin_analysis(temp, ptsExt,
                               os.path.join(r"C:\Users\Chris\Documents\GEOG_358\Trails_Processed.gdb",
                                              os.path.splitext(fc)[0]),
                               "JOIN_ONE_TO_ONE", "#", fieldmappings)
    fieldmappings.removeAll()
```

Figure 8

used to extract the elevation value at each point and add these values to a field in the feature's attribute table.

- The resulting field from the Extract Value to Points tool is renamed from the default 'RASTERVALU' to 'startElev.'
- Next, a Field Mappings object is created and the tables of the dice line features and the point features created by the Extract Value to Points tool are added to the Field Mappings object. Then a for-loop is run to set the merge rule of each field to 'First.' Setting this indicates to the subsequent Spatial Join to take the values of the first join feature that meets the join criterion for that particular target feature.
- The Spatial Join is run with the diced line features and extracted point features using the Field Mappings established above. The results of the join are line features with a new field named 'startElev' that describe the elevation at the start point of each line segment.
- The 'startElev' field in the extracted point features is then changed to 'endElev' and the process is repeated except the merge rule is changed to 'Last.' Setting the merge rule to this ensures that the opposite point touching the line segment is selected for the spatial join.
- The arguments for the second Spatial Join are the result of the first join, the extracted point features, and the Field Mappings object with the updated merge rule. The result of this second join will be a feature class in the "Trails_Processed" geodatabase of line features with fields 'startElev' and 'endElev' which describe the elevation at the start and end of each line segment, respectively.
- All these steps are completed for each feature class in the "Trails_Working" geodatabase and saved to the "Trails_Processed" geodatabase.

21. When this script is complete, close the Python Shell and IDLE windows.
22. Reopen ArcCatalog and navigate to the "Trails_Processed" geodatabase.
23. Open ArcToolbox and open the Merge tool.
24. In the tool window, select every feature class in the "Trails_Processed" geodatabase and set the output as "Trails_Seg" inside of the initial "Trails_Project" geodatabase. This process combines all of the separate feature classes into a single feature class.
25. Find the Add Field tool within ArcToolbox, right click on the tool, and click "Batch" to run the Add Field tool multiple times in succession.
26. In this window, create seven rows that have the "Trails_Seg" as the input table. Then each row to create each of the following fields: 'feetElev,' 'aveElev,' 'Slope,' 'slopeDiff,' 'elevDiff,' 'sumDiff,' and 'segRating' with the data type 'SHORT,' 'SHORT,' 'FLOAT,' 'SHORT,' 'SHORT,' 'SHORT,' and 'TEXT' respectively. Run the batch if the table resembles Figure 9.
27. Once these fields are created, open the Add Geometry Attributes tool in ArcToolbox. Run the tool for "Trails_Seg" to calculate the length of each line segment in meters. This will create another field in the attribute table titled 'LENGTH_GEO.' It is very important



	Input Table	Field Name	Data Type
1	C:\Users\Chris\Documents\GEOG_358\S	feetElev	SHORT
2	C:\Users\Chris\Documents\GEOG_358\S	aveElev	SHORT
3	C:\Users\Chris\Documents\GEOG_358\S	Slope	FLOAT
4	C:\Users\Chris\Documents\GEOG_358\S	slopeDiff	SHORT
5	C:\Users\Chris\Documents\GEOG_358\S	elevDiff	SHORT
6	C:\Users\Chris\Documents\GEOG_358\S	sumDiff	SHORT
7	C:\Users\Chris\Documents\GEOG_358\S	segRating	TEXT

Figure 9

for the slope calculation that the length is measured in meters.

28. This step can be completed in several different ways: using either the Field Calculator tool in ArcToolbox, the Field Calculator wizard in ArcMap, or with a Python script. For this application, I used a Python script because of the length of the code blocks were easier to work with in a Python IDLE.
 - a. Open another IDLE window.
 - b. Open the Python Shell window.
 - c. Within the shell, import arcpy and initialize a variable with the string to "Trails_Seg." (In the example code it is finalTrails)
 - d. Enter in the code blocks shown in Figure 10. These are conditional statements within a function that will be used to calculate difficulty thresholds.
 - e. Enter in each of the Calculate Field processes in Figure 11 into the Python Shell to run the tools.
 - f. Once these are complete, our dataset has been fully processed.
 - g. Close the IDLE and shell windows.
29. Switch to ArcCatalog and delete the "Trails_Working" and "Trails_Processed" temporary geodatabases.

```
BLOCK1 = """def slopeDiffCalc(slope):
    if slope >= 0 and slope < 2.86:
        return 1
    elif slope >= 2.86 and slope < 5.71:
        return 2
    elif slope >= 5.71 and slope < 8.53:
        return 3
    elif slope >= 8.53 and slope < 11.31:
        return 4
    elif slope >= 11.31 and slope < 20:
        return 5
    elif slope >= 20 and slope < 90:
        return 6"""

BLOCK2 = """def eleDiffCalc(elev):
    if elev >= 0 and elev < 7000:
        return 0
    elif elev >= 7000 and elev < 8500:
        return 1
    elif elev >= 8500 and elev < 11000:
        return 2
    elif elev >= 11000 and elev < 12500:
        return 3
    elif elev >= 12500 and elev < 20000:
        return 4"""

BLOCK3 = """def totalDiffCalc(num):
    if num >= 0 and num < 3:
        return 'Easy'
    elif num >= 3 and num < 6:
        return 'Moderate'
    elif num >= 6 and num < 8:
        return 'Hard'
    elif num >= 8 and num < 11:
        return 'Very Hard'"""
```

Figure 10

```
arcpy.CalculateField_management(finalTrails, "aveElev",
                                '(!startElev! + !endElev!)/2', 'PYTHON')

arcpy.CalculateField_management(finalTrails, "feetElev",
                                '!aveElev!*3.28084', 'PYTHON')

arcpy.CalculateField_management(finalTrails, "Slope",
                                'math.atan((abs(!endElev!-!startElev!))/!LENGTH_GEO!)*(180/math.pi)',
                                'PYTHON')

arcpy.CalculateField_management(finalTrails, "Slope_Diff",
                                'slopeDiffCalc(!Slope!)',
                                'PYTHON', BLOCK1)

arcpy.CalculateField_management(finalTrails, "Ele_Diff",
                                'eleDiffCalc(!feetElev!)',
                                'PYTHON', BLOCK2)

arcpy.CalculateField_management(finalTrails, "Diff_Sum",
                                '!Slope_Diff! + !Ele_Diff!', 'PYTHON')

arcpy.CalculateField_management(finalTrails, "Seg_Diff",
                                'totalDiffCalc(!Diff_Sum!)',
                                'PYTHON', BLOCK3)
```

Figure 11