Shana Negin
shana.negin@gmail.com
Geog 375 Final Project Documentation
May 15, 2021
Professor Jennings

"Time in Transit to My School"

Purpose, Goals and Summary

In Sacramento, students receive a free transit pass so they can use public transportation to travel to school.

I imagined a **map series** where a high schooler could **quickly determine how long it would take to get to school from anywhere within the school district.**

My goal was to produce a map series, containing a map for each high school in the Sacramento City Unified School District. Each map will show when someone who lives within the SCUSD boundary will have to leave to get to school by 730 am, in the form of a service area unique values map.

I used a combination of ArcGIS Pro and Arcpy to automate the creation of this map series. I used tools such has the Network Dataset, Service Area Analysis and UniqueValueRenderer to create a series of map PDFs intended for public use. My intention is that these maps could be used to help families make decisions about schools in Sacramento.

Description of Geoprocessing Tasks

ArcGIS Pro: Initial Tutorial:

https://pro.arcgis.com/en/pro-app/latest/help/analysis/networks/create-and-use-a-network-dataset-with-public-transit-data.htm is the tutorial I used to create the initial network dataset and sample service analysis

I used a GTFS dataset, which stands for General Transit Feed Specification. It is a common format for public transportation information, such as schedules and stops. Sacramento Regional Transit publishes a well formatted GTFS.

I had to manually create certain fields in order for the SacRT GTFS to be completely compliant with the tutorial example. I did this instead of using ESRI's sample data in order to dive right into my project.

The rest of the tutorial was relatively straight forward, as long as I kept track of where I was.

Python: For each school, I performed these geoprocessing tasks:

Service Area:

```
arcpy.nax.MakeNetworkDatasetLayer()
```

Creates a Dataset LAYER, which is a different thing than a dataset. It stores the edges, junctions and other pieces of the network dataset. It is a temporary construct, and used because it is less expensive from a processing standpoint, than to refer to the dataset directly.

Setting Properties for Service Area:

When you create a service area in ArcGIS Pro, you set these same properties using the GUI. So, when I went in to write the code, since I'd already created a service area through the GUI, I knew exactly what these were referring to.

Several Service Area properties are available. Some I used functions to get the same setting as my template, and others I hard coded.

```
arcpy.nax.GetTravelModes()
```

This function was used in creating a Service Area. It allows you to make sure that the travel mode you set in your network dataset is the one you use in the service area analysis property travelMode.

These are all properties I needed for determining the service area

```
service_area.timeUnits = arcpy.nax.TimeUnits.Minutes
service_area.travelDirection =
arcpy.nax.TravelDirection.ToFacility
service_area.defaultImpedanceCutoffs = [20, 45, 70, 95, 120]
service_area.timeOfDay = datetime(year=2021, month=5, day=6,
hour=7, minute=30) #datetime object for 0730 05.06.2021 (arrive
by)
service_area.travelMode = travel_modes
service_area.outputType =
arcpy.nax.ServiceAreaOutputType.Polygons
service_area.geometryAtOverlap =
arcpy.nax.ServiceAreaOverlapGeometry.Dissolve
sr = arcpy.Describe(schools_fc_Path).spatialReference
```

Inserting Facility into Service area:

In my use case, each service area had exactly ONE facility – the school for which I was creating the service area. The process of inserting the facility involves getting the fields from a feature class which contains a point for each school, the schools name, and its location as xy data. The insert fields can only be called 'Name' and then reference a shape token such as SHAPE@XY. They cannot be called anything else.

```
insertFields = ['Name', 'SHAPE@XY']
with
service_area.insertCursor(arcpy.nax.ServiceAreaInputDataType.Faci
lities, insertFields) as cur:
     cur.insertRow([sNameClean, sLoc[0] ])
del cur
```

is all it took to insert the facility into the Service Area. However, these 4 lines took a GOOD chunk of my time.

Solving the Service Area Analysis:

```
result = service_area.solve()
```

Actually "solves" the service area analysis, effectively the command that creates the completed Service Area

Layout

Symbology and Labels

For the Service Area Polygons, I wanted to use a specific colorramp, reversed. Arcpy's control of symbology is somewhat limited, so I had to extract the individual colors and then hardcode a list of colors to assign to the values.

```
            myColorramp = [{'RGB': [254, 232, 37, 100]}, {'RGB': [93, 201,
            99, 100]},{'RGB': [33, 145, 141, 100]},{'RGB': [59, 82, 140,
            100]}, {'RGB': [68, 1, 89, 100]}
                        ]
            maplist.addDataFromPath (output_polygons)
            for lyr in maplist.listLayers("ServiceArea*"):
                  if (output_polygons == lyr.dataSource):
                        sym = lyr.symbology
                        sym.updateRenderer('UniqueValueRenderer')
                        sym.renderer.breakCount = 5
                        for g in sym.renderer.groups:
                              for c, i in enumerate(g.items):
                                    i.symbol.color = myColorramp[c]
                        lyr.symbology = sym
```

For the Facility (School) layer, I wanted to use a style that I saved in a gallery, so I imported it

```
            # School Symbology and Layers
            lyr = maplist.listLayers('The_School*')[0]
            if (the_school_fc_Path == lyr.dataSource):
                  # symbology to show school as a star
                  sym = lyr.symbology
                  sym.renderer.symbol.applySymbolFromGallery('Red Star White
            Fill')
                  sym.renderer.label = sName
                  lyr.symbology = sym
```

Labelling the facility with the school name

```
            qry = """"CleanName" = '""" + sNameClean + """'"""
            arcpy.management.SelectLayerByAttribute(lyr, 'NEW_SELECTION',
            qry, 'INVERT'  )
            if int(arcpy.GetCount_management(lyr)[0]) > 0:
                  arcpy.DeleteFeatures_management(lyr)
            lyr.showLabels = TrueText Elements
            tElements = layout.listElements("TEXT_ELEMENT")
                  for t in tElements:
                        if t.name == 'School Name':
                              t.text = 'School Name: ' + sName
```

Save as PDF

```
            PDF_DIR = Path.joinpath(root, 'TurnIn', 'MapSeries')
            mapname = sNameClean + '_map.pdf'
            mapfilename = Path.joinpath(PDF_DIR, mapname)
            if mapfilename.exists():
                  mapfilename.unlink()

            if arcpy.Exists(mapfilename):
                  arcpy.Delete_management(mapfilename)
            layout.exportToPDF(mapfilename)
```

Summary of difficulties and resolution

GTFS data not QUITE in right format

I added columns and created data for if a transit stop needed wheelchair access, to classify the streets, and a few other modifications. ESRI explains all this in the tutorial, but it was tricky at first.

Understanding relationship between database files and ArcGIS Projects

At one point, I deleted my entire service area template because I thought I had backed it up in when I backed up my ARPX file.

Getting Facility inserted into Service Area

ESRI did not explicitly state that the fields HAD to be named 'Name' and then the shape token (@ShapeXY, in this case). I was trying to name the field and add other fields and it was not working, until I whittled it down to just 'Name' and '@ShapeXY'

Adding just the right symbology for Service Area Polygons

I wanted a reverse colorramp. Arcpy symbology is somewhat limited, so I ended up setting each color manually.

Next Steps

1. Improve Layout – add overlay of transit stops and streets
2. Improve Layout – cleaner custom base map
3. Optimize code – move certain steps outside of the loops.
4. Optimize code – do more checking if things are what they see
5. Optimize code – clean up memory more, delete more temporary objects explicitly