Web GIS Exercise #2

Web Map Services

WMS *stands for "Web Map Service." Developed by the Open Geospatial Consortium (**OGC**), this is a standard protocol for serving georeferenced images over the world wide web. When you build a base map from layers or georeference an image that you'd like to use online, irrespective of whatever specific framework or format you use to publish the map service, it will typically be published with WMS and/or **WMTS** support (WMTS stands for "web map tile service" protocol). Two of the most popular ways that organizations manage and publish hosted web map services for base maps is through ArcGIS for Server and GeoServer.*

- *Here is an instance of a GeoServer set up (at the University of Nebraska Libraries):* https://cors1601.unl.edu/geoserver/web/?wicket:bookmarkablePage=:org.geoserver.web.demo.MapPreviewPage. The "boschke1861" layer on that list, for example, is used as the default base map in this map application: http://civilwardc.org/maps/explore/
- *Here is an example of an ArcGIS for Server instance (used for the City of Lincoln's GIS apps):* https://gisext.lincoln.ne.gov/arcgis/rest/services. When you use the city's GIS Viewer https://maps.lincoln.ne.gov/default/index.html?viewer=GISViewer, many of the background services and images originate from the city's ArcGIS for Server instance.

*While all web map services draw data onto a map, WMS standards are not designed for querying information. Rather, Web Feature Service (**WFS**) protocols enable users to extract underlying attributes of published map layers. Consequently, you will most often see WFS used for publishing vector datasets when the intent is to allow user access to the feature data themselves and WMS or WMTS used when you simply want to serve the map layers without enabling users to query the information.*

*Related to these standards, you will often encounter the concept of "map **tiles**." Map tiles provide an avenue for delivering web maps online without the need to redraw an entire map image – or redraw all map layers – on the fly as a user zooms or pans within a window. By segmenting the map into rectangular "tiles," only those pieces needed for an individual user interaction can be loaded, reducing the strain on the server side from which the map layers are hosted and delivered and thereby improving load times for users. Published web map images are divided up into an increasing number of tiles based on increasing "zoom" levels, in order to minimize the number of higher resolution tiles needed in any particular instance. Think of this as a pyramid structure: a lower zoom level (zoomed out; smaller map scale) does not require as much map detail, so a map service at a small scale can be delivered by loading just a few tiles or one single tile, whereas a higher zoom level (larger map scale) will need many more tiles to provide an adequate level of detail in the map, however a user also will not need to load as many of these tiles at any time. Historically, **raster tiles** were commonly used for this purpose, and they are still widely used. Cartographers can effectively use as many*

layers as they want this way and style the map exactly as they want it to appear at various levels, and then effectively take a "snapshot" of the map at various scales, for each tile. When a user pans or zooms into some spot on the map, they are essentially calling up just those pictures relevant to that area. More recently, with improvements in computer technology, ***vector tiles*** *are growing in popularity. Vector tiles draw from an underlying GIS dataset rather than image caches or "snapshots" of one completed map at various scales. Rather than delivering a map that has been predetermined by the cartographer and completed as one single output, GIS data are delivered and then "styled" on the client side, which means they're more adaptive to the user's specific resolution/screen orientation. Vector tiles typically are also generated based on the density of data at a given point, rather than segmented based solely on zoom level, which means they can be faster to generate.*

The purpose of this lab is to generate a cartographically sound "base map" based on vector data sources. You will be publishing the map as raster tiles using standard ArcGIS Desktop software. Note: this only outlines a mainstream method for developing custom base maps. A comparable method, using QGIS, would be to use the "Generate XYZ tiles" option in the raster tools and publish to some other web host, perhaps one running a GeoServer instance. We are using ArcGIS this time in order to connect with the ArcGIS Online options available in the trial period, which enable hosting.

By completing this lab, you will learn the process of developing and publishing customized base map for the web, and you will incorporate hosted services from the previous lab exercise. PLEASE NOTE that I provide instructions in bullet format for general instructions, and sub bullets are there to provide additional guidance for those of you who are less comfortable using desktop software.

- Acquire road data for Lancaster County, Nebraska (all roads)
  - The TIGER (Topologically Integrated Geographic Encoding and Referencing)/Line files of the US Census Bureau makes these accessible at: https://www.census.gov/cgi-bin/geo/shapefiles/index.php
  - Use the form to select "roads" (under Features), and then ensure that you are downloading "ALL ROADS." You will then be prompted to select Lancaster County specifically.
  - The road data will come as a zipped folder. Unzip it.

- Start QGIS and load in the roads shapefile. These are road centerlines, line files that run down the "middle" of roads. *Look at the attribute table: there is a field called RTTYP indicating the types of routes. Keep a mental note of that*.

- Reproject the roads data so that they are in "Web Mercator (Auxiliary Sphere)." Save this file with a different name and get rid of the original file. *Web Mercator is a standard projection used by Google Maps, Esri, Bing Maps, et cetera.*
  - On the top menu, click on "Processing" and make sure that the Processing Toolbox is added. You will probably want to dock this on the right side of the

screen if it floats – just click, drag, and hold until the side panel lights up, then release.
    - Type in "reproject layer" in the search bar. Double-click to open the reproject layer option

Your data are in GCS North American Datum 1983. This is sometimes called "NAD 83" in shorthand, and due to its relative geodetic accuracy for the country, it's often used by civilian government agencies like the Census. Notably, it's an entirely different datum than the popular "Web Mercator" format commercial entities use, so get used to doing this transformation…

    - In the "Reproject Layer" window, click the little globe icon that comes right after the "Target CRS" layer. This is where you can pick a new coordinate system.
    - Use the filter bar at the top to search for "3857." This is EPSG:3857, which is "Web Mercator" in Esri products, but also known as "Pseudo-Mercator" by QGIS.
    - Select the project by clicking on it, then click OK
    - Pick a location to save the projected layer (otherwise it will throw it on an inaccessible drive location if you're in the lab)
    - Give it a save name and then RUN the reproject

- As with Esri products, the software automatically adopts the projection of the first item you add. In any case, ensure that your data frame also matches the Web Mercator projection.
    - In the lower right hand side of the QGIS window, click on the EPSG button
    - Filter once again for EPSG 3857. Select it.
    - Click Apply, and then OK.

- Also, style the roads so that they appear differently based on their route type classification
    - Right-click on the road layer name and select "properties" to open the layer properties
    - Click on the paintbrush icon (third one down on the left hand side of the window) – this is the layer symbology tab
    - Use the drop-down menu to change from single symbol to categorized
    - In the "Column" box, choose the variable you want to use to differentiate changes (in this case the RTTYP field).
    - Click "Classify" in the lower left side of the window.
    - Click Apply and OK when you are satisfied with your choices.
    - Repeat steps as needed.

- Label major routes with a sans serif font. You can choose which roads to label.
    - Return to the layer properties and pick the label icon "abc" just below the symbology icon
    - Use the drop-down menu to choose "rule based labels"
    - Click on that first (and only) item in rule-based labeling "(no filter)" and then click on the edit button (looks like a paper and pencil at the bottom of the window) – this brings up the "Edit Rule" option
    - Change the minimum scale to 1:25,000, keep the maximum as it is
    - Click Apply to return to the rules and then Apply and OK when you're satisfied

- Zoom into the area near downtown and UNL's city campus, just enough so that labels appear

- Return to your processing toolbox (on the right by now, hopefully)

- Type in "tiles" and select "Generate XYZ tiles (Directory)." Double-click to open.

- In the extent option (first on the parameters list), click on the ellipsis that follows the text entry and select "Use Canvas Extent"

- Change minimum zoom to 13 and maximum zoom to 15. You can keep the other parameters for now

- Go down to "Output html (Leaflet)" and click the ellipsis that follows it. Choose "Save to File…"

- Click "Run"

- Navigate to the location where you had saved the tiles. Click to open the HTML file.

- Preview your saved map tiles.

- Now open that same HTML file with either Atom.io or Notepad++ or some other enhanced text editor tool

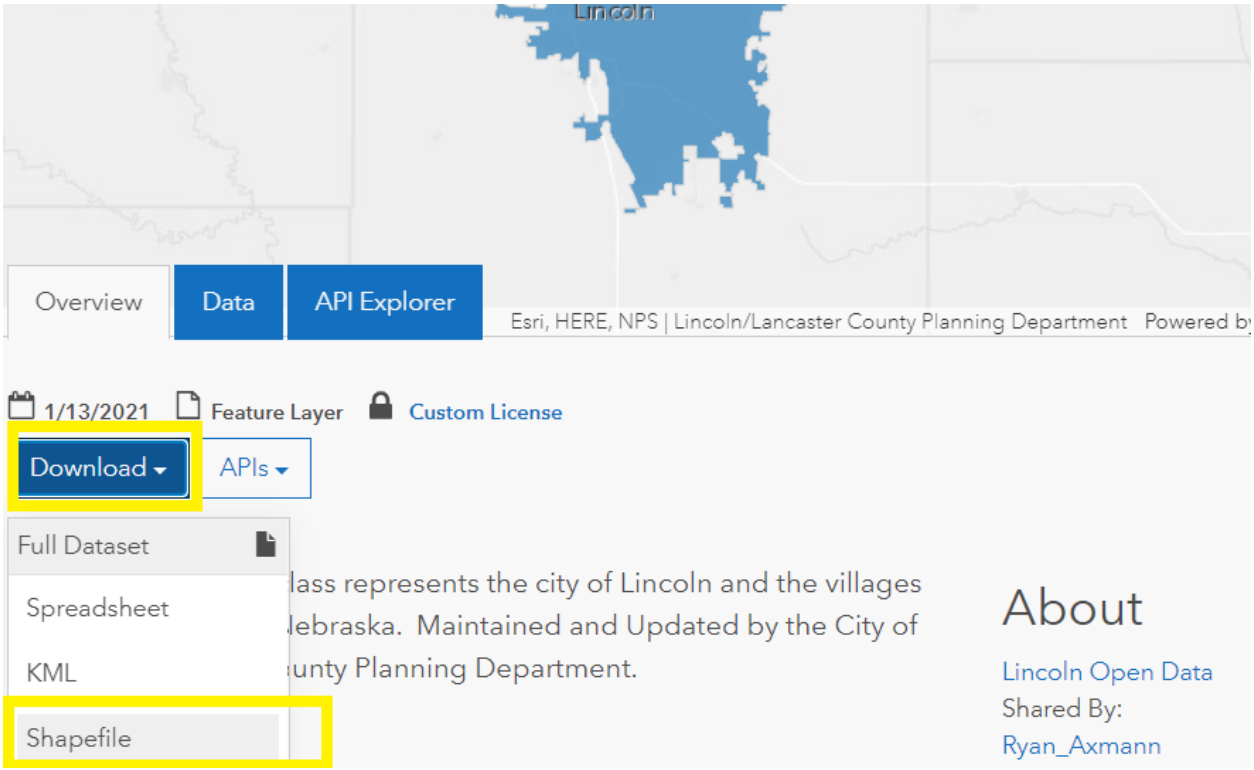
- Notice that you have a Leaflet-style HTML page, essentially. There's an L.tileLayer call to "hosted" tiles which are actually located somewhere on your computer's hard drive.

PART II: Fixing and Hosting the Tile Layer, and Adding Data

If you have been working from a lab computer in Burnett 126, you might have difficulty accessing the tiles. It's also highly likely that your background is gray. Let's fix those things:

- Return to QGIS

- Locate and download a shapefile of the city limits of Lincoln. Go to the city's Open Data Portal and download the file from this location (use the drop down menu to pick the shapefile option) https://opendata.lincoln.ne.gov/datasets/ef26cb9f428b4036af7c138783bffef6_1?geometry=-97.201%2C40.622%2C-96.170%2C40.986



- Add the city boundaries shapefile to your QGIS map

- Using steps learned in the previous lab, change the symbology so that the shapefile of the city has a white fill. (You may not notice the background color difference in your desktop software, but this will make your tiles fill in with white rather than gray after you convert them)

- Return to the Lincoln data portal. We will work through the steps of adding a web map service – but first we need to locate one.

- Search the Transportation category: https://opendata.lincoln.ne.gov/search?tags=transportation

- Find the "Future Trails (Committed)" Layer. You may need to click on "More Results" if it does not appear on your list. Click on Future Trails (Committed) when you locate it.

- Rather than downloading this layer, click on the "API" drop down menu located just to the right of the download options. Copy the GeoService layer. Open it in a new tab or browser window

This takes you to an ArcGIS for Server REST services layer, with a query page opened by default. This is the city and county's Esri GIS server where they store RESTful services for map layers and tiles relevant to city functions. Each service has its own URL so that it can be consumed by web maps. We will explore these services in greater detail in the future, but for now, we are just looking for a web map service (a raster file) of the bicycle and walking trails. If you look at the top of the screen, you will notice the query is nested within the existing pedestrian/bicycle facilities sublayer, which is part of the Trails (MapServer) directory:

Home > services > Planning > Trails (MapServer) > Existing Pedestrian/Bicycle Facilities > query

We JUST want to grab that map service URL. Click on the "Trails (MapServer)" link and then copy and record the URL for the web map service:
https://gisext.lincoln.ne.gov/arcgis/rest/services/Planning/Trails/MapServer

- In QGIS, navigate to your menu bar and go to Layer > Add Layer > Add ArcGIS Map Server Layer. You will create a connection to this web map service.

- In the QGIS data source manager, in the Server Connections menu that opens, click "New" to begin creating a connection.

- In the URL box (second one down) paste in the URL for the MapServer instance. Then give your connection a name, like "lab2"

- Click OK, and then click "Connect" back on the Server Connections menu.

- Notice that the server connections tab shows up very similarly to how it appeared on the server site itself. Click the second item – title 3 – representing all future trails. Then click on Add. You can now close the server connections manager.

- At this point, make any final stylistic changes to the road files one last time, and then zoom in so that most of city campus and downtown Lincoln are within the canvas extent

- return to the "Generate XYZ tiles" process – use the toolbar search feature if you need to find it again.

- Run the Generate XYZ tiles process. Choose minimum zoom 13, maximum zoom 15. Use Canvas Extent. This time, make sure to save an OUTPUT DIRECTORY to a location you can access, such as a folder on your desktop or documents drive. Save the HTML file to the SAME folder location as the tiles. Then click on RUN.

- Keep QGIS open. Double-click the HTML file to open it and preview your map tiles.

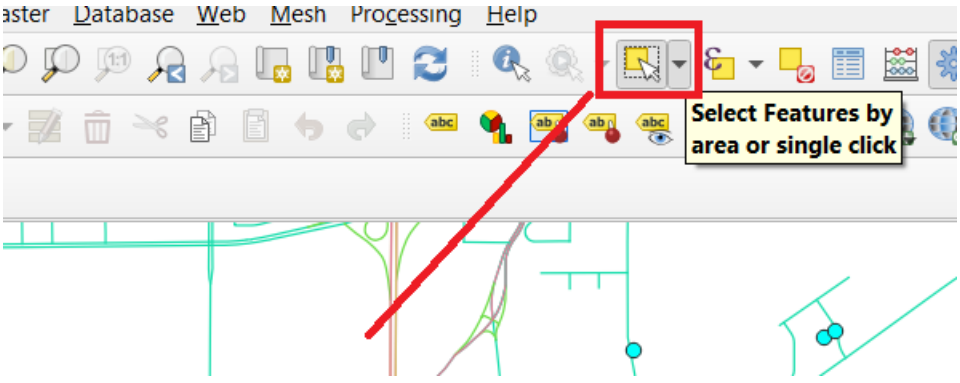
- Open the new HTML file with a text editor. We will add in a GeoJSON layer via HTML this time.

- Instead of locating a GeoJSON layer on the web as we did in previous exercises, this time we will step through developing one and getting it prepared for the web.

- Return to the data portal from the previous steps and navigate to the transportation section once again. Find the bus stop layer - https://opendata.lincoln.ne.gov/datasets/startran-bus-stops

- Download the shapefile and unzip it. Add it to QGIS

- Use the Select Features by Area Tool to select the bus stops in the mapped area you used for your tiles.



- Right-click the bus stop layer in the table of contents/layer list, and click Export > "Save Selected Features As…"

- Change the save format to GeoJSON. Click the ellipsis in the File Name section and choose the same folder to which you'd been saving your other work for this exercise. Keep the WGS84 projection. Save your file.

- Now navigate to the location where you saved that file. Open it with a text editor (Atom or Notepad++ is fine; you could also use Notepad). Your GIS data are saved as a text file! Technically, we could write code to use this file as it is now. Exercise 3 will further explore the concept of adding GeoJSON as variables and saving it into a separate .JS file. The following steps will explore how to build that from scratch.

- First, before any of your text begins, at the very beginning of the file, write:

  var bus =

- Finally, go to the very bottom of your file - after the text - and add a semicolon. That way it will be read into your webpage as if it were any other line of code with a variable (as in Lab 1), and the browser will know that this is the end of the object.

- Save your file as bus.js and make sure that it's in the same directory as your HTML file

- Return to your HTML file and add a new line to your <head> section to load in the GeoJSON dataset:

  <script src="bus.js"></script>

- Also, we'll need to physically add it to the map as a layer. Go just below your "map" variable line (but before the /script) and write in this line to actually *add* the data:

  var bus = new L.geoJson(bus).addTo(map);

- Save your file and open it with a web browser to ensure that it works.

- Rename your HTML file "index.html" and save.

- Now, upload the ENTIRE folder (renaming it to lab2 is ideal) to your GitHub account. This may take a few minutes depending on your zoom level. **You're not done.**

- Once you've uploaded the folder, either return to your index.html file (and then reupload it after this step) OR begin editing the index.html file in that folder on GitHub. **Replace the URL** for your map tiles with the location in which they now live on your GitHub Pages repository.

For example, if you were referencing files at "file:///C:/Users/student/Documents/" originally, and you uploaded them to a folder called "https://student.github.io/lab3/" then you want to swap in the new location accordingly. KEEP the /{z}/{x}/{y}.png structure.

so
file:///C:/Users/student/Documents/Lab2/{z}/{x}/{y}.png
becomes
https://student.github.io/lab2/{z}/{x}/{y}.png

- Resave your index.html page and ensure that it's uploaded/saved to GitHub

- Share the URL!

Rubric

40 points total

- Evidence of **rule-based labeling** in tiles: 30 points
- GeoJSON layer appears: 10 points

Bonus points:

- Custom icon for bus stops: 5 points
- Pop ups for bus stops: 5 points