Open Source Geographic Information Science

GEOG0323 M-W-F 10:10 – 11:00 in BiHall 438 and Th 1:30-4:15 in BiHall 632
Joseph Holler: josephh@middlebury.edu in BiHall 634
office hours: M-Th-F 11:00 – 12:00 or by appointment

Course Description

In this course we will study geographic information science (GIS) with open-source software and critical GIS scholarship. In labs, we will practice techniques to include: data acquisition and preparation for analysis, spatial SQL database queries, automating analysis, spatial interpolation, testing sensitivity to error and uncertainty, and data visualization. We will read and apply critical research of GIS as a subject and with GIS as a methodology. Spatial data sources for labs and independent research projects may include remote sensing, micro-data, smart cities and open government data, and volunteered geographic information (e.g. OpenStreetMap and social media).

Learning Goals

- Survey FOSS4G (Free and Open Source for Geospatial) in terms of its landscape of organizations and projects, research applications, and (radically) unique political economy of knowledge production.
- Expand your functional knowledge of the nature of geographic information with respect to data standards, structures, metadata, provenance, error, and uncertainty.
- Creatively apply FOSS4G to address compelling questions in human geography and problems in social and environmental sustainability.
- Critically reflect on emerging opportunities and ethical dilemmas in open-source geographic information science.
- Learn how to reproduce existing geographic research and to produce geographic research that is open, reproducible and replicable.
- Design and communicate research effectively in multiple media, including digital media, reports, presentations, maps, graphs, tables, data, and code.
- Become competent and confident in conducting research, learning new methods, and overcoming errors, uncertainty, and technical difficulties. Learn to "debug" problems and teach yourself new techniques through structured experimentation.
- Let's be honest, I'm just excited to learn and share new open source GIS technologies with you, expanding our imagination for the possibilities in open source, including future research and careers.

Evaluating Learning

Active participation in the course (**10%**) encompasses meaningful attendance for all class activities and preparation of otherwise ungraded assignments. These will include small assignments to prepare you for labs, class activities and discussions, or clarifications of issues after lab. Most Friday sessions (and a few others) will be discussions of geographic information science literature, and for these I will ask you to prepare by writing a brief reflection in the style of a blog post on your GitHub account. I'll also prompt you to write self-evaluations of your learning, giving us a chance to reflect on what works best and what the greatest needs are for the class as a whole and as individuals. The tradition of knowledge production in open source is *collaborative*, therefore part of active participation is also interaction with peers and posting questions and answers to theoretical and technical issues. We'll establish an organization team for this on GitHub.

Most of our learning will be organized by **practical exercises** spanning two weeks each, in which we integrate concepts and theory with skills we practice in labs and tutorials. Following two weeks of learning & practical exercises, you'll compile the products of our practical assignments into a blog-style page on GitHub, complete with references to relevant reading and any models, maps, code, or visualizations we create. These will typically be due in the next weeks' first class meeting. Cumulatively, these contribute **50%** of the grade.

Finally, by the end of the semester you'll have a chance to revise any of your previous work into a final portfolio. You are also expected to add one new independent open GIS application of your own design and choosing to the portfolio. The application may be very modest but should show evidence of *applying or repurposing a new technique* for spatial analysis and/or visualization. This final digital portfolio contributes **40%** of the grade, of which 20% is the cumulative revision and presentation of previous work and 20% is your new exercise. Throughout the semester, we'll encounter plenty of inspiration for this independent application!

The final week of the semester will be devoted to developing and polishing this portfolio, and it will be due in final form on the first day of finals week.

Expectations

Include Yourself

I aim to design courses with diverse interests, perspectives, identities, and learning preferences & abilities in mind. Beyond our class, the Center for Teaching, Learning & Research has peer and professional tutoring resources available for workload management, writing, speaking, and more! (go/ctlr). In addition, some students may need to confidentially consult Student Accessibility Services to request additional resources and letters of accommodation for successful learning. Middlebury's ADA Coordinator (go/ada) is Jodi Litchfield, at litchfie@middlebury.edu or 802-443-5936. In general, everyone should be conscious of how you learn best, and work actively to transform all of your course materials into forms you understand well.

Required Accounts

The course will require you to use GitHub and Twitter accounts. These are free and necessary for the course.

Meet Deadlines and Communicate

Assignments will be due at the *beginning* of class, unless otherwise noted. Late assignments will not be accepted unless prior arrangements have been made with the professor. If an unanticipated emergency or extenuating circumstance prevents you from completing a major assignment on time, you should be in communication with your professors and commons dean for assistance in modifying deadlines in your courses.

If you have difficulty completing an assignment at the last minute (e.g. the evening before it is due), I strongly suggest pausing for fifteen minutes to write a cover letter for what you have, outlining: 1) what have you accomplished thus far, and 2) what steps would you want to take in order to continue making progress toward your learning goals? You should always have at least this to hand in.

Attend Class

Timely attendance of all class meetings is expected. Life events continue outside of class and sometimes prevent you from attending: if you miss class, please let me know beforehand or as soon as possible afterward.

Deadlines at the Semester's End

The end of the semester can be stressful without planning in advance. Look closely at the expectations each of your courses has at the end of the semester and make a plan for meeting those expectations. Deadlines at the end of the semester usually cannot be changed. Incomplete grades are only permissible in extreme cases in which "a student has done acceptable work in the majority of course requirements or assignments, but cannot complete all course work for reasons of illness, emergency, or legitimate extenuating circumstances."

Anticipated Schedule

1) QGIS Processing Environment and Graphic Modeler
2) QGIS Processing Algorithms and using SQL for dissolving and statistics *
3) SAGA modelling terrain and hydrology; error and interpolating gaps in elevation models
4) Automated SAGA modelling with command-line and Python; uncertainty analysis in GIS modelling *
5) OpenStreetMap data and spatial SQL in PostGIS
6) Informal urban settlement detection/characterization with OpenStreetMap, PostGIS + SAGA *
7) Multi-criteria analysis of hazard vulnerability with QGIS and PostGIS
8) Sensitivity/uncertainty analysis of vulnerability models *
9) RStudio for spatial Twitter data acquisition
10) RStudio for spatial Twitter data analysis & visualization *
11) Interactive web mapping with Leaflet
12) Independent application & portfolio revision *

* Application due by the first class meeting following this week

License

Assignment #1 Survey of FOSS4G for Wednesday Sept 11

- Please create a GitHub account if you do not already have one (its free)
- Send me your GitHub account name
- Survey the presentations & academic articles for three recent FOSS4G Conferences (linked below)
- https://www.int-arch-photogramm-remote-sens-spatial-inf-sci.net/XLII-4-W14/
    - https://2019.foss4g.org/
- https://www.int-arch-photogramm-remote-sens-spatial-inf-sci.net/XLII-4-W8/
    - https://2018.foss4g.org/
- https://www.int-arch-photogramm-remote-sens-spatial-inf-sci.net/XLII-4-W2/
    - https://europe.foss4g.org/2017/
- Select *one* article from one of the three conferences' academic tracks (not mine, please).
- In two paragraphs, please review the article in a style suitable for blogs / podcasts. Be sure to mention the main purpose, significant results/conclusions, the technologies used, and whether you think it mattered that any of the technologies were open source.
- Watch a short video on the virtues of using GitHub for publishing web pages: https://www.youtube.com/watch?v=2MsN8gpT6jY&feature=youtu.be
- Create a repository in your GitHub account
    - Make the repository name identical to ____.github.io (fill in the blank with your GitHub account name. E.g. mine is josephholler.github.io
    - Apply a Jekyll theme of your choice: https://help.github.com/en/articles/adding-a-jekyll-theme-to-your-github-pages-site-with-the-jekyll-theme-chooser
    - Post your review with a link to the original article!
- I should be able to find your review by going to https://______.github.io where the blank is your GitHub user name!

Practical #1: QGIS Modelling

This is due on your GitHub pages by the beginning of class on Wednesday, September 25:

Expectations

Finalize your GitHub model, inclusive of good help documentation. The model should also include a URL link to your GitHub page under 'documentation help URL'. Your post could include an image of the model and illustration of its outputs and application to a different urban area in 2017.

Include a few paragraphs to situate the work you've done in the conversations about open source GIS and/or GIS as a science, referencing readings from the beginning of the course through Friday→Monday readings by Sieber and St. Martin and Wing. This is your space to personalize what GIS and open source mean to you and to communicate that authoritatively through your worked example and references to literature.

In sum, I'm hoping to see a page that has a brief introduction of and link to a completed QGIS model, illustration of how it works with a new case study, a geopackage of data for the case study, and about 3 paragraphs of broadly accessible yet slightly academic discussion, with references. The page should serve both as an illustrative answer to relatives bewildered by your study of geography and evidence for potential employers that you can practice geography & GIS professionally.

Learning Goals

- Orientation to QGIS desktop GIS
- Building and running models/algorithms
- Applying concepts of open science, open GIS, and GIScience to how you conduct and represent your geographic research
- Use OGR to translate data formats
- Use complex field calculations, including geometry functions, nested functions and CASE statements
- Use SQL queries for more advanced geometry functions
- Use DataPlotly to create interactive web-based graph visualizations
- Know how to download and use Census data on your own

Instructions

Add Help webpage to your model

1. Please edit your model's help documentation to add a "Documentation help URL", directing users to your GitHub pages. For example, I entered https://gis4dev.github.io/
2. Once you add this, you should be able to run the model, click the "Help" button at the bottom right, and see QGIS open a web browser to your own GitHub page.

Create data to test your model

1. In QGIS, you can change the file types and coordinate reference systems of data during the process of exporting layers. You can also export only the selected features and remove attribute fields while exporting.
2. Right click a layer and export → save features as…

    a. Save in the format of ESRI Shapefile

    b. Change toe CRS to a different state plane system, e.g. 3684 is the code for one of Vermont's coordinate systems.

3. Try saving the CBD and the tracts in different coordinate systems and as shapefiles and use your model on them.

4. Do any combinations of shapefiles and incorrect projections disappoint you?

Using CASE to classify data

1. In Field Calculator expressions, a CASE statement can be very helpful for calculating fields based on a set of one or more conditions.

2. For example, if a field "pctWhite" contains the percentage of the population that is white, it can be classified into an integer field from 1 to 4 where 1 is the most segregated minority area and 4 is the most isolated white area:

   ```
   CASE
   WHEN "pctWhite" >= 80 THEN 4
   WHEN "pctWhite" >= 50 THEN 3
   WHEN "pctWhite" >= 30 THEN 2
   ELSE 1
   END
   ```

3. Both the WHEN and THEN parts can be more complex expressions, e.g. "White" / "Total" * 100

4. A variety of other Aggregate functions can be helpful for classifying data, e.g.

    a. Min() and Range() can be useful to find breakpoints for equal interval classifications

    b. Q1(), Median(), and Q3(), which can be the break points for a quintile classification (four classes with equal frequency of observations in each)

    c. Mean()

5. Try to use Field Calculator classify the direction data you calculated into N, E, S and W, where each cardinal direction includes a 90 degree range.

    a. E.g. East should be from 45 to 135.

    b. Try visualizing the output using the Categorized symbol type.

6. Add a Field Calculator to your model to calculate and label four or eight directions (your choice)

Apply the transform function in field calculator for finding azimuth

1. For Azimuth calculations, the World Mercator projection does work correctly. As the help window's example indicates, the new coordinate system should be given as a text string including the authority:
   geom_to_wkt(transform($geometry, 'EPSG:2154', 'EPSG:4326')) → POINT(0 51)
   This code is a sample directly from the help file! You don't need to convert to text with geom_to_wkt().

2. The World Mercator projection is EPSG:54004

3. Try to implement coordinate transformation and azimuth calculation in your model with the World Mercator projection. Use inputs with different coordinate systems to verify that it's working correctly.

Use Execute SQL to unlock some of the spatial SQL power from QGIS processing

1. The **Execute SQL** algorithm (under 'Vector general') functions by creating and saving a virtual layer.
    a. https://docs.qgis.org/3.4/en/docs/user_manual/working_with_vector/virtual_layers.html
2. Note that the type of SQL is SpatiaLite, so it means you can use these functions: http://www.gaia-gis.it/gaia-sins/spatialite-sql-4.3.0.html
3. Search for distance(to find the distance functions available: they allow a third parameter of the value TRUE to use ellipsoidal geodesic calculations of distance.
4. Search for transform(to see how this transform function works.
    a. Does it take an integer number for the coordinate reference system, or a text string?
    b. Does it require the source coordinate reference system, or only the destination?
    c. Is there a centroid function?
5. Try opening Execute SQL from the Processing Toolbox and running a series of queries. You can leave the algorithm dialogue open and just modify the parameters. Note how each query changes (or doesn't) the output
    a. Set the input data source to Tracts2010… this can be called input1 in your queries.
    b. SELECT *
       FROM input1
    c. SELECT *
       FROM input1
       WHERE MedGrossRe is not null
    d. SELECT *
       FROM input1
       WHERE MedGrossRe > 1500
    e. SELECT fid, geometry, GEOID10, PopTotal, MedGrossRe
       FROM input1
    f. SELECT fid, geometry, GEOID10, PopTotal, MedGrossRe, Latinx / PopTotal as LatinxRatio
       FROM input1
    g. SELECT fid, geometry, GEOID10, PopTotal, MedGrossRe, 1.0 * Latinx / PopTotal as LatinxRatio
       FROM input1
    h. SELECT fid, geometry, GEOID10, PopTotal, MedGrossRe, st_area(geometry) as tractArea
       FROM input1
    i. SELECT fid, geometry, GEOID10, PopTotal, MedGrossRe, astext(centroid(geometry)) as middle
       FROM input1
    j. Try using st_x() and st_y() to find the x and y coordinates of the center of each tract in a query of your own!

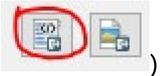
6. Make sure that CBD appears above tracts2010 in your Layers panel. Add a second layer, CBD, as an input. As long as CBD appears before tracts2010 in the list, CBD will be referred to as input1, tracts2010 will now be referred to as input2.
    a. SELECT *, astext(geometry) as geomOriginal, astext(transform(geometry,4326)) as geom4326 FROM input1
    b. SELECT * , distance(centroid(geometry), *(SELECT geometry from input1)*) as cbdDistSQL FROM input2
    c. The italicized section is a subquery that will fetch the point from the CBD layer. It works this simply because the layer only has a single feature.
7. Try combining what you learned in the two queries above to find the distance between tract centroids and the CBD in the 4326 coordinate reference system (WGS 1984). If the transform functions work, your results should be incorrect, measuring in decimal degrees rather than in meters.
8. Try adding TRUE as a third parameter in the distance function: now you should get accurate geodesic measurements!
9. Try replacing your distance field calculator with this Execute SQL algorithm in your model.

Download data for a second case study in urban structure

1. Find census boundary files here: https://www.census.gov/geographies/mapping-files/time-series/geo/carto-boundary-file.html
2. Select census tracts for a whole state or a set of counties covering a geographic extent sufficient to visualize one metropolitan area.
3. Find census attribute data at American FactFinder: https://factfinder.census.gov/
4. Use the Advanced search
5. Set geographies to census tracts and choose a state and county/counties
6. Set the Topics → Dataset to 2017 ACS 5-year estimates
7. Table B25064 has median gross rent
8. Table B03002 has Hispanic or Latino origin by race
9. When you download, don't merge the annotations or use descriptive field names
10. Use the metadata to determine which fields are important to you, focusing on the data estimates rather than on the margins of error.
11. One of the CSV files will contain your data and another will contain code translations for variable names
12. Remember to unzip all of your data (both CSV's and shapefiles) before using them in QGIS
13. Layer → Add Layer → Add Delimited Text Layer is the best way to add CSV files. It tries to detect data types correctly rather than assuming everything is a text string.
    a. Import with no geometry as a comma-delimited file
    b. Always double-check the layer properties → source fields to ensure data types are correct.

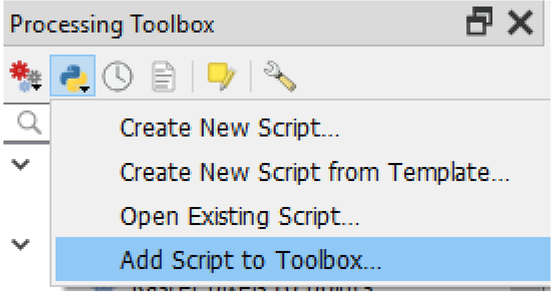
14. Join census data to census tract geographies by opening the properties of the geographic layer and using its Joins menu. For neat results:
    a. Select only the fields you want to join
    b. Make custom filed name prefix blank
15. Once joined, the data should be ready for analysis! (make a CBD on your own with 'centroid' and/or mean coordinates, and if you want a subset of your census tracts, just select them and export selected features)
16. Analyze your chosen case study using your model!

Save a Data Plotly plots as HTML and upload to your GitHub Pages

1. These graphs are dynamic code written for the internet, so let's include scatterplot of distance and polar plot of direction on your GitHub pages!
2. Once you create a plot, save it as HTML (button at bottom-right corner: )
3. Open the HTML file in Notepad++ or Notepad
4. Delete the script tags for polyfill.min.js, e.g. : <script src="file:///C:/Users/josephh/AppData/Roaming/QGIS/QGIS3\profiles\default/python/plugins\DataPlotly\jsscripts/polyfill.min.js"></script> and delete it.
5. Replace the script tags for plotly-1.34.0.min.js with: <script src=https://cdn.plot.ly/plotly-1.34.0.min.js></script>
    a. This is replacing a local version of the javascript for graphing with a link to a website hosting this code. I learned about this by reading about the Plotly library: https://plot.ly/javascript/getting-started/
6. Search for the keywords 'name' and 'title' if you want to change series labels, axis labels, and graph title. Each of the keywords is followed by a value in double quotes, which you can edit.
7. Upload the .html file to your GitHub repository and make a link to it from your post about this model.

Optional: export model to python

1. QGIS algorithms can be further customized by exporting to python scripts and editing the code.
2. Load revised code as an algorithm here in the python menu of the Processing Toolbox:



3. Examples of the python code for all the QGIS algorithms can be seen here: https://github.com/qgis/QGIS/tree/release-3_4/python/plugins/processing/algs/qgis

4. Here is a documentation on creating new scripts: https://docs.qgis.org/3.4/en/docs/user_manual/processing/scripts.html

5. Some easy first revisions to try include re-organizing the order of input parameters and editing parameter names. (Graphic models always order the parameters alphabetically).

6. More advanced revisions would check for duplicate attribute field names and warn the user if their central point features covered too large of an extent compared to their census units.

Optional: automate joining attribute data

1) Since the census tracts geographies and American FactFinder data tables are always formatted in the same way, it's possible to automate joining the tables as part of a model. Use the **Join attributes by field value** algorithm.

Lab Three: Global Digital Elevation Models

Purpose

The practical objectives of this lab are to learn how to find and download digital elevation models for developing countries and how to derive a variety of terrain products from those models using SAGA. SAGA is an open-source desktop GIS developed by German physical geographers and contains an excellent suite of raster analysis tools. We will learn three ways to run SAGA tools: using the GUI, using the command prompt, and using Python scripts. In doing so, we will become real power users of GIS, capable of altering steps in a complex analytical process in order to test for error propagation and uncertainty.

Software

- SAGA Project website: http://www.saga-gis.org
- SAGA Open source code & download: https://sourceforge.net/projects/saga-gis/
- SAGA Tool Reference: http://www.saga-gis.org/saga_tool_doc/6.2.0/index.html

Data

- NASA/METI/AIST/Japan Spacesystems, and U.S./Japan ASTER Science Team. *ASTER Global Digital Elevation Model V003*. 2019, distributed by NASA EOSDIS Land Processes DAAC, https://doi.org/10.5067/ASTER/ASTGTM.003.
- NASA JPL. NASA Shuttle Radar Topography Mission Global 1 arc second. 2013, distributed by NASA EOSDIS Land Processes DAAC**,** https://doi.org/10.5067/MEaSUREs/SRTM/SRTMGL1.003.

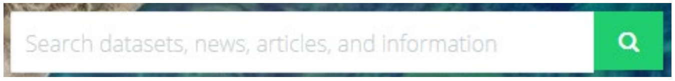
Deliverable

Please choose *either* Aster *or* SRTM as your data source for the lab. Create a new page in your GitHub account to illustrate the steps of for terrain analysis in this lab by creating visualizations of outputs at each step.

Instructions

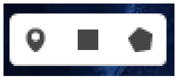
Finding Global Elevation Data

- Go to the **Earth Data Search** website: https://earthdata.nasa.gov/
- In the search datasets tool, look for the relevant global datasets



Search datasets, news, articles, and information

- The relevant datasets are:
  - Aster Global Digital Elevation Model V003
  - NASA Shuttle Radar Topography Mission Global 1 arc second V003
  - NASA Shuttle Radar Topography Mission Global 1 arc second number V003 (error assessment of the SRTM data)
- Once you find the correct dataset, view the results in **Earthdata Search**
  - Zoom to your area of interest
  - Draw a point, bounding box, or polygon around your area of interest to add spatial criteria to your data search



- Now you can view the specific data available for your region of interest and download!
- Please download the two tiles covering Mt. Kilimanjaro… where the tiles each cover 1 degree by 1 degree and are named by the lower left coordinates: S03E037 and S04E037
- You'll have to create an Earthdata account and login:

🔒 Earthdata Login

- Remember to unzip the files once you download them!

The SAGA graphical user interface

- **Start** ⊞ → **SAGA 6.2** to launch SAGA 6.2
- Show or hide windows with the windows menu…
- **Manager** window contains
  - Tools: menu with all of the SAGA algorithms, also found in Geoprocessing menu
  - data: loaded data layers
  - maps: numerous maps can be created in the same project: organize them here
- **Data Sources** window contains browser to the file system or databases
- **Object Properties** window contains all the details for tools and data layers, including parameters, metadata, settings, attributes, legend, and classification/symbology options for data layers. Saving from the properties window saves the *parameters* for a tool or *symbology* for a data layer, but not the data itself.
- **Messages** window contains a log of your work.

Important tips for using SAGA	• SAGA does all of its analysis in a raster format called the SAGA Grid, which is actually a collection of files. • While operations are running in SAGA, a green status bar will show progress. You must be patient for grid analysis! Don't attempt to run additional tools before the current operation is finished. • Running tools in the SAGA interface keeps outputs in working memory, but not permanently saved until you instruct the program to do so! Some tools overwrite existing files unless you specify <create> as output. • SAGA organizes all of your data by *grid system*, i.e. data having the same projection, cell size, and extent • SAGA can import and export from almost any grid format, but SAGA grids themselves are not viewable in QGIS or ArcGIS. QGIS *can* import them with GDAL. • Take care to pay attention to the exact tool parameters for each tool. Even parameters like the input raster(s) and output raster(s) change from tool to tool. Mandatory inputs have a >> symbol, optional inputs have >, and mandatory outputs have << and optional outputs have < • Outputs are generally *temporary* layers. You can save them as permanent grid files by right clicking & save, or SAGA will prompt you to make permanent saves of any temporary grids when you try to close them or exit the program.
Running tools in the SAGA Graphical User Interface: Importing Rasters with GDAL	• Open the **tools tab** of the **Manager** window • Browse to the **Import/Export → GDAL/OGR → Import Raster** • In the **Properties** window, view the settings for the import raster tool ◦ Set **files** to one of the DEM tiles you downloaded ◦ Do not apply a **transformation** ◦ **Apply** (this stores the tool settings for next time!) ◦ **Execute** (this runs the tool) • Go to the **Manager window → Data tab** ◦ Select the grid you just imported ◦ View its settings, description, & properties int eh properties window ◦ Double click the grid to add it to a map
Explore the SAGA interface and manipulate map layers	• Return to the **import raster** tool and try to import the other three girds and add them to the same map ◦ In the **Maps** tab, both grids should appear under the same map. ◦ Try reordering the layers (click and drag) ◦ Try making the layers transparent (double click) • Place the cursor over your map and notice the status bar at the bottom of the window: ◦ `X` and `Y` coordinates display position in the grids' spatial reference system ◦ `Z` is the value stored in the raster at each location *for the highlighted map layer*

`Mosaic the four grids into one grid`	• Open the **Grid** → **Gridding** → **Mosaicking** tool ◦ Tool id: `grid_tools 3` • >> Grids: add both grid DEM tiles • Name: `MosaicAster` *or* `MosaicSRTM` (whichever is appropriate to your data!) • Data Storage Type: same as first grid in list • Resampling: Bilinear Interpolation (because this is continuous quantitative data) • Overlapping Areas: Maximum • Match: None • Target Grid System: (these parameters fill in automatically when you add grids) ◦ Cellsize: `0.000277778` • Use the option of limiting the final size / extent of the grids by changing the left, right, bottom, and top parameters: ◦ Left: `37` ◦ Right: `37.7` ◦ Bottom: `-3.5` ◦ Top: `-2.7`
`Project the grid to the correct UTM zone`	• Open Tools → Projection → Proj.4 → UTM Projection (Grid) ◦ Tool id: `pj_proj4 24` • >> Grid System: choose the grid system with the largest extent • >> Source: `01. Mosaic…` • Resampling: `Bilinear  Interpolation` • Preserve Data Type: ☑ • Target Grid System: change cell size to `30`. Otherwise, use defaults • Zone: `37` • South: South ☑
`Create a Hillshade visualization`	• Open Tools → Terrain Analysis → Lighting, Visibility → Analytical Hillshading ◦ Tool id: `ta_lighting 0` • >> Elevation Grid system begins with 30, named 01. Mosaic… • << Analytical Hillshading: `<create>` • Shading method: standard • Sun's position: azimuth and height • Azimuth: `315` • Height: `45` • Exaggeration: `1` (because both the projection and elevations are measured in the same units, meters, we don't need to adjust interpretation of height) • Unit: radians • To experiment with how this works, try adjusting the azimuth (compass direction) and height (angle above the horizon) to see the landscape shaded at sunrise: lower in the east) • Save your SAGA project and any unsaved grids.

Detect Sinks and determine flow through them, so that hydrological analysis doesn't get stuck in either real holes or holes created by data errors	• Open Tools → Terrain Analysis → Preprocessing → Sink Drainage Route Detection ◦ Tool id: `ta_preprocessor 1` • >> Elevation Grid system begins with 30, named 01. Mosaic… • << Sink Route: <create> • Threshold: ☐ (default) • Threshold Height: 100 (default) • This operation results in a grid with data on which direction water should flow when it encounters sinks. • This operation is difficult and time-consuming! Patience… • Remember to save your project & grid(s)
Remove Sinks from the DEM by filling them	• Open Tools → Terrain Analysis → Preprocessing → Sink Removal ◦ Tool id: `ta_preprocessor 2` • >> Elevation Grid system begins with 30, named 01. Mosaic… • > Sink Route : Sink Route • < Preprocessed DEM <create> • Method: Fill Sinks • Threshold: ☐ (default) • Threshold Height: 100 (default) • This operation results in a new elevation model with sinks filled, for purposes of hydrological analysis. • This operation is difficult and time-consuming! Patience… • Remember to save your project & grid(s)
Where will water go? Calculate flow accumulation.	• Open Tools → Terrain Analysis → Hydrology → Flow Accumulation (Top-Down) ◦ Tool id: `ta_hydrology 0` • >> Elevation: Sink-filled DEM • >> Sink Routes : Sink Route • << Flow Accumulation: <create> • Other input/output grids are not necessary • Step: 1 • Flow Accumulation Unit: number of cells • Method: Multiple Flow Direction • Thresholded minimum flow: ☑ • Linear Minimum: 500 • Convergence: 1.1 • The values in this output indicate how many locations contribute water flow to each cell. Values of 1 mean no other location flows into that cell. Value of 100 would mean water from 99 other cells flow into that cell. Once enough flow accumulates, we assume that a stream has formed.

`Where are the streams? Channel Network`	• Open Tools → Terrain Analysis → Channels → Channel Network ◦ Tool id: `ta_channels 0` • >> Elevation: sink-filled DEM • << Channel Network: <create> • << Channel Route: <create> • << Channel Network (Shapes): <create> • >> Initialization Grid: Flow Accumulation • Initialization Type: Greater Than • Initialization Threshold: 1000 (these two parameters start channels where flow accumulation is > 1000 cells) • Minimum Segment Length: 10 (default) • This tool will create both a raster and a vector version of your streams and rivers!

Lab Four: Model Error Propagation and Uncertainty

Purpose

This lab builds upon the previous exercise, in which we used SAGA to model hydrology and stream channels using digital elevation models from SRTM or Aster. We will take that exercise further at a conceptual level to drill into sources of error and compare elevation models and their resulting hydrological models for error, error propagation, and uncertainty. At an operational level, we'll learn how to create batch processing algorithms for SAGA tools in order to automate processing tasks. We'll also learn a few new tools in SAGA for reclassifying data, masking one raster layer with another, and interpolating unknown data points with surrounding known points, and comparing outputs of various models by calculating the mean differences.

Software

- SAGA Project website: http://www.saga-gis.org
- SAGA Open source code & download: https://sourceforge.net/projects/saga-gis/
- SAGA Tool Reference: http://www.saga-gis.org/saga_tool_doc/6.2.0/index.html

Data

- NASA/METI/AIST/Japan Spacesystems, and U.S./Japan ASTER Science Team. *ASTER Global Digital Elevation Model V003*. 2019, distributed by NASA EOSDIS Land Processes DAAC, https://doi.org/10.5067/ASTER/ASTGTM.003.
- NASA JPL. NASA Shuttle Radar Topography Mission Global 1 arc second. 2013, distributed by NASA EOSDIS Land Processes DAAC**,** https://doi.org/10.5067/MEaSUREs/SRTM/SRTMGL1.003, and it's number grids: https://lpdaac.usgs.gov/products/srtmgl1nv003/

References

- Batch scripting tutorial: https://www.tutorialspoint.com/batch_script/index.htm

Deliverable

As a deliverable, please update your week three GitHub page. In week three, you have explained and illustrated a single iteration of hydrological modeling. In this lab, you will present the results from two iterations of hydrological analysis: one for Aster and one for SRTM. Write up a description of your analysis, including how to automate hydrological analysis and compare the results of the same model with different inputs. Include your batch scripts and illustrations of your findings. Cite the data sources, software used (including version), and any literature supporting your observations. In your explanations, please state which data source is preferable, why it is preferable, and your recommendations / cautions for its use.

Although it is not required, feel free to switch to a different region of the world, as long as it is mountainous and not located in the United States or Canada.

Instructions

`Acquire elevation data and number data for both SRTM and ASTER for Mt Kilimanjaro`	• You probably have at least one of these datasets already, from lab 3. • Go to the **Earth Data Search** website: https://earthdata.nasa.gov/ • Download the additional data. • Remember that Aster data includes elevation and number information in a single zip file. However, SRTM has two separate data series, one for elevation and one for the number information. Also, remember to use the 1-second SRTM resolution. • Unzip all the data, and do not use spaces in any of your folder or file names.
`Convert SRTM number data into usable SAGA grids`	• Open the `srtmNUMtoSAGA.py` file in Idle (right-click and Edit with Idle) • Near the top, change the `WORKDIR` from `"W:\where_are_your_num_files"` to the folder in which your SRTM `.num` files are stored. Don't include a final backslash. • Save the changes. • Go to **Run → Run Module**, or press F5 on the keyboard. • Be patient, and don't close any of the black popup command line windows! • The python script should create a set of saga grid files inside a `sgrd` folder within the same folder as your SRTM `num` files. It's automatically running a set of three steps: ◦ Import a binary file as a SAGA grid ◦ Assign the WGS 1984 geographic coordinate system to the grid ◦ Mirror the grid (because they import backwards in this version of SAGA) • For those of you who know Python, this script is essentially building a set of SAGA commands to be run by the Windows command shell. It allows dynamic customization of the commands, e.g. by identifying the lower left coordinates of each graticule by reading them from the graticule's file name.
`Try running the SAGA program from the Windows command prompt`	• **Start** ⊞ **→ Command Prompt** to launch the Windows Command Prompt • Change to the SAGA program directory by typing: `cd c:\saga6` • Press enter! The new prompt should be: `C:\SAGA6>` • Try to run the SAGA command line program by typing: `saga_cmd` • It may take a moment for the program to list all of the tool libraries. • Try to list all of the tools in a specific library, grid tools, by typing: `saga_cmd grid_tools` • Try to see the options for a specific tool, Mosaicking, by typing: `saga_cmd grid_tools 3` • The real power of using the command prompt isn't doing all your work by typing commands: it's in creating batch scripts to automate data management and analysis tasks. • Type `exit` to close the command prompt.

`Examine a batch script for mosaicking and projecting grids`

- Open the `mosaic_utmproj.bat` file in **Notepad ++**
- A `.bat` file is a batch processing file, which can run a series of commands in Windows' command shell. Essentially, each SAGA tool can be run as a command, so you can batch script a series of tools.
- Lines beginning with colons `::` are comments, and are not run.
- The first three non-comment lines set up file paths and names for the script.
  - ◦ `PATH` allows the script to find the SAGA program
  - ◦ `pre` allows you to add a prefix to folder and file names in your outputs, so that you can run the script multiple times with different sets of outputs.
  - ◦ `od` allows you to set the folder to which you want outputs saved.
- line sets up a path to the Saga6 program folder, so that when we enter `saga_cmd` later, it will be able to find that program.

```
SET PATH=%PATH%;c:\saga6
```

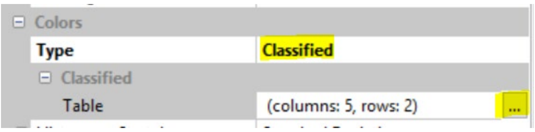
- The second line runs the **mosaicking** tool.
  - ◦ All of the options in the SAGA graphic user interface are available as variables in the command line as well. You can see details of the options and possible values in the **Description** tab of the **Properties** for each tool, or online at http://www.saga-gis.org/saga_tool_doc/6.4.0/
- Using either the SAGA description tab or online help, answer the following:
  - ◦ What is the tool library and tool number for Mosaicking?
  - ◦ What are the options for Resampling in this tool, and what numeric code is used for each?
  - ◦ What code should you use for qualitative data, like the numeric quality grids?
  - ◦ What code should you use for quantitative data, like the elevation grids?
  - ◦ Check these answers before you use the tool!

`Edit the batch script to mosaic and project your data`

- Starting with the SRTM num grids, let’s modify mosaicking.bat to do the mosaicking and projecting work for us.
- Two colons `::` indicate a comment line, which is not run, but leaves some notes about what is supposed to happen.
- Edit the prefix by which you want to identify your outputs for this script, modifying the line `SET pre=Prefix`
  - Hereafter, `%pre%` will be replaced with the prefix you just entered, so that you can organize your outputs. Remember that we’ll run this script numerous times, so create a short and sensible prefix to identify each one.
- Edit the directory in which you want to save your outputs by modifying the line `SET od=W:\lab4\data`
  - Hereafter, `%od%` will be replaced with the directory you just entered, so that you can organize your outputs. Remember that we’ll run this script numerous times, so create a sensible unique folder for each set of data.
- Edit the **Mosaicking** command, including:
  - Edit the `–GRIDS` option to list the srtm number grids, separated by semicolons as in the given example.
  - Edit the `–RESAMPLING` option to use the appropriate resampling technique for qualitative data
- Edit the **UTM Projection** command, including:
  - Edit the `–RESAMPLING` option to use the appropriate resampling technique for qualitative data
  - Edit the `–UTM_ZONE` option to zone `37`
  - Edit the `–UTM_SOUTH` option to `1`
(`0` typically indicates false, while `1` indicates true)

`Run the batch script`

- Save the .bat file *in the same folder as your num grids*.
- Run the bat file from the same folder as your num grids.
(double-click, and accept the risks!)
- It’s important to run the batch script from here, because we have lazily referred to the grids by name rather than by their full address on the `W:` drive, so the script will look for them in the same folder from which the script is saved and run.
- See if a new mosaicked and projected grid has been created, and load it into SAGA to see if it worked!

`Visualize the Number grids`	• For your analysis of error, you'll need to visualize the number grids. For this, be sure to check the documentation for SRTM and Aster so that you know what the values mean. From there, you can categorize the grid visualization. • Change the grids color **Settings** to **Classified**. • Open the ellipsis (…) for the classification **Table**.  Colors Type: Classified Classified Table: (columns: 5, rows: 2) ... • Edit the colors, names, minimum, and maximum values of the first two classes to capture the first two number types you want to visualize. The minimum value is *inclusive* (i.e. >=), but the maximum is *exclusive* (i.e. <), so add one to all of your maximum numbers. • Use the Add button to add more classes to capture additional number types. • Okay to finish. • Save the project, and visualize the Aster data later. The numbers *are not the same* for Aster. You'll find the numbering systems in each datasets' documentation, linked in the data section of the lab.
`Create projected mosaics for all 4 datasets`	• Try modifying the script to mosaic and project each of your four datasets: SRTM Num (complete), SRTM Elevation, Aster NUM, and Aster Elevation.
`Does your region contain oceans or a large lake?`	• If your region contains any oceans or a very large body of water, it may be impossible for SAGA to resolve the flow direction algorithm. To fix this, we can reclassify all elevations of zero or below to *nodata*. If your region's lowest elevations are a large lake, you could reclassify all elevations at or below the lake. • In SAGA, go to the **Grid → Tools → Reclassify Grid Values** tool ◦ Set your DEM as the input grid ◦ Set Reclassified grid to <create> ◦ Set the Method to **single** ◦ Set the Old Value to `0` ◦ Set the new value to `0` ◦ Set the operator to **<=** ◦ Under NoData Output Grid, **Assign…** a **user defined NoDataValue** of `0` • The ocean should now appear as NoData, so that it will not confuse hydrological analysis.

`Create a batch script to automate the hydrology analysis from the previous lab`	• Save a new copy of the batch script so that you can modify it to replicate the analysis from the previous lab. • Add a comment to the top of the script with your name and a description of the scripts' purpose. Batch script comments begin each line with `::` • Replace the Mosaicking and UTM Projection commands with new SAGA tools! • Remember to insert `%pre%` wherever you want to include the prefix in a name, output file name, or input file name and `%od%` wherever you need the output directory. Follow my example script as a model for this. • There's a quick way to start generating new script commands! ◦ Navigate to a tool in SAGA and enter in some of the in the tool's **properties** window. If you open your previous SAGA project, your previous options may even be saved there. ◦ Right-click the tool and **Copy to Clipboard**. Select the **Command Line** option. ◦ Paste the command into your script, and make any necessary changes! ◦ Try this for the **Hillshade** tool. • Save the script in a folder containing your input data, and try running it! • Once you see that Hillshade works, start adding in the other tools until you reach the end: stream channels. Remember, don't use Hillshade as an input for anything else: it's just a visualization not meant for analysis.
`Use batch scripts to create two versions of the model outputs`	• Now that you have a batch script, you can easily experiment with changing inputs or changing any of the script parameters! • Take advantage of this to create two versions of your hydrology analysis: one with SRTM inputs and one with Aster inputs.
`Visualize the differences between two outputs`	• In SAGA, use the **Grid → Calculus → Grid Difference** tool to subtract one set of input **elevation models** from the other, and one set of **Flow Accumulation** from the other.
`Visualize results in 2.5D!`	• If you're visualizing an elevation dataset, you can colorize it however you like and then add a *shading* option to the layer properties. • To best view continuous / quantitative data, change the display resampling to bilinear interpolation. • To best view flow accumulation, I suggest using graduated colors, changing the maximum of the value range to a lower number, e.g. 1000, and changing the scaling mode from Linear to Logarithmic (up) • Hint: there is also a *topography* color preset.

`Visualize results in 3D!`	• Once you have a map of results, it's really easy to render the whole map in three dimensional in SAGA: • Hit the 3D button while you're viewing a completed two-dimensional map. • Assign the appropriate elevation data set, and consider lowering the resolution to the same as the elevation data. • Usually three-dimensional maps use **exaggeration** elevation by a factor somewhere between 1.5 and 2.0. Experiment this to see what you like. A factor of 1 somehow looks too flat, even though it is accurate. • Other defaults are ok.
`Examine resulting streams in QGIS`	• To help consider the causes and implications of data error in the elevation models *vis a vis* the resulting stream network, open **QGIS**. • Add the **QuickMapServices** plugin to QGIS (**plugins → manage and install plugins**) • Go to **Web → QuickMapServices → Search QMS** ◦ A Search QMS window opens. Try searching for `Google Satellite Hybrid` and adding it to QGIS. • Load the shapefiles from your channel analysis into QGIS. • Zoom into areas that you know are problematic from your visualization of the number (quality) tiles, from comparison of the two data models, and from hillshades. • Create some graphics to visualize what you think are the main causes and consequences of DEM errors in hydrological analysis.
`Independent project possibilities`	• Extend this analysis to include catchment / watershed delineation and a few other forms of Terrain Analysis, for a different region.
`Experiment with Interpolation and its uncertainties`	• This is optional if everything else is in working order! • What if you find all options for inputs unsatisfactory and you want to exclude a particular source of elevation and replace it with your own interpolation? ◦ Set **No Data** for a number grid in SAGA to create a mask excluding unwanted data areas (e.g. areas with an unsatisfactory data source), and Apply ◦ Then use the **Grid → Tools → Grid Masking** tool (found in Grid → Tools) to erase those areas from the digital elevation model. ◦ Subsequently, add a **Grid → Tools → Close Gaps with Stepwise Resampling** function into your workflow to fill the holes you just masked into your digital elevation model. Options include the three principal interpolation techniques: i) Nearest neighbor ii) Bilinear interpolation iii) Bicubic Spline • Run a new hydrological analysis using this interpolated grid as the input, and compare to previous results.

Pre-Lab Five: Recall you already started learning SQL…

1) Remember these queries you practiced in Lab Two with the QGIS **Execute SQL** tool:

    a) SELECT *
    FROM input1

    b) SELECT *
    FROM input1
    WHERE MedGrossRe is not null

    c) SELECT *
    FROM input1
    WHERE MedGrossRe > 1500

    d) SELECT fid, geometry, GEOID10, PopTotal, MedGrossRe
    FROM input1

    e) SELECT fid, geometry, GEOID10, PopTotal, MedGrossRe, Latinx / PopTotal as LatinxRatio
    FROM input1

    f) SELECT fid, geometry, GEOID10, PopTotal, MedGrossRe, 1.0 * Latinx / PopTotal as LatinxRatio
    FROM input1

    g) SELECT fid, geometry, GEOID10, PopTotal, MedGrossRe, st_area(geometry) as tractArea
    FROM input1

    h) SELECT fid, geometry, GEOID10, PopTotal, MedGrossRe, astext(centroid(geometry)) as middle
    FROM input1

    i) Try using st_x() and st_y() to find the x and y coordinates of the center of each tract in a query of your own!

2) Make sure that CBD appears above tracts2010 in your Layers panel. Add a second layer, CBD, as an input. As long as CBD appears before tracts2010 in the list, CBD will be referred to as input1, tracts2010 will now be referred to as input2.

    a) SELECT *, astext(geometry) as geomOriginal, astext(transform(geometry,4326)) as geom4326
    FROM input1

    b) SELECT * , distance(centroid(geometry), *(SELECT geometry from input1)*) as cbdDistSQL
    FROM input2

    c) The italicized section is a subquery that will fetch the point from the CBD layer. It works this simply because the layer only has a single feature.

3) Try combining what you learned in the two queries above to find the distance between tract centroids and the CBD in the 4326 coordinate reference system (WGS 1984). If the transform functions work, your results should be incorrect, measuring in decimal degrees rather than in meters.

4) Try adding TRUE as a third parameter in the distance function: now you should get accurate geodesic measurements!

5) Try replacing your distance field calculator with this Execute SQL algorithm in your model.

Lab Five: PostgreSQL / PostGIS Introduction

An Honor Pledge for Spatial SQL

I will always create table names and field names beginning with a lower-case letter. The names will only include lower-case letters, numbers, and the underscore character _. I'll keep names short and simple so that I can write them in queries. No spaces, and no capital letters!

Table names and column names also cannot be identical to an SQL keyword:
https://www.postgresql.org/docs/8.1/sql-keywords-appendix.html

I will pay attention to syntax, punctuation, data types, and projections.

I will use Notepad++ or another simple text editor to keep a record of the queries I run, and I will write comments to explain to myself and others what the query does. The comments are formatted as follows:

--two dashes at the beginning will comment the remainder of a line.

/* slash and asterisk at the beginning and asterisk and star at the end will comment a whole block of text ***/**
I will save this text file as a **.SQL** file so that Notepad++ colorizes syntax in my queries.

I am aware that references for non-spatial SQL and spatial SQL are linked on Canvas.

Connecting with PG Admin

1) Use the PGAdmin client to connect to your database:
   a) Launch the PG Admin program. It will open in a web browser.
   b) PG Admin will ask you for a password: this is just local to this particular machine and to the PG Admin application. Choose any password that you like.
   c) Add/connect to a new Server
   d) Name the connection Name the connection -----
   e) Server host is: -----
   f) Server port is: ----
   g) Maintenance database: your database name (first name in lower case letters)
   h) Your user name is your first name as it appears in BannerWeb, in all lower case letters
   i) Your password is your student ID number without the leading zeros.
2) Extend your database with spatial functions
   a) Expand the cylindrical menu for your database
   b) Expand the **Extensions** menu
      i) By default, you should see one extension: plpgsql
   c) Right-click **Extensions** and **Create > Extension…**
      i) Search for the name: **postgis**
      ii) **Save**
   d) The postgis extension should appear under your Extensions menu.
3) As you populate the database with data tables, they will appear under **Schemas** → **public** → **tables**
4) Expand the Login/Group Roles menu
   a) Right click on your user name → **properties**

    b) Switch to the **definition** tab
    c) Enter a new password if you wish
    d) Save

5) Close PGAdmin
6) Note: you can install PGAdmin on any operating system… if you want to connect to artemis from off-campus, you need to have a VPN (Virtual Private Network) connection to Middlebury connected.
7) It is also possible to install your own PostgreSQL/PostGIS server on your own computer and create databases locally.

Load Given Data into QGIS

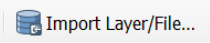
1) Load the three given shapefiles into QGIS
    a) I purposefully saved the central business district in an incorrect projection so that we can test our skills in PostGIS!
2) Also load the given .csv file, which is a data table with no geometry/geography data.
    a) Remember to always use **Layer → Add Layer → Add Delimited Text Layer** to load text files so that QGIS has a chance to try to detect data types correctly.
    b) Remember there is a **geometry** option while importing a text file, and this table should have **no geometry**

Connect to your Database

1) In the Browser panel, right click on the Elephant **PostGIS** tree and **New Connection…**
    a) Name: anything you like
    b) Service: ____
    c) Host: -----
    d) Port: ----
    e) Database: your database name (your first name)
    f) Test Connection to see if you can connect. It will prompt you for your user name (first name) and password (student ID)
    g) If successful, you should be able to navigate to the Public schema, but will not see anything inside yet.

2) Now open DB Manager by going to **Database → DB Manager**
    a) Connect to your database here by expanding the PostGIS menu and navigating to your database. You may be prompted to re-enter your user name and password.

3) Once the database is connected, you can import data into it with the **Import Layer/File…** tool



    a) Input: centralBusinessDistrict
    b) Table: change this to **cbd**
    c) Primary key should default to *id*, geometry column to *geom*, source and target SRID to *EPSG:2791*
    d) ☑ convert field names to lowercase
    e) **OK**

4) Also import the two polygon shapefiles for tracts and the census data table for 1940
   a) Re-name them to simple table names as you import: **tracts2010** and **tracts1940** and **table1940**
   b) ☑ convert field names to lowercase

5) Highlight your database name in the Providers tab and click the Refresh button to see that your data has loaded into the database

6) Check the Info tab for each table. Does it appropriately identify the geometry type, projection, fields, and field data types?

7) Check the table tab for each table. Does it appropriately show your data?

8) For spatial data, you can also use the Preview tab to get a simple view of the geometries.

Investigate Cardinality of tracts1940 and table1940

1) What are the possible join field(s)? Inspect each set of fields and data and ask yourself, which value or set of values is likely to help me match table1940 rows to tracts1940 rows?

2) Check the uniqueness of fields you want to use for joining. If duplicate values exist, then you might not have a one-to-one cardinality relationship. Selecting the count() of all distinct(unique) values, and of all values, helps us to do this. If there are fewer distinct values than total values, we could be in trouble.
   a) Open an SQL window:
   b) Check tracts1940 by writing this SQL query:
   SELECT COUNT(DISTINCT gisjoin), COUNT(gisjoin) FROM tracts1940
   c) **Execute**. What do you find? How many total rows are there, and how many of those are unique?

   d) And check table1940:
   SELECT COUNT(DISTINCT gisjoin), COUNT(gisjoin) FROM table1940
   Execute. What do you find? How many total rows are there, and how many of those are unique?

3) If you find any duplicate values, you'll have to determine if you can either delete the duplicates or group them into a single record before you proceed to joining.

Join data to shapes

4) PostgreSQL has left, right, inner and outer joins, with the differences illustrated in the Venn diagram at: http://www.postgresqltutorial.com/postgresql-joins/

5) If tracts1940 will be table a and table1940 will be table b, which form of a join will keep all of the tracts (regardless if they match any table1940), but will not include any table1940 data rows that do not match to a tract?

6) Try this join:

```
SELECT *
FROM tracts1940 AS a LEFT OUTER JOIN table1940 AS b
ON a.gisjoin = b.gisjoin
```

   a) In the query above, I *alias* tracts1940 to a and table1940 to b for ease of referring to each input table's fields in the other parts of the query.

7) Check how many tracts1940 rows are unmateched to table1940 rows by adding a *where* clause to limit the output rows to those meeting a condition where the matching foreign key is null.

```
SELECT *
FROM tracts1940 AS a LEFT OUTER JOIN table1940 AS b
ON a.gisjoin = b.gisjoin
WHERE a.gisjoin IS NULL
```

   How may tracts went unmatched?

8) Try to modify the query above to check for table1940 data that does not match to any tracts1940 record.
   a) Hint: switch from LEFT OUTER JOIN to RIGHT OUTER JOIN
   b) Hint: change the WHERE clause to look for NULL values in tracts1940 rather than in table1940.
   c) How many data rows went unmatched?

9) Based on your assessment of duplicate values and unmatched rows in the join, do you think there is any problem using the query in #7 (a LEFT OUTER JOIN) for analysis? What type of cardinality is there?
   a) Zero to one
   b) One to one
   c) Zero to many
   d) One to many

Customize the fields we choose to join, the order of output, and the rows that we keep

1) We know that duplicate field names are a problem in GIS, so we need to alter this join to select only the new fields that we want from table1940**: poptotal, white, nonwhite, medgrossrent**

```
SELECT a.*, b.poptotal, b.white, b.nonwhite, b.medgrossrent
FROM tracts1940 AS a LEFT OUTER JOIN table1940 AS b
ON a.gisjoin = b.gisjoin
```

   a) In the query above, **a.*** means to get all of the columns from table a, which is tracts1940

2) Order results by most expensive rent descending to least expensive:

```
SELECT a.*, b.poptotal, b.white, b.nonwhite, b.medgrossrent
FROM tracts1940 AS a LEFT OUTER JOIN table1940 AS b
ON a.gisjoin = b.gisjoin
ORDER BY medgrossrent DESC
```

3) **Remove zero-rent values from results**:

```
SELECT a.*, b.poptotal, b.white, b.nonwhite, b.medgrossrent
FROM tracts1940 AS a LEFT OUTER JOIN table1940 AS b
ON a.gisjoin = b.gisjoin
WHERE medgrossrent > 0
ORDER BY medgrossrent DESC
```

4) Some Wisdom
   a) Modifying the SELECT line's list of columns changes which columns you'll get
   b) Adding a WHERE clause changes which rows/records/features you'll get
   c) The order of a selection query's parts thus far goes:
      i) SELECT
      ii) FROM (optional JOIN … ON …)
      iii) WHERE
      iv) ORDER BY

Create a View

1) A VIEW is a selection query that appears in most respects like a table. It's like a *virtual* layer in GIS: a view pulling from other data tables in which the data is permanently stored. Try this query:

```
CREATE VIEW join1940 AS
SELECT a.*, b.poptotal, b.white, b.nonwhite, b.medgrossrent
FROM tracts1940 AS a LEFT OUTER JOIN table1940 AS b
ON a.gisjoin = b.gisjoin
WHERE medgrossrent > 0
ORDER BY medgrossrent DESC
```

2) Refresh the database to see your results. Try to right-click **join1940** and **Add to Canvas**. It should now be a layer in QGIS! This was a simple query, but the beauty of it is that the data is not stored in a single table: SQL is generating a virtual table that can be used *as if* the data were permanently stored this way. You can create *many different views* from the same database table(s) in order to suit your purposes.

Extend what we already learned about Distance & Direction

1) Calculate direction from CBD to join1940 using the 3395 World Mercator projection
   a) PostGIS has its own database of projections, which may not always perfectly match those in QGIS. For this query, let's use World Mercator projection with EPSG code 3395.
   b) Otherwise, the code should be the same as you've seen before, using functions:
      i) DEGREES
      ii) ST_AZIMUTH
      iii) ST_TRANSFORM
   c) And defining the central business district point with a subquery:
      i) (SELECT ST_TRANSFORM(geom, 3395) FROM cbd)

2) PostGIS has a unique data type: *geography*. Unlike *geometry*, the *geography* data type is special for data stored with latitude and longitude coordinates, and it calculates everything with geodesic calculations.
   a) See https://postgis.net/workshops/postgis-intro/geography.html
   b) If a *geometry* data is using geographic coordinates, it can be converted for geodesic calculations with the GEOGRAPHY() function. For example, a geography version of the centroid could be queried like this:
   (SELECT GEOGRAPHY(ST_TRANSFORM(geom, 4326)) FROM cbd))
   c) If both points in a function like ST_DISTANCE() are geography types, then the calculation will automatically be geodesic.
   d) Try to add a second function to your direction query to also calculate geodesic distance.
   e) Once the query seems to be working, create a VIEW of the query named tracts1940

Buffer Analysis

1) Keep in mind that every table should have a primary key which is a unique integer (typically **id** or **fid** or **oid**), and a geometry column (typically **geom)**. Also, function names are not good column names! Therefore, when we use functions, we always give them a proper alias.

2) Let's select a new version of CBD with the proper projection:

```
SELECT id, ST_TRANSFORM(geom, 3528) AS geom
FROM cbd
```

3) We can create a permanent copy of this by creating a table:

```
CREATE TABLE cbd3528 AS
SELECT id, ST_TRANSFORM(geom, 3528) AS geom
FROM cbd
```

4) Now it will be easier to buffer the point, starting in the correct projection. The buffer function is predictably, ST_BUFFER(geometry, distance) and the distance is measured in the same units as the input geometry. Let's create a 5km buffer around the central business district in order to assess the property values in the most central tracts.

```
SELECT id, ST_BUFFER(geom,5000) AS geom
FROM cbd3528
```

5) Create a view based on the query above, named cbd5km

Spatial Query

1) Let's select all of the tracts that are within 5km of the CBD by adding a spatial function for intersection, ST_INTERSECTS in the WHERE clause:
   a)
```
SELECT *
FROM join1940
WHERE ST_INTERSECTS(geom, (SELECT geom FROM cbd5km))
```

2) Try to accomplish the same selection without using cbd5km, but by applying an ST_BUFFER() function to the cbd3528 geometry.

3) Try replacing ST_INTERSECTS with ST_WITHIN.
   a) How many rows/features does each query return?
   b) The PostGIS geometry functions guide has some information on all the spatial queries, but the ultimate authority is always the open standards for simple features: https://www.opengeospatial.org/standards/sfa

4) Try creating a few of the tracts intersecting the 5km buffer, named cbd1940

Aggregation & Dissolve

1) SQL supports several *aggregate functions* which will summarize a collection of rows into a single row. When building a query with aggregate functions, every single output must be some form of aggregate.
2) You have already used one of these, the COUNT() function.
3) Let's find the minimum, mean, count, and average of the rent in cbd1940 tracts:

```
SELECT COUNT(id) AS countID, MIN(medgrossrent) as minRent, AVG(medgrossrent) as avgRent,
MAX(medgrossrent) as maxRent
FROM cbd1940
```

4) There is also one aggregate function for geometries, to union them together. Let's see what that gives us:

```
SELECT 1 as id, COUNT(id) AS countID, MIN(medgrossrent) as minRent, AVG(medgrossrent) as
avgRent, MAX(medgrossrent) as maxRent, ST_UNION(geom) as geom
FROM cbd1940
```

   a) What do the results of this query look like? Create a view of them and load into QGIS…

That's enough!

Check your answers with peers and your professor. This wasn't anything dramatically new, but hopefully you're getting comfortable with writing SQL queries and using them for some fundamental spatial analysis!

More on Databases & SQL for Disaster Resilience!

1) Dissolve with statistics, but group by one or more categories, extending the last query from lab: use a CASE statement to categorize tracts by segregation types… and then:

```
SELECT 1 as id, COUNT(id) AS countID, MIN(medgrossrent) as minRent, AVG(medgrossrent) as
avgRent, MAX(medgrossrent) as maxRent, ST_UNION(geom) as geom
FROM cbd1940
GROUP BY segregationtype
```

2) You *can* **Watch** a repository… as long as you accepted the invitation to join the team, you can sign up to watch for notifications of changes / posts!
3) Additional reasons for learning SQL in PostGIS
   a) Web applications, e.g. Carto, GeoNode, DevelopmentSeed
   b) SAGA can log in to postgis databases as well: and postgis has an extension for raster data
   c) The functions are the same as OGC simple features functions, so you'll have the same options in R or GeoPandas Python, or so many other platforms
4) Means for getting OSM data
   a) QGIS Plugin: Quick OSM
   b) Download .osm file directly from OpenStreetMap
5) Adding web map service (WMS) layers to QGIS (right click WMS menu in Browser)
   a) Find web service url, e.g. Resilience Academy is https://geonode.resilienceacademy.ac.tz/geoserver/ows
   b) And the Columbia University gridded population of the world is: http://sedac.ciesin.columbia.edu/geoserver/wms
6) Adding web feature services (WFS) to QGIS (right click WFS menu in Browser)
   a) Find the web feature service URL, e.g. Resilience Academy is: https://geonode.resilienceacademy.ac.tz/geoserver/ows
7) Adding XYZ tiles to QGIS
   a) Ramani Huria: https://a.tiles.mapbox.com/v3/worldbank-education.pebkgmlc/{z}/{x}/{y}.png
   b) Many (most?) tile services will require an authentication key, and may not be public.
   c) This is like using QuickMapServices plugin, but for tiled data that is not publicly indexed, e.g. you can make your own tile styles with a Mapbox account and use that for your QGIS background!

8) Additional SQL to 'see' characteristics of your geometry data:

   a) `St_astext()`

   b) `St_srid()`

   c) `Geometrytype()`

   d) `St_dimension()` : what dimension is the geometry? Yours should almost always be 2

   e) `St_numgeometries()` : how many parts to a multi-part geometry?

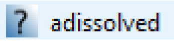
   f) `St_numpoints()` : how many vertices in a line?

   g) `St_isvalid()`

9) `St_makevalid()` : fix invalid geometry

10) Can you make a CASE statement to check for invalid geometries, and convert them with `st_makevalid()` if necessary?

11) One challenge with spatial databases: geometry metadata (projection, geometry type, dimensions) is stored in *two places*: within the geometry itself, and in the `geometry_columns` index of all the database's geometry data. These *must always match*, which sometimes requires you to manually ensure they are in sync and up-to-date. For example, if a table, like `adissolved`, shows up like this in DB Manager:



adissolved

, it's an indication that the `geometry_columns` information is incomplete.

   a) `Create table St_transform(geom, 4326)::geometry(multipolygon, 4326)  as geom` will ensure that the `geometry_columns` index will be updated with the geometry information for your new table... in case it's geometry type showed up as unknown:

   b) `select populate_geometry_columns()`
   will look through all of your tables, try to detect any geometry data, and update `geometry_columns` index table accordingly. You can specify it to update just a specific table with select `populate_geometry_columns('public.tableName'::regclass)` Just replace tableName with the actual name of your table.

   c) A particular problem with some overlay or dissolve operations is that the results might be a mixture of single-part and multi-part geometries, making it impossible to update `geometry_columns` with a single geometry data type. ☹ If this happens to you, use `st_multi()` to convert all of the geometries to multi-part geometries.

A wild idea

It seems like informal settlements could be characterized by their disorganization: the buildings are constructed helter-skelter at different angles. What if we could create a GIS raster layer that represented the level of disorganization or variance in the directions buildings face compared to other nearby buildings? We'd want to capture the compass direction of the longest wall of a building...

1) `St_geometryN(geom, 1)` will get the first point of multi-point, first polygon of multi-polygon, etc. This is a good way to convert a single multi-point feature to a point.
2) `St_exteriorRing(geom)` will give a linestring for the outside of a polygon
3) Can you combine `st_geometryN` and `st_exteriorring` to get the exterior ring of the first polygon in a multi-polygon feature?
4) `st_pointN(geom, 1)` will get the first point in a linestring, while `st_pointN(geom, 2)` will get the second point in a linestring, and so on.
5) Can you use `st_pointN` on the exterior ring of a building layer to calculate the length of the first side of a building? How about the second side of a building?
6) Can you imagine a `CASE` statement that would find the *direction* for the longer of the first two sides of a building?

Lab Six: Resilience Academy in Dar es Salaam

Please develop spatial questions about resilience and vulnerability in Dar es Salaam into spatial analysis with data from Ramani Huria, available through OpenStreetMap and Resilience Academy. Implement a solution to your question(s) in PostGIS and thoroughly document your analysis, at first using Notepad ++ for drafting and commenting queries. The final product will be a practical lesson plan / tutorial for students learning how to apply geospatial analysis to urban hazards management.

You could take inspiration for tutorial development from this course, or from online examples like the DUSPviz tutorials on QGIS geocoding or Intro to postGIS: http://duspviz.mit.edu/tutorials/#gismapping

At least one input for your question should come from the OpenStreetMap data.

At least one final step to your question should be to quantify a dimension of resilience or vulnerability in each subward of Dar es Salaam, so that your findings could be translated back into policy action consistent with existing administrative hierarchies and geographic regions.

Considering data error and accuracy, can you get some insight into the data you're using by querying `osm_user`, `osm_uid`, `osm_version` and/or `osm_timestamp`? These metadata, associated with every feature, are explained here: https://wiki.openstreetmap.org/wiki/Elements

You may work individually or in pairs on the analysis in this lab assignment, but please post the final product to both GitHub accounts so that you each have your own copy.

Please try your queries on small samples of data first, by using the `LIMIT` clause or using a `WHERE` clause to apply a condition to select a small sample of features. Remember to add indices to accelerate time-consuming queries.

Next Wednesday and Thursday, we will discuss and implement methods for visualizing large datasets in interactive web maps, for which you will use your results from this lab. Therefore, do not worry about spending time visualizing the complete set of your results. Instead, focus on visualizing small samples of your results for instructional purposes.

Proposed Questions

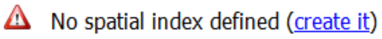
Your research question may include any of these proposed topics, or something new:

- Accessibility of residential buildings to roads passable by service/emergency vehicles
- Density of storm drains, especially comparing inside & outside of floodplains
- Presence of community assets
    - You can do network analysis with PostGIS with the pg_routing extension to also get accessibility to community assets, but I suggest this is a topic for an independent project later
- Presence of open space or green space / parks, and maybe trees?
- Land cover in floodplains, using the presence / area of buildings as a proxy
- Conditions of drains and culverts in each locality, and potentially co-location of drain conditions and flood conditions
- Proximity of residential & commercial buildings to floodplains and to drains

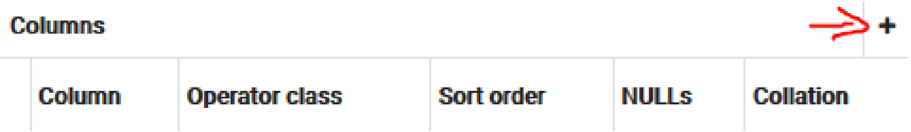
Tips for efficient spatial SQL work

Please read this section to remember strategies for making SQL work with big data more efficient.

1) While you're just trying to develop a query, add `LIMIT 100` or `LIMIT 10` to the end of your query so that it runs only on the first 100 or 10 rows. For example, rather than previewing a full table, instead write a query:
   a)
   ```
   SELECT *
   FROM my_huge_table
   LIMIT 100
   ```

2) Use spatial indices on data before you apply spatial selection or overlay operations!
   a) In DB Manager, the layer properties may indicate  ⚠ No spatial index defined (create it), in which case use the "create it" button and wait a bit while it generates an index.

3) Use column indices on large datasets before you use those columns for joins, grouping, or sorting.
   a) In PG Admin, right click on a table → **create** → **index**
   b) Make up your own index name and set the table space to `pg_default`
   c) Under Definition, add a column with the + button.



Columns +

	Column	Operator class	Sort order	NULLs	Collation

   d) All the other default settings are ok

4) Spatial and attribute indices are listed in both PGAdmin (dependents menu of a table) and DB Manager (info menu for a table)

5) PostGIS will automatically use any indices that will help a query run faster: you don't need to change your query in any way.

6) Recall the two most useful references:
   a) For non-spatial PostgreSQL questions: http://www.postgresqltutorial.com/
   b) For spatial PostGIS functions: http://postgis.net/docs/manual-1.5/ch08.html

Adding and calculating columns in PostGIS

Please read this section to get a sense of possibilities for analysis in PostGIS, and return to it later when you need these types of functions.

1) You can add columns to existing tables! In order to add a column, use an `ALTER TABLE` query:
   a) `ALTER TABLE table_name ADD COLUMN column_name data_type`
      i) Replace `table_name` with the name of the table you are changing
      ii) Replace `column_name` with your new column name, following the honor code of naming in SQL!
      iii) Replace `data_type` with your new data type, which may be any of these:
         (1) `Boolean`
         (2) `Integer`
         (3) `float8` (decimal number)
         (4) `text`
   b) http://www.postgresqltutorial.com/postgresql-alter-table/
   c) http://www.postgresqltutorial.com/postgresql-data-types/
   d) https://www.postgresql.org/docs/9.5/datatype.html

2) If you want to add a *geometry* column, there is a special function, `addgeometrycolumn()`
   a) `SELECT addgeometrycolumn('public', 'table_name', 'column_name', srid, 'geometry_type', 2)`
      i) 'public' refers to the Public schema, does not need to be changed.
      ii) Replace table_name with the name of the table you want to add a column to
      iii) Replace column_name with the name of the new geometry column
      iv) Replace srid with the EPSG number for the coordinate reference system you want to use. This is not in quotes.
      v) Replace geometry_type with the type of geometry, which may be:
         (1) POINT a single point
         (2) MULTIPOINT a collection of points (multi-part)
         (3) LINESTRING a single line
         (4) MULTILINESTRING a collection of lines (multi-part)
         (5) POLYGON a single polygon
         (6) MULTIPOLYGON a collection of polygons (multi-part)
      vi) Keep the number 2, for two-dimensional geometries.
   b) https://postgis.net/docs/AddGeometryColumn.html

3) You can calculate or re-calculate values in an existing table column using an `UPDATE TABLE` query:
   a)
   ```
   UPDATE table_name
   SET column_name = formula_or_expression
   ```
      i) Replace table_name with the name of the table
      ii) Replace column_name with the name of the column you are calculating
      iii) Replace formula_or_expression with another field name, a formula, or a sub-query

4) You may need to update the values in one table based on information contained in another table. In this case, use a form of UPDATE that includes a join to another table:

   a)
   ```
   UPDATE first_table
   SET column_name = formula_or_expression
   FROM another_table
   WHERE join_criteria
   ```
   i) Replace `first_table` with the name of the table you are updating
   ii) Replace `column_name` with the name of the column you are updating
   iii) Replace `another_table` with the name of the table you are joining for additional information
   iv) Replace `join_criteria` with either an attribute join in the form of:
   ```
   first_table.join_field = another_table.join_field
   ```
   or with a spatial join in the form of:
   ```
   st_intersects(first_table.geom, another_table.geom)
   ```

5) Optionally, you can calculate or re-calculate values on a sub-set of rows in a table (like calculating on *selected features only*) by adding a WHERE clause:

   a)
   ```
   UPDATE table_name
   SET column_name = formula_or_expression
   WHERE selection_condition
   ```
   i) Replace selection_condition with your criteria. These can be based on either attributes or spatial functions, like `st_contains`, `st_intersects`, or `st_within`

6) Spatial overlay functions can be accomplished similarly to calculation of fields. For example, PostGIS recommends this type of query for an intersection spatial overlay. Note that `st_intersects()` gives you a true or false answer (yes it intersects, or no it doesn't), while `st_intersection()` gives you a new geometry, which is the result of overlaying two features and finding where they intersect.

```
SELECT a.column_name, b.column_name,
   CASE
   WHEN ST_CoveredBy(a.geom, b.geom)
   THEN ST_Multi(a.geom)
   ELSE
    ST_Multi( ST_Intersection(a.geom,b.geom) ) END AS geom
 FROM first_table AS a
   INNER JOIN second_table AS b
    ON (ST_Intersects(a.geom, b.geom)
      AND NOT ST_Touches(a.geom, b.geom) );
```

   a) First_table, aliased to a, should be the smaller features. Second_table, aliased to b, should be the larger features.
   b) The example shows keeping just one attribute from each table, but add as many attribute columns as you need to the first row of the SELECT statement.
   c) One key to optimizing speed in this query is first joining by only the features that intersect one another (`st_intersects()`), excluding ones that only intersect on their boundary (`not st_touches`)

d) A second key is to simply keep geometries unchanged if they are completely covered by the other layer, and only bother with a spatial overlay intersection if they are partly within and partly outside of another feature. This is accomplished with the `CASE` evaluation of `st_coveredby()`

e) The query structure is copied from https://postgis.net/2014/03/14/tip_intersection_faster/

Loading OSM Data into PostGIS

Please consider which data layer(s) and columns you'll need from OpenStreetMap before completing this section. The tool in this section will only import features with the tags you ask for, and those tags will become columns in a PostGIS table.

1) Someone has programmed a small command-line tool to parse through OpenStreetMap files and load them into a PostGIS database.
   a) OSM2PGSQL on GitHub: https://github.com/openstreetmap/osm2pgsql
   b) Hacker's guide to OSM2PGSQL https://www.volkerschatz.com/net/osm/osm2pgsql-usage.html
   c) Download for windows: https://ci.appveyor.com/project/openstreetmap/osm2pgsql
   d) Download for macOS: use Homebrew, on which osm2pgsql is a predefined package: https://brew.sh/

2) Go to the `K:\` network drive (`\\splinter\courses`) and to the `gg323` folder

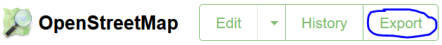
3) Copy the `osm_script` folder to your own `W:\` drive

4) Open the `smallmap.osm` file in Notepad ++
   a) The entire OpenStreetMap database is stored as XML tags like this! Every feature is a node (point) XML tag or way (line or line encompassing a polygon) XML tag, and its attributes follow in "tag" XML tags.
   b) The more complete Dar es Salaam file is too large (1.5 gb) even to open in Notepad ++ , and I've left it on the `K:\` drive so that we can all use it without duplicating…

5) Edit the `convertOSM.bat` file in Notepad ++
   a) Change the database name to your database, and the user name to your user name
   b) All of the parameters are explained in comments at the end of the document.
   c) Save your changes.

6) Open the `dsm.style` file in Notepad++. This file instructs osm2pgsql which types of features to load, and which tags to use.
   a) In case you need to add any additional tags as columns in the database, you'd need to edit the `dsm.style` file. For example, we talked about whether drains were blocked or not, so I added a line to ingest the blockage information:
      ```
      node,way   blockage     text          linear
      ```
   b) You can replace node,way with node if you only want points, or way if you only want lines or polygons
   c) You can replace blockage with new tag names.
   d) You can replace linear with polygon *if you only want this tag for polygons*.

7) Run the convertOSM.bat script.
   a) It will prompt you for a password, and the cursor will not move as you type. Just type your password and press enter…
   b) The script with parse through the dsm_osm.osm file, and load relevant data into your PostGIS database!
   c) The data will be loaded in WGS 1984 geographic coordinates (epsg:`4326`). You'll want to transform most data into UTM Zone 36S, EPSG code `32727`

8) How did I download the osm file?
   a) At https://www.openstreetmap.org , zoom and pan the map to the extent of data you wish to download
   b) Use the Export button 

   c) Click the **Overpass API** link to download from a mirror of the OpenStreetMap database (using a mirror offloads the work of exporting to alternative servers, so that the main server can keep doing its thing: editing data and serving it up as OSM tiles!
   d) Make sure the downloaded file is saved as a `.osm` file, or just rename it afterwards.

Loading Resilience Academy Data into PostGIS

1) Add a WFS feature layer (in QGIS **Browser**, right click **WFS → New Connection…**)
   a) The name can be anything, but it makes sense to use Resilience Academy
   b) The URL is `https://geonode.resilienceacademy.ac.tz/geoserver/ows`
   c) OK

2) Add the `Dar es Salaam Administrative Sub-wards` layer to QGIS

3) Open up the table and notice: which field is best used as the unique integer primary key?
   a) Notice the mess that is the attribute data for this layer. The database ethnographer in me suspects that they subdivided the work of digitizing subwards and then merged several different shapefiles without standardizing columns & field names… some of the fields I recognized from Tanzanian Census shapefiles, while others are a mystery.
   b) This is a great example of a lack of *internal consistency*: you don't need any other data source to know that the attribute data in this layer is inconsistent.
   c) Also, FYI: if someone had created a PostGIS database schema for this project at the beginning, you'd never see this mashup of inconsistent attributes.

4) Open DB Manager and connect to your database

5) Import Layer/File…
   a) Select the Dar es Salaam Administrative Sub-wards layer
   b) Change the table name to a proper table name: subwards
   c) Change the Primary key to the field you identified as a good primary key
   d) ☑ Convert field names to lower case

    e) ☑ Create spatial index (so that you'll optimize spatial queries from the beginning!)

6) Data in other layers may already exist in the OpenStreetMap data you imported, or you may add additional layers now from Resilience Academy.

Now that you've loaded millions of data points…

1) Draw a workflow for the spatial analysis in your question given the layers available in your database and the spatial analysis techniques you're familiar with from QGIS or ArcGIS.
    a) Consider simpler questions and solutions initially to gain some experience and confidence first.

2) Translate one small piece of the workflow at a time into SQL, practicing with SELECT queries (that do not modify data) with a small subset of data, before you start making more permanent layers with CREATE TABLE queries, or modifying existing layers with ALTER TABLE or UPDATE queries. Keep notes in Notepad++

3) Finalize the SQL workflow and visualize some examples of your outputs at each step for instructional purposes.

Lab Six.2 Web Mapping with Leaflet

The goal for this new lab material is to visualize the final product(s) from your Resilience Academy analytical lesson in the form of a Leaflet map on GitHub. If any of the instructions contained here are ambiguous or could use an accompanying video tutorial, please let me know by Friday at 2:30 pm.

New resource: PostGIS Cheat Sheet: https://www.postgis.us/downloads/postgis20_cheatsheet.html

Creating interactive "slippy" maps (those you can pan and zoom easily) for the internet is getting easier and easier! Two powerful open-source JavaScript libraries for map construction have been game-changers:

1) OpenLayers: https://openlayers.org/
2) Leaflet: https://leafletjs.com/
   a) Documentation: https://leafletjs.com/reference-1.5.0.html
   b) Tutorial examples (a great way to learn Leaflet from the bottom up): https://leafletjs.com/examples.html

QGIS has a QGIS2WEB plugin for automatically generating webpage files for Leaflet. I'm most familiar with Leaflet, so let's go with that!

1) QGIS2WEB: https://github.com/tomchadwin/qgis2web

Web Mapping Technologies

1) Vector data is usually stored in GeoJSON format: https://geojson.org/
   a) This is a geographic version of JSON: JavaScript Object Notation

2) HTML (hypertext markup language) is the language used to format webpages, based on tags contained in brackets.

3) CSS (cascading style sheets) is a notation for styling webpages: think of it as a code version of a set of styles or a template in a program like Microsoft Word.

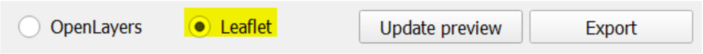
4) JavaScript is a light programming language in which code can be written for interactive web applications.

Using QGIS2WEB

1) Before exporting a map to the internet:
   a) Export or query feature layers with *minimum* of attributes: don't include anything you don't want public or used for styling or popups!
   b) Symbolize points as simple marker circles for ease of translating to Leaflet `circleMarker` symbols.
   c) Do not attempt to display all buildings or building centroids: the datasets are too large. To successfully create an interactive map of such a large dataset, you would need Leaflet to be able to connect directly to your database in order to optimize querying only the features in the current map view, and you would need to make the layers invisible until the map is zoomed closely enough to query only a smaller set of features. This is not possible with our current database setup.
   d) Include at least one base layer, e.g. by using **QuickMapServices** to add the **OpenStreetMap** layer.

2) Load the QGIS2WEB plugin
   a) Plugins → Manage and Install Plugins →
   b) Web → QGIS2WEB → Create Web Map

3) Set the exporter to use **Leaflet** with the option at the bottom of the plugin:

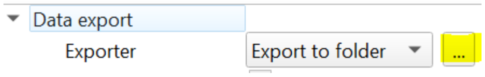


○ OpenLayers ● Leaflet

4) **Layers and Groups** tab settings:
   a) Check only the layers you want to include in the web map.
   b) Your map readers will be able to turn layers on or off. Uncheck the **Visible** option if you want to load a layer into your map, but to start with the layer being invisible until the map reader selects it in the legend.
   c) If any of the layers are WFS (web feature service) layers, check the option to create a GeoJSON for them. This essentially makes your own local copy of that data.
   d) If any of the layers are WMS or XYZ Tile layers, the exporter will direct the Leaflet map to acquire data directly from the same internet-based WMS or XYZ servers without saving any local data.
   e) If any of the layers are local vector layers (e.g. from your own database), then the exporter will convert them to `.js` files using the GeoJSON format.

5) **Appearance tab:**
   a) Change **Add layers list** to *expanded* to always list each layer and allow them to be turned on and off
   b) Change Template to *full-screen* to create maps that fill the web browser.
   c) Scale/Zoom can be customized:
      i) **Extent** is setting the default zoom level and extent of the map when it loads in a webpage. Zoom and pan your current QGIS view to where you'd like the map to start, and set this to *Canvas extent*.
      ii) **Max zoom level** limits how far the user can zoom in. Most web map base layers do not zoom beyond level `19`
      iii) **Min zoom level** limits how far the user can zoom out. The world looks dumb zoomed out too far in world Mercator, and this is a localized project. Zoom level `6` is already large scale enough to allow someone to see all of Tanzania on their screen.
      iv) ☑ **Restrict to extent** : you may use this to prevent the map from ever panning out of the bounds of the extent currently displayed in QGIS.
   d) By default, let's not complicate the map with options for address search, geolocate user, highlight on hover, layer search, project CRS, measure tool, or popups.

6) **Export tab:**
   a) Set the **exporter** to *export to folder*, and use the ellipsis button set the folder to a new folder on your `W:\` drive



Data export

Exporter: Export to folder

   b) Leave other options as default: no labels
   c) With this setting, the plugin will create a time-stamped folder inside your export folder, containing all the files for your map.
   d) You can set **precision** to your max zoom level. This can simplify vector geometries to the precision needed to draw them correctly at your largest visualization scale.

7) **Update preview** will show a preview of the map on the right site. It can take quite some time to load with complicated vector datasets.

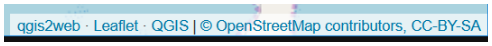
8) **Export** to save the map in a time-stamped folder inside the export folder you set above.
   a) It may take some time to preview or export layers with large numbers of features to GeoJSON.

9) The export probably opens a webpage of your map.
   a) If not, open the folder you set to the export folder and open `index.html`.
   b) Enjoy your map for a minute!
   c) Let's edit final touches of the map before uploading to GitHub.

Editing the HTML page

1) Open the `index.html` file for your map in Notepad ++
   a) Each time you make a change to this file, I suggest saving it and refreshing the page in your web browser, to see if the change worked.

2) Search for the `map.attributioncontrol.SetPrefix` function.
   a) This function creates the text shown at the bottom-right of your map:



qgis2web · Leaflet · QGIS | © OpenStreetMap contributors, CC-BY-SA

   b) Anything in the prefix will *always appear*, and additional attributions for individual layers will appear only when the layer is activated. Try turning layers on and off for your map: if you have an OpenStreetMap layer, its attribution probably appears and disappears with the layer.
   c) All maps should have a prefix crediting qgis2web, Leaflet, and QGIS as technologies used to create the map.
   d) Notice the whole prefix is enclosed in single quotes.
   e) Notice each attribution is actually HTML code for a link so that you can click the attribution and go to its webpage. For example: `<a href="https://qgis.org">QGIS</a>`
      i) `<a href="https://qgis.org">` is the beginning of the link, where href= defines a web address, in double quotes
      ii) `QGIS` is the text shown on the map for the link
      iii) `</a>` ends the link.
      iv) Notice how other links include `target="_blank"` ? Try clicking the QGIS link, and then clicking the Leaflet link. Notice that one opens a new window/tab and the other does not?
      v) Try adding `target="_blank"` to the QGIS link:
      `<a href=https://qgis.org target="_blank" >`
      vi) Save the HTML file and refresh your map and try to see if this forced the QGIS link to also open a new tab.

vii) Note: `&middot;` is code for the small dot between attributions, but most are separated by `|` (the vertical bar on your slash `\` key)

f) Try changing the prefix to start with your name and a link to your GitHub account. For example, I have revised my prefix function to:

```
map.attributionControl.setPrefix(
'<a href="https://gis4dev.github.io">gis4dev</a> |
<a href="https://github.com/tomchadwin/qgis2web" target="_blank">qgis2web</a>
&middot;
<a href="https://leafletjs.com" title="A JS library for interactive
maps">Leaflet</a> &middot;
<a href="https://qgis.org" target="_blank">QGIS</a>');
```

i) I did not add `target="_blank"` to my own link, because I want users to be able to go from my GitHub page to my map and back again without opening a new tab.

3) Search for `attribution` and be sure to give appropriate credit for all layers.

a) Tile layers like OpenStreetMap have this filled automatically, as an attribute in their layer definition. E.g. my OpenStreetMap layer has a line:

```
attribution: '<a href="https://www.openstreetmap.org/copyright">©
OpenStreetMap contributors, CC-BY-SA</a>',
```

b) I will change the line for my Subwards layer from `attribution: '',` to

```
attribution: '<a href="https://geonode.resilienceacademy.ac.tz/"
target="_blank">Resilience Academy</a>',
```

Optional Leaflet Customizations

1) Would you like a scale bar on the map?

a) At the end of the document, *just before the final* </script> tag to end the JavaScript, add a line:

```
L.control.scale().addTo(map);
```

b) Scale can be further customized, referring to https://leafletjs.com/reference-1.5.0.html#control-scale

c) For example, optionally a set of parameters with one to remove imperial measurements and keep metric measurements:

```
L.control.scale( {
imperial: false,
metric: true
} ).addTo(map);
```

2) Each vector layer has a function to define its style, and if the layer is classified, it will have a series of *if* conditions to decide first which class the feature is in, and next which style to apply. You may experiment with the following variables, some of which apply only to circle markers and some of which only apply to lines or polygons. Find these functions by searching for `function style_`

a) *Radius*: size of circle markers.

b) *Opacity*: with `1` being opaque and `0` being transparent.

c) *Color*: is the color of a line or outline, with an `rgba()` function

i) First number is Red, on scale from `0` (no color) to `255` (full color)

ii) Second number is Green, on scale from `0` (no color) to `255` (full color)

iii) Third number is Blue, on scale from `0` (no color) to `255` (full color)

iv) Fourth number is transparency, on scale from `0` (completely transparent) to `1` (completely opaque)

v) For help on choosing colors, use https://www.w3schools.com/colors/colors_picker.asp to find the red, green, and blue values you want.

d) *Weight*: is the width of a line or outline

e) *fillOpacity*: is the opacity of a marker or polygon fill

f) *fillColor*: is the color of a marker or polygon fill

g) *interactive*: values of `true` and `false` control whether pop-ups will be functional for each feature and each layer. True is functional, false is disabled.

3) If you have enabled feature pop-up content, you can change its formatting. Search for `popupContent.`

a) The popup content creates a text string of HTML code enclosed by single quotes.

b) It sets up a table with a row for each attribute. Do you see the begin table `<table>` tag and end table `</table>` tag?

c) Each row contains one attribute and begins with `<tr>\` and ends with `</tr>\`

d) You can delete any rows you don't want to see in the popup, or if you know HTML, format the popup any way you wish.

4) You can customize the initial map extent, maximum and minimum zoom levels, and limits to how far the user can pan the map.

a) Near the top of the HTML document, the map itself is initialized with a new variable:
var map = L.map('map', { … })

b) Within the {} object, you can set the `maxZoom` and `minZoom` levels, as long as `zoomControl:true` exists to turn on zooming control.

c) The `L.map()` function is followed by a `.fitBounds` function, giving the lower-left and upper-right coordinates of the initial map view in decimal degrees.

d) For example, in the map initialization below, the map must not zoom in beyond level 19 or out beyond level 6, and initializes by viewing and area bounded by 7.154 degrees South, 38.786 degrees East, 6.508 degrees South and 39.851 degrees East:

```
var map = L.map('map', {
    zoomControl:true, maxZoom:19, minZoom:6
}).fitBounds([[-7.154,38.787],[-6.508,39.851]]);
```

e) If you want the user to be limited to panning beyond a set limit, find the `function setBounds(){ … }` and change its contents to:

```
function setBounds() {
     map.setMaxBounds([[-7.154,38.787],[-6.508,39.851]]);
}
```

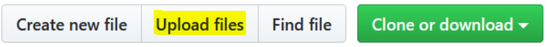
You may change the bounding coordinates to any region you need, and I suggest starting by copying the coordinates from the map extent in the `fitBounds()` function above.

Uploading to GitHub

1) Rename the folder containing your Leaflet map to something simple with no spaces, e.g. `dsmmap` (for Dar es Salaam map)
    a) Note: it is nearly impossible to delete an entire folder with contents using GitHub online. Therefore, I really suggest using a simple consistent name for your Leaflet map(s) and if you need to modify that, you can upload again and overwrite the previous map.

2) Log into your GitHub account and navigate to your `.github.io` repository

3) Go to the **Upload files** button



    a) Drag the *folder* onto gitub's file upload window.
    b) Once all the files have uploaded, **commit changes**

4) Edit your index.md file to add a link to your map. For example, mine is:

```
[Leaflet map of pharmacies in Dar es Salaam](dsmmap/index.html)
```

5) Congratulations! You've made an interactive web map with the Leaflet JavaScript library!

Learning from others' Leaflet maps

1) If you want to learn more, start by looking at the Leaflet tutorial maps: https://leafletjs.com/examples.html
2) For any leaflet map you see on the internet, right-click the page and **View Page Source** to see their code. For example, the first simple leaflet tutorial https://leafletjs.com/examples/quick-start/ has links to "see this example stand-alone" below each sample. Open those links and view the page source to see how the example was made.
3) This instruction is for Firefox, but all web browsers tend to have a similar option.

Lab Seven: Multi-criteria analysis of Vulnerability in Malawi

1) The practical goals for this lab are to:
   a) Replicate Figure 4, a map of average resilience scores in Malawi in 2010.
   b) Replicate 80% of Figure 5, a map of vulnerability to climate change in Malawi.
      i) 20% of the data inputs for the map are unavailable: Livelihood Sensitivity from FEWSnet.
2) Reference:
   Malcomb, D. W., E. A. Weaver, and A. R. Krakowka. 2014. Vulnerability modeling for sub-Saharan Africa: An operationalized approach in Malawi. *Applied Geography* 48:17–30.

Database Connection

1) If you get frequent errors to the effect of "could not translate host name "artemis" to address: Unknown host", try using the IP address of `artemis` in place of the name in your PostGIS database connection, as: **Host**: `140.233.36.33`

Summarizing DHS surveys by Traditional Authority

1) Data for lab, including DHS survey metadata, traditional authorities (2 versions, one from the census and one from GADM), and cluster points: mwi_data.zip
2) Draft SQL code for analyzing resiliency by Traditional Authority according to Malcomb's methods. You have the following tables:
   a) **dhsclusters** with a numeric column named **dhsclust** storing a unique id for each cluster
   b) **dhshh** with a row for each household and columns according to this list (translated to lower-case):
      DHS_metadata.txt
      DHS_Survey_Vars.pdf
   c) **mwita** with polygon geometries for each Traditional Authority
3) Inspect the DHS survey metadata and compare to Malcomb's paper
   a) Which variables do you need?
   b) What is their *nodata* or *null* value?
   c) How are they coded, and how can those codes be ranked on a scale of 5, with low scores having low resilience and high scores having high resilience?
4) Plan to rescale each indicator, aggregate the indicator into resilience scores for each household, and average households by traditional authority.

Reproducibility Challenges, Reflections

Friday: Meet in Davis Library Room 105A

Challenges

1) Flood risk data is badly misrepresented in the UNEP Global Risk Platform portal.
    a) The Malawi extract was clipped incorrectly.
    b) The Africa extract and the Malawi extract are actually population exposure to risk!
    c) Only the Global layer is correct, with a scaling of 0 – no risk up to 5 – high risk.
        i) "normalization from 0 to 5"
    d) There are no values of 5 within the extent of Malawi

2) We know we need to rescale exposure to drought to a 1 to 5 scale
    a) Our best guess is that Malcomb et al would have used a *quantile* equal count method to do this.
    b) Should we be including Lakes, National Parks, and areas outside of Malawi in this? It appears that data has been removed from those regions in Figure 5.

3) GADM shapes are rough: they extend far into Lake Malawi and have other topological errors.
    a) In attempting to *reproduce* the work, we should probably not make substantial revisions to the GADM errors (reproduce first, improve later…), but it does seem like the authors removed the large Lakes.
    b) Two options:
        i) *Difference* the GADM Traditional Authorities with Lakes
            (1) GADM polygons currently extend beyond the East side of Lake Victoria. ☹
        ii) *Clip* the GADM Traditional Authorities with Livelihood Zones (which seem to have reasonable coverage of land in Malawi)
            (1) There are some very small splinters / gaps in between Livelihood polygons

4) Before you undertake any raster grid analysis, you must choose, and know the properties of your input data, for:
    a) The coordinate reference system
    b) The *extent* in terms of the Xmin, Xmax, Ymin and YMax
        i) Grids will 'snap' to the lower-left corner: Xmin,Ymin coordinates, so be sure these are consistent across all inputs.
    c) The *spatial resolution* in terms of the cell size
    d) The *nodata* value
    e) Any *mask* you want to apply (i.e. *nodata* values for areas to be excluded in the study)
    f) All rasters should be resampled into identical grid systems for analysis

5) The drought exposure and flood risk rasters are provided with different spatial resolutions, so one must be resampled to match the other.

A kit of tools complements of GDAL and GRASS

1) **GDAL Warp** : what does this *not* do? Project, clip, resample…
   a) Set CRS (keep both as epsg:4326)
   b) Set nodata value
   c) Set output file resolution (cell size)
   d) Change data types
      i) Positive or negative integers: byte, int16, int32
      ii) Only positive integers: Uint16, Uint32
      iii) Decimal numbers: Float32, Float64
   e) Georeferenced *extents* of output file to be created
      i) Setting the CRS for this extent is not actually optional. You must set it.

2) **GDAL Rasterize (vector to raster)**
   a) Convert vector data to raster
   b) Rasters only store one value in each location, so you must choose *one attribute* from the vector data. This is the *burn in value*.
   c) You should know your anticipated raster CRS, extent, and cell size and set these accordingly

3) **GDAL Clip raster by extent**
   a) Clip a raster down to a smaller extent!

4) **Raster Calculator**: also, what can this *not* do?
   a) Raster version of overlay: it calculates a formula on collocated cells' values.
   b) Inputs should already be in the same extent, projection, resolution, etc.
   c) You can use this for *masking* (i.e. set one raster to *nodata* wherever a second raster is already *nodata*)
      i) Hint: comparative operators result in 1 for true or 0 for false. E.g. `5 < 0` is equal to `-1`
   d) You can use this for *weighted combination* of raster layers
   e) You can even use it for *reclassification* with the right formula setup

5) **GRASS r.Quantile**
   a) Calculates quantile statistics for an input raster and saves them as a simple `.html` webpage file
   b) With the *generate recode rules* option, the output table is actually a *reclassification* table for converting the raster to quantile classes
   c) Open the resulting html file and save it as a simple text `.txt` file
   d) droughtClip, 5 quantiles x Generate recode rules -> droughtQuantiles.html -> save as droughtQuantiles.txt

6) **GRASS r.Recode**
   a) Given inputs of a raster and a recode table generated by r.Quantile and saved as a text file, this tool will reclassify the raster into quantile classes.

7) Cautionary Notes:
   a) There is a GDAL Clip Raster by Mask Layer tool to clip a raster with a polygon. I found this extremely difficult (impossible) to use with complex polygons. Alternatively, convert vector data you want to use for masking into a raster (GDAL Rasterize) and then use Raster Calculator to apply a mask.
   b) There is a QGIS tool to *Align Rasters* available in the main menu → Raster menu. There does not appear to be a processing toolbox version of this, however, so I've kept to GDAL Warp for this functionality.

Critically Reflect

The final product this lab should be documentation of the data sources, the methodology (including those parts you completed yourself, and those parts we completed as a class), and your results. Include as much as you can discern about who produced the data, what time period the data represents, the coordinate reference system, extent, spatial resolution, and any information about uncertainty or sampling methods.

It should also include, to the best of your knowledge, an assessment of the reproducibility of this work with reference to our readings. In this regard, I suggest reviewing Malcomb et al from top to bottom.

1) Did your maps appear to give the same results as Malcomb et al? Why or why not?

2) Do you consider the research paper to be *reproducible*? In what ways did it succeed or not succeed in this regard? Work through the paper from beginning to end, following Tate's outline of approaches to geographic vulnerability analysis.

3) Do you think Malcomb et al adequately considered error and uncertainty in their paper? Support your position with specific examples.

4) Based on this experience and recent readings, what are some specific strategies you can use in order to become a better geographer doing research with GIS?

5) In order to be replicable, research should be rigorous enough in terms of its conceptualization (terms, definitions and relationships), data, and methodology, to produce similar findings in new contexts (e.g. different countries or time periods) with respect to the level of uncertainty inherent in the system of study. Are there problems of uncertainty inherent in vulnerability "systems of study" that would make it difficult to find similar results applying the same concepts and methods to new cases?

Lab Eight: Investigating Malawi Vulnerability with GIS Models

You're already asking great questions about reproducibility and uncertainty in Malcomb' et al's vulnerability model. Now, let's use GIS to start digging into some of those questions… culminating in a GitHub blog post answering the discussion questions above through your own documentation of the analysis.

All of the data, SQL code, and the model are packed in a single zip file on Canvas.

Data Sources

1) Download the global flood risk layer from UNEP Global Risk Map
2) Download the Malawi drought physical exposure layer from UNEP Global Risk Map
3) DHS Cluster Points, with info here: https://dhsprogram.com/What-We-Do/GPS-Data-Collection.cfm
4) DHS Survey Data, with information provided in Lab Seven.
5) FEWSnet Livelihood Zones: https://fews.net/fews-data/335
   (try re-typing the hyphen if this URL doesn't work)
6) Major Lakes, from OpenStreetMap via MASDAP: http://www.masdap.mw/layers/geonode:major_lakes
7) GADM version 2.8 Boundaries for Malawi: https://gadm.org/download_country_v2.html
   a) (taking an older version, but I should have probably gone back to version 2.7. Boundaries and boundary data change over time, and the authors haven't specified when they downloaded which version)
8) DHS Survey Region boundaries, which map to the districts in Malawi: http://spatialdata.dhsprogram.com/boundaries/#view=table&countryId=MW

GIS Analysis

Let's imagine that Malcomb et al had published in a more reproducible way. You'd be able to read their code and inspect their model to learn from their research process, right? Let's reproduce that: I'll provide you my model (`vulnerability.model3`) and you can use that to learn and critique the modelling process. Don't take the seeming ease of this for granted, though. Make sure you look into and understand what I'm doing at each step of the model.

1) First, Open QGIS ***WITH GRASS***

2) Second, make sure you have all the data sources above available to you. In addition, the processed adaptive capacity data is available on Canvas, based on the SQL code we developed collaboratively in Lab Seven.

3) Third, open the vulnerability model and finish the last steps for the model as-is:
   a) Look into all the model steps and discuss with partners to understand what the model is doing. Document the workflow for yourself. Some parameters are hidden within the 'advanced parameters' menu.
      i) Consider adding additional saved outputs from the model to see what the results look like at each step…
      ii) Notice that I manually coded the extent for GDAL Warp at one step and then forced other Warp or Raster creation steps to use the same extent as that layer. I did this to ensure that all the grids would be aligned by starting at the identical lower-left coordinates and having identical cell sizes.
   b) Run the model and save the outputs.

c) Use the **r.Quantile** and **r.Recode** algorithms as described above to reclassify the drought layer to a scale of 1 to 5

d) Do you think the flood risk layer needs to be reclassified? Or is there an easier solution for this input?

e) Use the Raster Calculator to combine the final layers according to Malcomb. et al's weighting:

   i) Adaptive capacity – 40%

   ii) Drought exposure – 20%

   iii) Flood Risk – 20%

   iv) Hint: we need to invert the Adaptive Capacity score, because currently a high score for capacity correlates to a low vulnerability. You can invert the score by subtracting it from its maximum. The maximum possible score is 2 (40% of 5), so take `(2 - adaptive capacity).`

4) Clearly the model is producing a map at a coarser resolution than that of Malcomb et al.

   a) Save an alternative version of the model and change it to more closely match Malcomb et al by making a map with 2.5 minute resolution (`0.04166666` decimal degrees) rather than a 5 minute resolution (`0.08333333` decimal degrees).

   b) In case you want to flexibly model results at different resolutions, could you add a parameter to the model in which the user could input the cell size they prefer for the final results? Try…

      i) Pay attention to the resampling method for each Warp tool… for Flood use a method that will preserve the discrete (classified) input of only integers from 0 to 5, and for Drought Exposure use a method that will preserve the continuous nature of the input data.

5) Final project idea:
   The DHS Cluster points are randomly displaced, and we'd like to model the uncertainty in their possible assignment to different traditional authorities. You may do this either with SQL code in your database or with a QGIS model.

   a) Displacement is up to 2km in Urban areas and up to 5km in Rural areas.
      (let's not consider the 1% possibility of a 10km displacement at this time)

   b) They are *not* displaced across *district* boundaries.

   c) Can you determine how many traditional authorities each cluster may *actually* be located in, and the probabilities of the TA's being located in each?

   d) Can you determine the *probability* of each cluster's location within a given traditional authority?

   e) Hint: the GADM district names (level 2) associated with each traditional authority (level 3) are not identical to the DHS region names, but it seems like the centers of the traditional authorities are consistently located within the correct DHS district…

   f) The simplest answer is to visualize how many traditional authorities each point could potentially be located in.

   g) A better answer is to *calculate* and visualize how many traditional authorities each point could be located in using the size of the point as a visual variable.
      The best answer is to calculate the *probability*, based on the percentages of area, of each cluster to traditional authority combination. With this information, it'd be possible to *randomize* the household capacity calculations or to *weight* the household adaptive capacity analysis according to those probabilities.

Final Lab: Sharpie versus Storm Surge in the Twittersphere of Hurricane Dorian

Final Products

The deliverable for this lab is to create a GitHub page detailing your analysis of Twitter data on Hurricane Dorian. The page should include:

1) Analysis of twitter activity over time during the week of the storm
2) Analysis of most common language / keywords in tweet content during the storm
3) Analysis of association of common keywords in tweet content during the storm
4) Heatmap (Kernel Density) of tweet activity during the storm, based on tweet locations
5) Spatial hotspot analysis (Getis Ord G*) of tweets per 10,000 people during the storm by county
6) Spatial hotspot analysis (Getis Ord G*) of a normalized tweet difference index by county: (tweets about storm – baseline twitter activity) / (tweets about storm + baseline twitter activity)
7) As usual, document your analysis as thoroughly as possible through code and comments or written text. Do not publicly provide twitter data: but for maximum reproducibility you may include twitter status ID's used in your final analysis.
8) Interpret your work and findings with reference to our readings. Which factor seems to drive more Twitter activity? Fake sharpie maps or real hurricane paths?

Data Provided

1) Find the dorianForLab.RData data file for R on `K:\gg323` This R data file contains two data frames (tables)
   a) `dorian` (tweets presumably related to the storm)
   b) `november` (baseline twitter activity for a week in November)

2) The R script used to create these data frames is available as `dorianTwitterScript.R`

First Steps in R

1) Use code from the previous lab to upload the twitter data into your PostGIS database

2) Use code from the previous lab to download county-level geographic and population data from the U.S. Census and upload all the counties into your PostGIS database.

3) Part of the lab is to conduct textual analysis of the tweets in R, but you already know how to do that, so I suggest saving your R work and returning to that familiar territory last.

Spatial Analysis in PostGIS

Through QGIS and DB Manager, prepare twitter and county data for spatial analysis and count tweets by County with a spatial join.

1) Add a projected coordinate system to your PostGIS database
   a) We need to use a projection consistent for analysis of distance and area over Eastern United States. Although it's not perfect, the USA Contiguous Lambert Conformal Conic projection is a reasonable choice. Its SRS code is `102004`, and it does not currently exist in the database's `spatial_ref_sys` table, although it exists in QGIS.
   b) Search for this projection at https://www.spatialreference.org
   c) Copy it's PostGIS INSERT statement for adding it to a database
   d) The statement has one error: a `9` has inexplicably been added to the first CRS id. Remove that 9 and run the code.
   e) Confirm that the insert worked with a query:

```
Select * from spatial_ref_sys where srid = 102004
```

2) Create a point geometry for each set of twitter data, while transforming the point into the Lambert Conformal Conic CRS
   a) Use the `AddGeometryColumn()` function to add a `geom` column to each table of twitter data
   b) Use an `UPDATE` query to calculate the geometry, first making a point, then setting it's SRID to WGS 1984 (the system Twitter uses), and finally transforming the point to the Lambert Conformal Conic projection. For example:

```
UPDATE dorian
SET geom = st_transform( st_setsrid( st_makepoint(lng,lat),4326), 102004)
```

3) Update the geometry data for counties by transforming it into the Lambert Conformal Conic projection
   a) The query might look like this:
      UPDATE counties SET geometry = st_transform(geometry,102004);
   b) Register the geometry data for counties with the `populate_geometry_columns()` function
   c) Note: you can only get away with this if the geometry column has not yet been registered. If it has, just create an additional column for the transformed geometry using `AddGeometryColumn()`

4) Delete the states we are not interested in using a trick: `NOT IN ()` where the state ID is not in a list of ID's enclosed by parenthesis.
   a) You can figure out the list of counties by loading counties in QGIS, selecting counties from the area you're interested in, and running the `List unique values` algorithm.
   b) Or, just use this query:

```
DELETE FROM counties
WHERE statefp NOT IN ('54',   '51', '50', '47', '45', '44', '42', '39', '37',
      '36', '34', '33', '29', '28', '25', '24', '23', '22', '21', '18', '17',
      '13', '12', '11', '10', '09', '05', '01');
```

5) Count the number of each type of tweet by county
   a) Add text columns of length 5 to each of the twitter layers, e.g.:

```
ALTER TABLE november ADD COLUMN geoid varchar(5);
```

   b) Set the new `geoid` columns equal to the `geoid` of the county they intersect
   c) Group the twitter layers by GEOID while counting the number of tweets
   d) Add integer columns to the counties data for Dorian's tweets and November tweets.
   e) Set the new integer columns equal to `0`
   f) Set the new integer columns equal to the count of tweets by county

6) Normalize the Twitter data
   a) Create a new column in counties of type real for calculating the tweet rate
   b) Calculate the number of tweets per 10000 people
   c) Create a new column in counties of type real for calculating the ndti (normalized tweet difference index)
   a) Calculate the ndti = (tweets about storm – baseline twitter activity) / (tweets about storm + baseline twitter activity)

Spatial Hotspot Analysis with GeoDa

1) Open an open-source GIS program for spatial statistics: GeoDa
2) Connect to your PostGIS database to load your counties table
3) Create a spatial weights matrix
   a) Go to **Tools** → **Weights Manager**
   b) Create a new weights matrix
      i) **weights file id variable**: `geoid` (this is asking for a unique ID)
   c) set a **Threshold distance**: leave at default so every county has at least one neighbor.
   d) **Create**, and wait for the matrix creation to complete

4) Calculate the local G* statistic
   a) Go to **Space** → **local G* cluster map**
   b) Set the variable to your tweet rate column
   c) Include a significance map. cluster map, and row-standardized weights

5) Experiment with the importance of choosing a statistical significance level
   a) Try right-clicking on the hot-spot / cold-spot map and changing the significance filter to see its effect

6) Save your results
   a) Right-click the hot-spot / cold-spot map and save results, *changing variable names to all lowercase*
      i) This adds columns to the temporary table in GeoDa, but does not save a permanent copy of the data yet…
   b) Go to **file** → **save as**
      i) switch to the Database tab
      ii) save as a new table, e.g. `countieseastg`

7) Note: this analysis can be accomplished in R as well, with the `spdep` package. Learning how to implement this would make for a good final project.

Heatmap (Kernel Density) Visualization of Tweets

1) convert counties into centroid points
2) In QGIS, run the Heatmap (Kernel Density Estimation) algorithm
    a) Set the radius to `100 kilometers` (allowing some overlap between adjacent counties)
    b) Set **Weight from field** to the tweet rate column
    c) Set the pixel sizes to `500` (meters)
    d) These are arbitrary selections, guided by the desire for continuity between data points (choosing a radius larger than typical distance between points) and a smooth visualization that will not run too long (determined by the cell size. 100 meters computes very slowly, 1000 meters is fast but looks course, and 500 meters is for Goldilocks)