Take home: Challenge prompt

Challenge Description

The objective of this exercise is to create a dataset in the form of a heat map representing the likelihood of finding a cobalt deposit at each point on the map. The heat maps should be stored in such a way that we could easily view the numeric value for the likelihood of finding cobalt at each and every location.

Tasks

To determine where cobalt is likely to occur, we are going to assume that cobalt deposits are found in the presence of two different types of rock. We are interested in locations where these rock types are in contact with each other, or in close proximity to one another. The likelihood of cobalt occurring should fall off smoothly to zero where distance between the two rock types exceeds some distance (~10 km).

1. Build a software tool capable of creating heat maps based on the proximity of any two rock types, with an adjustable fall-off distance parameter.
2. Generate a heat map based on proximity of **serpentinite** and **granodiorite.** Serpentinite is a subtype of “ultramafic” rock; so here, for the purpose of this exercise, assume that all **ultramafic** rocks are also serpentinite.

Data

We have provided a set of data that shows the type of bedrock (what lies beneath the soil) at each location in an example region. This data comes from the British Columbia Geological Survey. The coordinates in this dataset are provided in the coordinate reference system (CRS) EPSG :26910 which has units meters. Allowing distances between points to be calculated in meters without any conversions.

The map is provided here as a shapefile (BedrockP.shp), which is a proprietary data format associated with one particular GIS program, but it is more widely used for storing geospatial information and there are numerous free packages available for importing shapefiles and manipulating geospatial data in python. We have found the python packages *geopandas* and/or *shapely* to be helpful, as well as the PostGIS extension to PostgreSQL. BedrockP.shp consists of a set of polygons that collectively cover the area and a table of attributes associated with each polygon.

Getting started

To get you started, here are some code extracts (using geopandas) that will help you load and view the shapefile data.

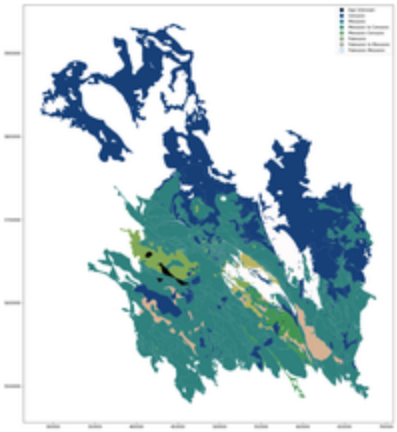
Loading the data:

```
import geopandas as gpd

INPUT_FILE = "BedrockP.shp"
bedrock_data = gpd.read_file(INPUT_FILE)
print(bedrock_data).head()
```

Visualizing the data:

```
bedrock_data.plot(column="era", categorical=True, legend=True, figsize=
(20,20), cmap='gist_earth')
```



Saving the outputs:

Matplotlib's savefig is useful for saving outputs and avoiding potential issues with rendering the map in an interactive window.

https://matplotlib.org/stable/api/_as_gen/matplotlib.pyplot.savefig.html

If you are having trouble setting up your environment, please contact us; installing packages like geopandas and their dependencies isn't meant to require a significant amount of time.

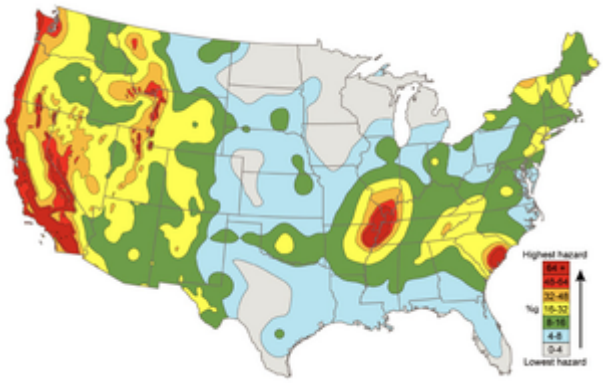
What to submit to us

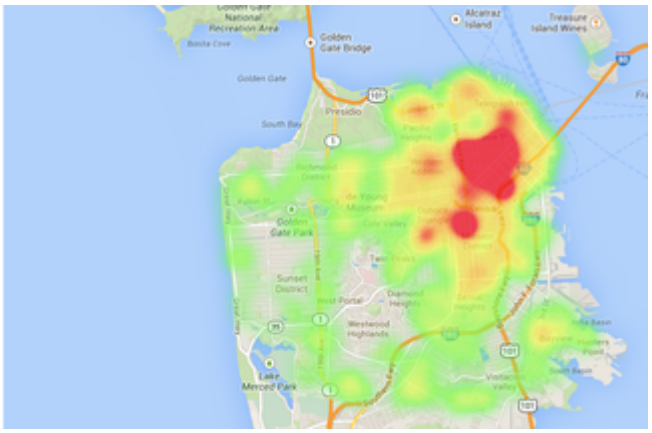
1. A **heat map**, which shows the likelihood of finding cobalt at any specific location, based on the above logic. The heat map should show the highest likelihood of finding cobalt along the interfaces of serpentinite and granodiorite, and should fall smoothly to zero at some distance away (between 5-10 km for this task). Even if serpentinite and granodiorite never touch, the heat map should be non-zero where they are close together.

A static map or screenshot is fine. If you choose to create an interactive map, please also include a static image so we can see the output as you see it when running the script (anything that can be easily opened in a browser is fine too).

Note: If a polygon representing serpentinite is touching a polygon representing granodiorite, it doesn't mean that the whole polygon is a good place to find cobalt; only the part on or near the interface of the two polygons should be considered prospective, regardless of the type of rock at those locations.

If you are unfamiliar with heat maps, below are some examples:





2. All **code** written to create the map. Please use Python for this task.

3. A README file with instructions for running your code along with a brief description of how you solved the problem.

What we're looking for

- *Correct answer*: the heat map should clearly and accurately reflect the regions that best fit the criteria and should be in a format that permits access to the value shown on the heat map at each location.
- *Clear communication/documentation*: how you approached this problem and the steps you took to get to the answer, including whatever data exploration you did to formulate your answer.
- *Code quality*: The code submitted does not need to be production ready but should be clean, and easy to follow, understand and use.