Technical Challenge

Overview

This technical challenge is designed to take less than 2 hours, however you will have 4 hours to return it in case you want some extra time. Please send us an email with your results as a link to the Github repository storing your script(s) and files within 4 hours of receiving the email from us. Late submissions may be disqualified.

Goal: We would like you to create a small pipeline that: (1) reads in data from the four csv files provided, (2) does some basic data manipulations and aggregations (see below for specific instructions for each dataset), and then (3) writes the transformed data into four corresponding csv files. You will be evaluated on the functionality, readability, and overall architecture of the pipeline that you create.

Please use Python and make use of any Python libraries that you see fit. The final GitHub repository should have three folders: `inputs`, `scripts`, and `outputs`. Load the four csv files that we provided into `inputs`, your py file(s) into `scripts`, and the resulting transformed data csvs into `outputs`. Include a short readme with instructions on how to run your code, as well as a pip file for full reproducibility. The entire pipeline should run from a single command line call, such as `python your_script.py [INSERT ARGS]`. A failure to provide clear instructions/environment variables/library imports so that we can run your code to generate your outputs disqualifies the submission.

Bonus: If you complete the above in less than 2 hours, you may additionally include in your pipeline an extra step that merges the four transformed datasets into a single dataset that is then written to a fifth csv file. We would like this merged dataset to be at the field-level, meaning that each row is uniquely identified by a `field_id` and all data attributes have been aggregated such that they represent field-level metrics that might describe interesting dynamics on that field. It is important to note that there is no foreign key that easily links all of the datasets together; instead you will have to use the provided spatial information (all geometries are given in the same coordinate reference system (CRS) where CRS = 4326) to join and aggregate the datasets in such ways that you see fit. You may do any exploratory data analysis that you wish to inform your methodology, but we also suggest taking into account the information below:

- Each field (`field_id`) can be split up into tiles (`tile_id`) that are discrete and have no overlap; these will be neatly contained within a field
- Each field (`field_id`) can be split up into map units (`mukey`) that are discrete and have no overlap; these may expand outside a field

If you choose to take on this bonus task, please also include in the readme a description of the structure of your merged dataset/fifth csv file and your methodology for this part of the project (this shouldn't be more than a paragraph or two). **Note that you are not required to complete**

the bonus; you may accomplish the technical challenge by successfully completing just the first part.

Datasets

(1) crop.csv - this dataset provides crop type for a given field and year

Table 1. Data dictionary for crop.csv

Column	Description
`field_id`	Field identifier
`field_geometry`	Text representation of geometry (CRS = 4326)
`year`	Year identifier
`crop_type`	Crop type in that field for that year

Task: Extract the data from the year 2021.

Desired data: crop type by field for the year 2021. Please save as a csv file with the following headers: `field_id, field_geometry, crop_type.`

(2) spectral.csv - this dataset provides two spectral reflectance bands (from satellite imagery) for a given tile and date

Table 2. Data dictionary for spectral.csv

Column	Description
`tile_id`	Tile identifier
`tile_geometry`	Text representation of geometry (CRS = 4326)
`date`	Date when image was taken
`nir`	Near-infrared band (832.8 nm)
`red`	Red band (664.6 nm)

Task: First compute the normalized difference vegetation index (NDVI; see formula below). Using your output, then find the peak-of-season (POS; maximum value observed over the year) and date that the POS occurred.

NDVI formula: `(nir - red) / (nir + red)`

Desired data: POS value and date of POS (for NDVI) by tile for the year 2021. Please save as a csv file with the following headers: `tile_id, tile_geometry, pos, pos_date.`

(3) soil.csv - this dataset provides soil attributes (om, cec, and ph) for a given hzname-cokey-mukey. Note that there are multiple horizontal layers (hzname) contained within each component (cokey), and multiple components contained within each map unit (mukey)

Table 3. Data dictionary for soil.csv

Column	Description
`mukey`	Map unit identifier
`mukey_geometry`	Text representation of geometry (CRS = 4326)
`cokey`	Component identifier
`comppct`	Percent that component makes up for that mukey
`hzname`	Horizontal layer identifier
`hzdept`	Top depth of that horizontal layer
`hzdepb`	Bottom depth of that horizontal layer
`om`	Soil organic matter (OM)
`cec`	Soil cation exchange capacity (CEC)
`ph`	Soil pH

Task: First compute the weighted average of horizontal layers for each component (see formula below). Using your output, then compute the weighted average of components for each map unit (using `comppct`).

Horizontal layer weights: `abs(hzdept - hzdepb) / hzdepb`

Desired data: soil attributes (om, cec, and ph) by mukey. Please save as a csv file with the following headers: `mukey, mukey_geometry, om, cec, ph.`

(4) weather.csv - this dataset provides average temperature and total precipitation for a given fips code and month-year

Table 4. Data dictionary for weather.csv

Column	Description
`fips_code`	FIPS identifier for state and county (first two digits are state identifier; last three digits are county identifier)

`year`	Year identifier
`month`	Month identifier
`precip`	Total precipitation in that county for that month
`temp`	Average temperature in that county for that month

Task: Extract the data from the year 2021 and then compute the total (summed) precipitation and minimum, maximum, and mean temperature observed over the year for each FIPS code. Using your output, then map the provided `fips_code` to the correct `field_id` using the `field_geometry` given in the crop.csv file. You may find the following code useful where the latitude and longitude can be extracted from the center point of the `field_geometry`:

```
url = 'https://geo.fcc.gov/api/census/block/find?latitude=%s&longitude=%s&format=json' % (lat, lon)
response = requests.get(url)
data = response.json()
state = data['State']['FIPS']
county = data['County']['FIPS'][2:]
```

Desired data: total precipitation and minimum, maximum, and mean temperature by field for 2021. Please save as a csv file with the following headers: `field_id, precip, min_temp, max_temp, mean_temp.`