Exercise 2

1. Merge all data sets

Start a new Python script on your local machine. You can also work on the FASRC, e.g., using a Jupyter Notebook. In this case, all case study related data is publicly available at
`/n/holyscratch01/cga/python-workshop-gis-big-data`
However, working in your go-to environment might be faster for you.

If you are working on your own device, ensure to have a copy of all data set files ready. Also, create a new Python environment, providing the same packages as we installed in Chapter 1. You can use the same `requirements.txt` file: https://raw.githubusercontent.com/cga-harvard/python-workshop-gis-big-data/main/Chapter%201/requirements.txt

Now, create a new Python file. This version of the exercise sheet won't provide solutions to you. However, if you want some inspirations, you can read the `Exercise 2 with code` version. But first, try to solve the challenges yourself.

```
#!/usr/bin/env python

import pandas as pd
import numpy as np
from datetime import datetime
import xarray as xr
```

1.1 Load the TSGI dataset

To load a CSV file, use Pandas' `pd.read_csv(filename)` function. This reads the file and returns a Pandas data frame (DF) object.

For better readability, I suggest to rename the columns of the file, e.g., to

```
['date', 'country', 'state', 'county', 'sentiment_score', 'tweets']
```

Lastly, state and county names are concatenated, e.g., `United States_New Mexico_Torrance`. To have only the state name and only the county name in the corresponding columns, extract the relevant names using string-processing.

Now, select a few states for our toy example. This limitation can later easily be removed, but for now it allows us to work with a much more manageable subset.

```
state_subset = ['Massachusetts', 'Connecticut', 'Rhode Island']
```

If you now inspect your subset, it should contain 9671 rows. Let's add geo coordinates to the table. As discussed, the geocoding was already done and can be loaded as a separate DF.

To merge this new DF with your existing tweets subset, employ Pandas' merge function.

Ensure you only consider rows that have a lat & lon value, and drop all rows for which the merge wasn't successful. If you now inspect your DF variable, the DF should still have 9671 rows, but reach row should have two more columns: lat and lon coordinates.

1.2 Load the NOAA CPC dataset & augment the tweets DF

To load the NOAA CPC dataset, we use the `xarray` package that provides functionalities specific to the NetCDF data format. The code is very similar to what we're used to when working with the Pandas library:

```
        noaa_cpc_dataset = xr.open_dataset(filename)
```

Now we have access to the precipitation values through the `noaa_cpc_dataset` object. While the object also stores a lot of meta data, the `noaa_cpc_dataset.precip.values` provides the three dimensional array with the `[day, latitude, longitude]` dimensions.

However, the coordinates of our counties don't line up perfectly with the 0.5 x 0.5 grid of the NOAA CPC dataset. Hence, when we want to augment the tweets with precipitation data, we need to find the closest NOAA CPC coordinates per county. Also, the tweets come with a date in the format "2022-05-27" to indicate the day of the year, while our NOAA dataset expects a single value in `[0, 364]` (note: zero-based numbering) for the day of the year.

If you want some inspirations how to do this, have a look at the `Exercise 2 with code` file. But try it first yourself.

2. Analyze the results

To investigate our research question, we can now – that we have all relevant information in one table – analyze the results. We seek to contrast all the days of rain against all the days of no rain, but only per county.

To only count rainy days with a significant amount of rain, let’s define

```
    RELEVANT_PRECIPITATION_THRESHOLD = 12 * 2.5
```

This is equal to 12h of 2.5mm rain, typically what is considered as medium intense rain.

If you want some inspirations how to do this, have a look at the `Exercise 2 with code` file. But try it first yourself.

What you learned in this exercise:

- How to load CSV and NetCDF files into Python
- How to merge DFs using Pandas
- How to filter DFs and work with `groupby()` and `apply()`