Spatial and Temporal Data Mining

1 Spatial

1. Spatial Clustering
   (a) Load datasets `US_city_T.csv` and `US_City_Locations.csv` from the website.
   (b) Considering city temperatures as features, compute a clustering in 3 clusters based on `AgglomerativeClustering` from sklearn.
   (c) Use library `folium` to create an interactive map in which cities are represented by dots, and colors correspond to cluster. Can be something like

Listing 1: plot on a map

```
import folium
colors = ['red', 'blue', 'green']
map_center = [df_locations['Latitude'].mean(), df_locations['Longitude'].
    mean()]
folium_map = folium.Map(location=map_center, zoom_start=3)

# Add circle markers to the map
for i, row in df_locations.iterrows():
    folium.CircleMarker(
        location=[row['Latitude'], row['Longitude']],
        radius=6, # Size of the circle marker
        color=colors[df_clustered2['cluster'][i]], # Color based on the
            cluster
        fill=False, # Enable fill
        fill_color=colors[df_clustered2['cluster'][i]], # Fill color
        fill_opacity=0.6, # Opacity of the fill
        popup=row['City'] # Popup to show city name
    ).add_to(folium_map)

# Display the map
folium_map
```

   (d) Observe that some clusters are not contiguous
   (e) Enforce a spatial clustering by 1)Computing a connectivity matrix using sklearn's `kneighbors_graph`, 2)Using this connectivity matrix in the `connectivity` parameter of the `AgglomerativeClustering` method

2. Spatial autocorrelation
   (a) Create a new dataframe in which each city has a single feature, its average temperature (e.g., `df.mean`
   (b) Using the connectivity matrix, compute the spatial lag for each point, i.e., the (weighted) average of values. You can for instance tranform it into a networkx graph with `nx.from_scipy_sparse_array(connectivity_matrix)`.
   (c) Draw Moran's plot. Don't forget to standardize the data
   (d) Color nodes according to their category: HH, LH, HL, LL.

(e) Use these categories to color points on the Map. What are the ”hot spot”, the ”diamonds”?...

2 Temporal Data

1. Temporal Autocorrelation
   (a) Using plotly (e.g., `px.line`), draw the time series for `new_york` and `los_angeles`
   (b) Using `statsmodels.tsa.stattools.acf`, compute the Autocorrelation Function for these cities. Interpret to check that there are seasonal patterns, as expected.
2. Clustering
   (a) Compute a clustering of cities with similar temporal patterns (note that it is similar to what we did in the spatial part). Due to the seasonality, and equal length, time warping is not relevant here.
   (b) Plot the average time series for each cluster.
3. Seasonal Decomposition
   (a) Using `statsmodels.tsa.seasonal`, perform a classic seasonal decomposition. You need to choose an appropriate time period. For instance, you can check with 1 year (12 timesteps), and 10 years
   (b) Draw the trend timeseries (`decomposition.trend`). Interpret: is the average temperature stable over the period? Does it change a lot?
   (c) Draw the seasonal series(`decomposition.seasonal`)
   (d) Finally, draw the residuals (`decomposition.resid`)
   (e) Now, look at the trend as a new time series. Is there some meaningful structure in it, i.e., multi-annual cycles...? Try an ACF, and another decomposition...

3 Going Further

Such data can be used to study climate change. When removing the effects of seasons, what remains in the trend is the evolution of climate over the years. Would you say that the climate is stable over this period or not? This is called **stationarity**. Search for a stationarity test to use on your data. Now, are all cities following the same climate evolution trends, or some cities are more affected than others? Controlling by the average temperature, which is a fonction of the position, are there clusters of cities for which the climate evolve similarly? Are these clusters the same than taking into account the average temperature and seasonal patterns?