# Geospatial Data Analysis Scripts

This document outlines the descriptions of three required scripts used in the analysis of the data: sampling, mapping, and clustering.

## 1. Sampling Script

The sampling script will convert the data to a spatial format and compute distances along the road and sample the data every 10 meters, reducing duplication.

**Script Functionality:**

**Loading Libraries:**

- Using **dplyr** for data manipulation and **geosphere** for distance calculations between geographical coordinates.

**Data Reading and Preparation:**

- Reading the data from a CSV file and renaming columns for easier reference.
- Converting the timestamp column to POSIXct format, which is more suitable for handling time data in R.

**Sampling Function:**

- The `sample_at_equal_distances` function iterates through the dataset, organized by time. It starts with the first data point and iteratively checks the distance to subsequent points using the Haversine formula (provided by `distHaversine`). It adds a point to the sampled dataset only if the distance to the last included point is at least as large as the specified sampling distance (10 meters in this case).

**Data Output:**

- The function is called to sample the data at 10 meters.

---

## 2. Mapping Script

**Script Workflow:**

**Load Required Libraries:**

- **dplyr**: For data manipulation.
- **leaflet**: For creating interactive maps.
- **gstat**: For geostatistical modeling including interpolation.
- **sp**: For dealing with spatial data.

**Data Preparation:**

- Reading sampled data from the CSV file.
- Renaming columns for consistency.
- Converting the data frame into a spatial object (sp package) using geographic coordinates (longitude, latitude).

**Interpolation Setup:**

- Determining the bounding box of the data to define the area for interpolation.
- Creating a grid (resolution determined by `by = 0.001` in both latitude and longitude) which will be the basis for interpolation.

**Interpolation Process:**

- Using Inverse Distance Weighting (IDW) to interpolate pressure values over the defined grid. This method weights the input points such that the influence of one point decreases with distance.
- Converting the interpolated SpatialPixelsDataFrame back into a regular data frame for visualization.

**Visualization with Leaflet**

- Constructing an interactive map using leaflet.
- Adding base tiles (default OpenStreetMap).
- Adding circle markers to represent interpolated pressure values.
- The color of these markers is determined by a color gradient from blue to red, representing low to high pressure.
- Including a legend to help interpret the pressure values visually on the map.
- Implementing pop-ups to display pressure values upon interaction with each marker, providing instant data access.

**Saving the Output:**

- The map is saved as an HTML file using `htmlwidgets::saveWidget`, which can be viewed in any web browser.

---

## 3. Clustering Script

**Script Explanation:**

**Library Loading:**

- **dplyr**: Used for data manipulation tasks such as selecting and renaming columns.
- **ggplot2**: Provides comprehensive plotting capabilities to visualize data, especially for creating the cluster plot.
- **sf**: Employed to handle spatial data in R.
- **cluster**: Provides clustering algorithms, including k-means, used in the script.
- **png**: Necessary for reading and displaying PNG files directly in R, which is used to show saved plots within the R environment.

**Data Preparation:**

- Data Loading: Reading the CSV file containing sampled atmospheric pressure data.
- Renaming Columns: Ensuring column names are consistent and understandable which aids in easier scripting.
- Timestamp Conversion: Converting the timestamp column from character type to POSIXct, which allows for date and time operations if needed later in the analysis.

**Clustering Process:**

- Setting Seed: Ensuring reproducibility in clustering results by setting a seed for the random number generator used in k-means.
- Coordinates Selection: Extracting longitude and latitude from the data for use in clustering. Clustering here is based on geographical location, meaning similar positions will be grouped together.
- K-means Clustering: Applying the k-means algorithm to the coordinates with a specified number of clusters (5 in this case). K-means tries to minimize variance within each cluster, effectively grouping data points that are geographically close to each other.

**Cluster Integration and Visualization:**

- Cluster Assignment: Adding the cluster identification for each data point back to the main data frame, allowing for easy tracking and visualization.
- Plotting: Using ggplot2 to create a scatter plot of the data points colored by their cluster ID, which visually represents the geographic grouping.
- Plot Aesthetics: Enhancements like setting the plot title, applying a minimal theme, and adjusting the color scale make the plot more informative and visually appealing.

**Cluster Validation Using Silhouette Scores:**

- Computing Silhouette Scores: After performing the k-means clustering, we use the silhouette function from the cluster package to compute the silhouette scores for each point relative to the clusters. This helps assess how well each object lies within its cluster; a high silhouette value indicates good internal cohesion and poor matching with neighboring clusters.
- Visualizing Silhouette Scores: Plotting the silhouette scores can help visually assess the quality of the clustering. This is crucial for validating the appropriateness of the number of clusters chosen and can guide adjustments to improve clustering outcomes.

**Saving Outputs:**

- Plot Saving: The plot is saved as a PNG image.
- Data Saving: The data with cluster assignments is saved back to a CSV file. This is useful for further analysis or verification of clustering results.