COSC3337 Group Project

POIMAGIC: Early Warning System

Saim Ali | smali35@uh.edu James Aquino | jgaquino@uh.edu
Justin George | jjgeorg4@uh.edu Johan John | jnjohn3@uh.edu

Preface

This report pertains to the group project for COSC3337 - Data Science 1. In this report, we will present our group's findings on our POIMAGIC implementation on the earthquake location data given to us. Throughout the project, all members contributed significant amounts of work in the manner of coding and crafting our methodology. Additionally, all members contributed equally to the report. We used the relatively new data science notebook Deepnote in order to work together on the project. Deepnote allowed us to place a notebook on top of a Docker file denoting an anaconda environment while at the same time allowing us to work together.

Introduction

Our task was to create an Early Warning System Animation called POIMAGIC for streaming spatial events based on Earthquake hotspots. We were also told to consider building our project in such a way that it can be reused to detect earthquakes in a different region with different amounts of available data. Given the task we determined that the best method to create POIMAGIC would be a pipeline function with various smaller functions built in it. The pipeline function would take in the two density thresholds along with the input data and provide a list of resulting hotspot images.

```
def pipeline(Dict, epi, mins):
    for key,DataFrame in Dict.items():
        print("Set ",key)
        cluster_list = dbscan(DataFrame,epi, mins)
        print("Set ", key, " information")
        print("Number of hotspots :" , len(cluster_list)-1)
        size = len(cluster_list)-1
        for i in range(size):
            print("Hotspot Center (Lat, Long): ", Center(cluster_list[i]))
            print("Spread of Hotspot (Var of Lat and Long pts of the hotspot)", DistCenter(cluster_list[i]))
            print("Avg Magnitude of Earthquakes in the hotspot: ", MagPerCL(cluster_list[i]))

        dbscanplot(DataFrame,epi, mins)
```

Figure 1: Pipeline function

Input

As input we were given 12 sets of earthquake data that occurred over a select region of the earth for over a year, and we were told to organize them into 10 sets of data that would serve as the input into the program. Each consecutive set takes in data from one new month and two of the preceding months. This was done so that in theory clusters from the previous dataset will repeat in the new dataset since two-thirds of the data of every dataset is a repetition of the previous dataset.

Since this preprocessing step would be unique for every use of the project, we decided to keep it out of our pipeline function. Instead, we would feed in the processed sets directly. To do this efficiently in one for loop, we decided to use a dictionary as the vessel to store and utilize the newly processed sets.

```
Dictionary = dict()
Dictionary = { 1: Set1, 2:Set2, 3:Set3, 4:Set4, 5:Set5, 6:Set6, 7:Set7, 8:Set8, 9:Set9, 10:Set10 }
```

Figure 2: Dictionary of the Sets we created encompassing a timeframe of three months each

Input Values

The dataset has 5 attributes. Their range is given in brackets.

1. *Time*. In UTC. Format: YYYY-MM-DDTHH:MM:SS.000Z. Indicate the date and time when the earthquake initiates rupture, which is known as the "origin" time. Note that large earthquakes can continue rupturing for many 10's of seconds.
2. *Latitude*. [-90.0, 90.0] Decimal degrees latitude. Negative values for southern latitudes. Coordinates are given in the WGS84 reference frame
3. *Longitude*. [-180.0, 180.0] Decimal degrees longitude. Negative values for western longitudes. Coordinates are given in the WGS84 reference frame.
4. *Depth*. [0, 1000] Depth of the event in kilometers.
5. *Mag*. [-1.0, 10.0] The magnitude for the event. Commonly a moment magnitude that is based on the scalar seismic-moment of an earthquake determined by calculation of the seismic moment-tensor that best accounts for the character of the seismic waves generated by the earthquake.

Figure 3: Attributes of the earthquake data provided to us

The picture above lists the five attributes of the dataset. We determined the primary attributes to create the hotspots would be the *latitude* and *longitude* values. *Longitude* and *latitude* can be plotted in a 2d longitude-latitude plane and be used as the basis of our classification method. We also planned to use the *Mag* attribute in our pipeline since our research on geothermal hotspots gave us an interesting find: locations with frequent geothermal activity tend to produce earthquakes with a lower magnitude while other locations with less frequent earthquakes produce earthquakes with a larger magnitude. By understanding the average magnitude of earthquake events at the hotspot, we get an understanding of the future behavior of the hotspot.

The two attributes we decided to ignore were the *time* attribute and the *depth* attribute. We chose to ignore the *time* attribute since the preprocessing step will already organize the dataset in consecutive months and the only other use of this attribute would be in a time series prediction task which would be beyond the scope of the project. We ignored the *depth* attribute since it would not contribute to the project as we could not find any meaningful relationships in our limited research which utilized the *depth* of the earthquake events.

Clustering the Datasets

The animation requires us to plot the hotspots as a contiguous polygon in the longitude-latitude plane. The polygon can be defined as a shell placed around the events that occur in the hotspot. Given this definition we decided to first classify the different hotspots as a series of clusters and then convert those clusters into the contiguous polygon. DBSCAN is a clustering method that uses a core point and the parameters for the radius around said point, epsilon, and a minimum number of points inside the radius, minpoints, to classify dense regions which are combined to form clusters. This algorithm would be perfect for the task since it allows us to find clusters that will

satisfy our minimum density threshold and the algorithm is naturally resistant to noise.

The only issue we saw with this approach would be the fact that it does not retain the position of the cluster between the different datasets. An alternative we proposed was to first classify using DBSCAN to get the cluster and then load the center of these clusters into KMEANS to classify the rest of the sets. This idea unfortunately failed because KMEANS kept collecting noise and it did not recognize when a new hotspot cluster was detected.

But this failure inspired our group to merge the strengths of both methods when creating our pipeline. We decided to classify the datasets using DBSCAN, removed the outliers and then printed out the centroid of each cluster along with various other observations. Since two-thirds of each dataset is retained from the previous one, it became fairly easy to find out which information belonged to which cluster.

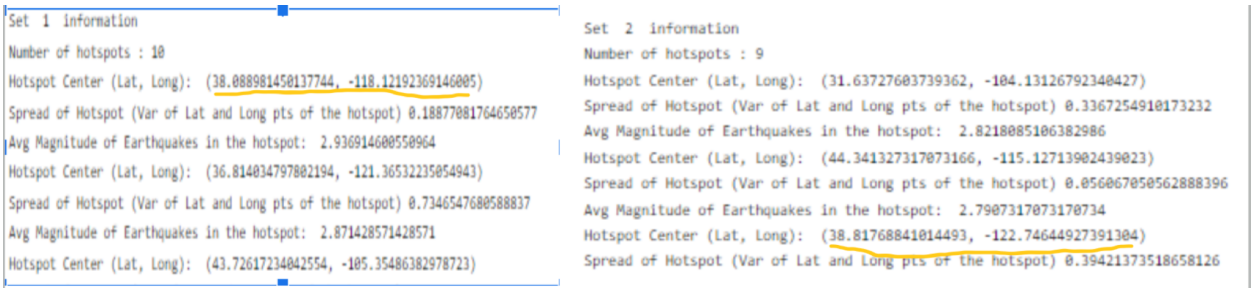


Figure 4: Shows how clusters translate between different datasets

Density Thresholds

The density thresholds were found by looking at the different clusters produced by DBSCAN and arbitrarily assigning what is a high density and what is a medium high density. To get good clusters for DBSCAN we tinkered around with the epsilon and minpoints values. After many experiments we decided the best results came when we set the epsilon to 0.15 and minpoints to 50. We set the first density threshold d1 to 200 and the second density threshold d2 to 80. We made two functions to find hotspots. The functions work by first taking in

a dataset and getting the clusters by running the DBSCAN function on them. Then it loops through the list of clusters. If a cluster's size is greater than a density threshold, that cluster is put into a dataframe and returned. findHighHotspots is used to find clusters with a density threshold greater than d1. findMediumHighHotspots is used to find clusters with density threshold greater than d2 but less than d1.

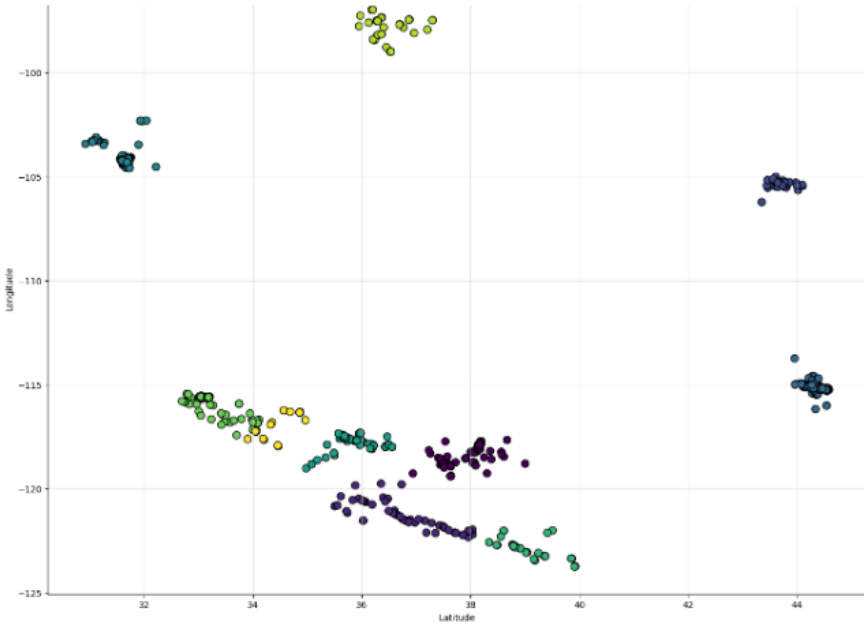


Figure 5: DBSCAN Generated cluster example

Getting Information From the Clusters (Change Analysis) Extra Credit So

After getting the clusters from DBSCAN we created functions to obtain valuable information from the clusters.

The functions were able to identify the center of a cluster in terms of longitude and latitude, the spread of a cluster from the center, and the average magnitude of earthquakes in a cluster.

```
def Center(x):
    meanLat = x['latitude'].mean()
    meanLong = x['longitude'].mean()
    return meanLat, meanLong
```

Figure 6: Cluster Center Function

This function takes in a dataframe as an input and returns out the values of the centroid of the cluster in terms of *latitude* and *longitude*. By knowing the location of the centroid for each cluster we can use this to keep track of clusters as we run through the different datasets.

```
def DistCenter(x):
    distance = 0
    LatDist = x['latitude'].var()
    LongDist = x['longitude'].var()
    distance =  LatDist + LongDist
    return distance
```

Figure 7: The DistCenter Function

The DistCenter function takes in the cluster dataframe and returns a metric created from the variance of the cluster's longitude and latitude. The metric is related to the spread of the cluster. We choose to report the spread of the cluster since it is a computationally inexpensive way to find the change of the cluster between datasets. We can use the spread of the cluster in one dataset and compare it with the spread of the cluster in another dataset to see if the cluster grew larger or smaller. If the future spread is smaller than the metric of the past spread then the cluster shrinks in size. On the other hand if the spread of the cluster increases between datasets then the cluster grows in size. An alternative method would be the computationally more expensive method of finding the area of the contiguous polygon.

```
def MagPerCL(x):
    return(x['mag'].mean())
```

Figure 8: Magnitude Function

The MagPerCL function takes in a dataframe and returns the average magnitude of the earthquake events in the cluster. This can be used in conjunction with future datasets to as an indication if the hotspot behavior is changing. A smaller average for the magnitude of a cluster implies that the hotspot has a larger amount of events

surrounding it and alternatively the larger the magnitude of the cluster the smaller the events surrounding it.

```
Set  1  information
Number of hotspots : 10
Hotspot Center (Lat, Long):  (38.088981450137744, -118.12192369146005)
Spread of Hotspot (Var of Lat and Long pts of the hotspot) 0.18877081764650577
Avg Magnitude of Earthquakes in the hotspot:  2.936914600550964
```

Figure 9: Fully implemented example going through the functions listed in Figure 7, Figure 8, and Figure 9.

Basemap

To visualize the results for the hotspots we used basemap, which is a library used for the purpose of creating geographical map representations. We created two functions to set up the values required for visualizing the basemap: mapSetUp1 sets up the hotspots with clusters above density threshold d1 and mapSetUp2 sets up the hotspots with clusters above density threshold d2 but less than d1. The functions worked by first setting up a data grid for the contour plot. Then it would run a kernel density estimation. Finally, it would fit the trained model on a grid. The functions would return the datasets' longitude, the datasets' latitude, and the density. Another function would use these values to actually show the density estimations on the map: visualizeMapHotspot1 plots clusters above our high density threshold d1 while visualizeMapHotspot2 plots clusters above our medium high density threshold d2 and below our high density threshold d1.

Change Over Time

Looking at our animations and pictures overtime, hotspots whose density threshold is above d2 transitioned from having high activity in north western Texas to having high activity around lower California and finally ending up with high activity around Northern California. Hotspots whose density threshold is above d1 had a different change. First there was only one hotspot which was between the border of California and Nevada and as time passed this hotspot showed some

activity and another hotspot appeared around north western Texas. As time passed the hotspot in Texas became more and more active and finally the hotspot in California was not above the density threshold d1 anymore, making the only hotspot above density threshold d1 to be in Texas.

Animation

Figure 10: Animating d2 density threshold change over time period

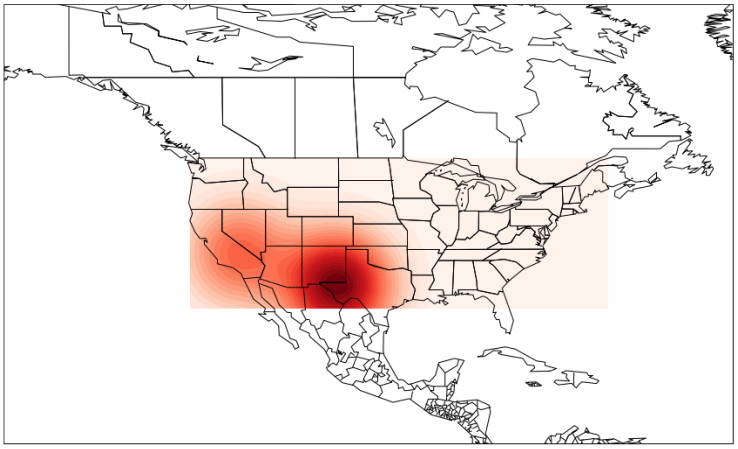
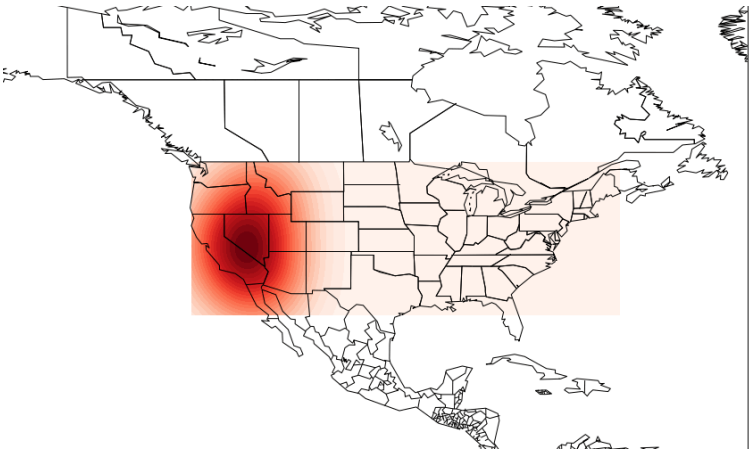


Figure 11: Animating d1 density threshold change over time period

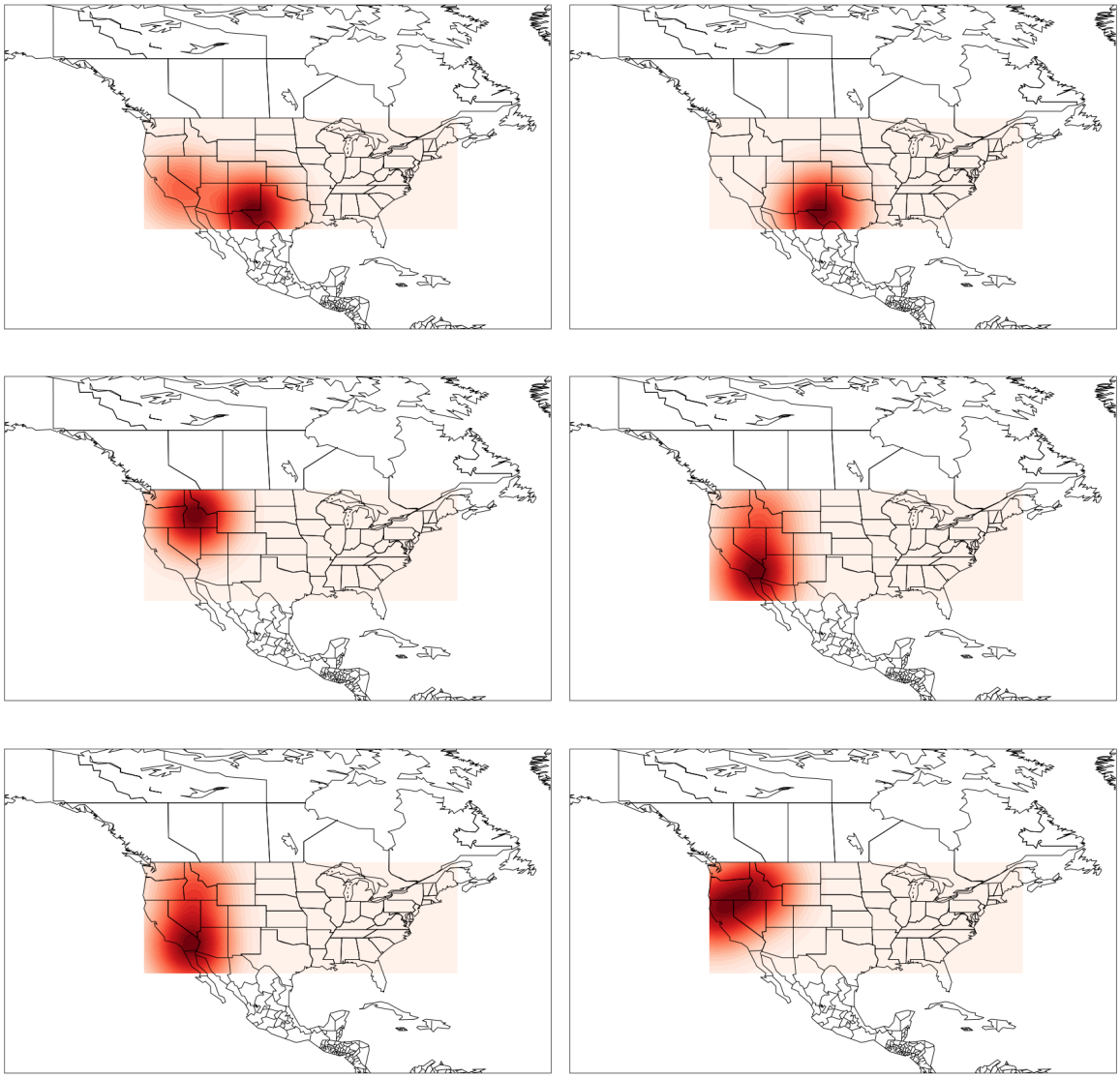


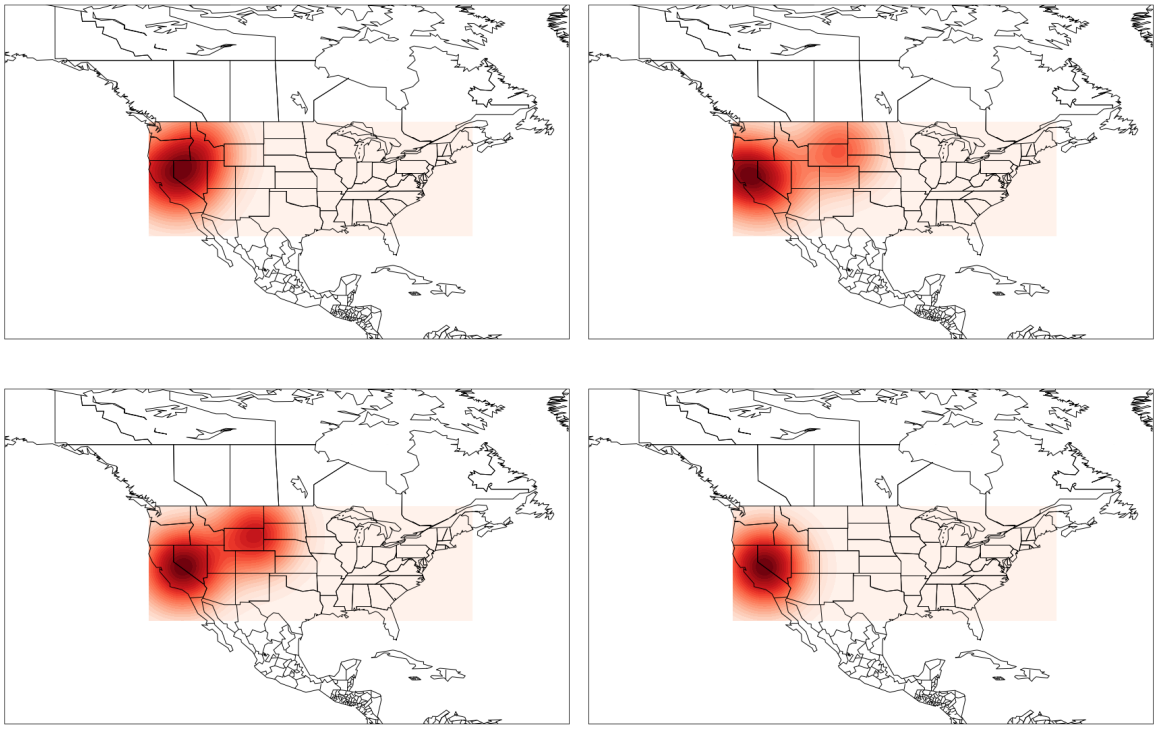
Conclusion

To conclude this report we would like to reiterate our methodology and all produced results, along with their implications. In brief, we first preprocessed the data given to us in order to format it to the specifications of the project and our ease of access. Next we decided to focus on the latitude and longitude attributes of the data, with a smaller focus on magnitude and no focus on time and depth, and loaded those into our clustering methods. These included DBSCAN which allowed us to get exactly what we wanted, without noise. This also helped us to understand our density thresholds which we assigned using information from the DBSCAN. Using those aforementioned clusters, we were able to obtain specific information about the hotspots from them, such as their center of location, spread, and average magnitude. Finally, we loaded those clusters into a basemap function to map them all and animate them to show change over time. All of this was eventually put into a pipeline function to allow an easily progressible project, except for the animation which was done outside of python while still taking advantage of the figures produced from our basemap functions. The animations showed us hotspot movements across Northern and Southern California along with Western Texas, varying depending on the density threshold used. With the understanding we have gained from this project, applying it to any other dataset would be simple. This could easily be applied to the Europe dataset spanning 2016-2018, as all we would have to do is preprocess that data into Sets like we did with the U.S.A. data and load it into our pipeline. As such, our system is great for the purpose of reusability.

Appendices

Figures 12-22: Plotting d2 “medium high” density threshold onto basemap for Sets 1-10 respectively





Figures 23-32: Plotting d1 “high” density threshold onto basemap for Sets 1-10 respectively