Trinity College Dublin
Coláiste na Tríonóide, Baile Átha Cliath
The University of Dublin

Kernel Density Estimation on Spherical Domain

Yash Shukla, B.E.

A Dissertation

Presented to the University of Dublin, Trinity College

in partial fulfilment of the requirements for the degree of

Master of Science in Computer Science (Intelligent Systems)

Supervisor: Dr. Athanasios Georgiadis

August 2022

Declaration

I, the undersigned, declare that this work has not previously been submitted as an exercise for a degree at this, or any other University, and that unless otherwise stated, is my own work.

Yash Shukla

August 19, 2022

Kernel Density Estimation on Spherical Domain

Yash Shukla, Master of Science in Computer Science

University of Dublin, Trinity College, 2022

Supervisor: Dr. Athanasios Georgiadis

In this data-driven world, loads of data are generated every second. To get insights from this data, we need to process it through statistical algorithms or some pre-defined ML models. This is feasible only for data with any patterns or parameters that can be predicted. It is tough to get any insights from any dataset that has randomness. The solution for such data is a non-parametric estimation. This research attempts to get insights from such random data using kernel density estimation (a non-parametric estimation technique) in a spherical domain. The dataset used here consisted of earthquakes from 1950 to 2022 of magnitudes 7 and 6, which the U.S. Geological Survey collected under the Earthquake Hazards program. This dataset has random longitude and latitude coordinates of the places where earthquakes occurred. The data was first cleaned and analyzed using python scripts. Then, data was converted from two to three dimensions using trigonometric functions as the research is in a spherical domain. The kernel density estimation approach uses Legendre polynomials to estimate the randomness easily. The data was processed through a kernel for different bandwidths h = 0.01, 0.25, 0.5, and kernel levels k = 1, 10, plotted on a graph to visualize the estimated values. These graphs were compared to the 2D graphs, which were generated with just coordinates placed on the map to check if the estimated values were similar to those of the original values. The major challenge during this research was executing the code for huge data sets, but it was solved by making some changes in the sphere generation. This research also evaluates the estimator using the log-loss sample method and plots for multiple h-value and truncation point combination where $N = 50, s = 0.5$ gives the perfect estimation. The evaluation was done on the basis of both datasets.

Acknowledgments

I would like to thank my supervisor Dr. Athanasios Georgiadis for his guidance and support throughout during the research. And also for his research in statistics which helped me to carry out my dissertation.

Further I would like to thank my parents Sushil Shukla and Rekha Shukla for making me capable to take a step further in my life. I would also like to thank my entire family and friends for their support.

Yash Shukla

University of Dublin, Trinity College
August 2022

Contents

List of Tables

List of Figures

Chapter 1

Introduction

This Chapter discusses the motivation behind the research, objectives of the research, contribution, and finally, ends with the challenges faced during research.

With the advent of data, every organization is trying to make their decision from the data they have been collecting for years, Kusiak (2009). This helps them increase their profits and improve the company's efficiency. However, not every chunk of the data collected has the capacity to give the knowledge efficiently. You need to process it and then give it to a machine learning model, which gives you an output that will help in the decision. A dataset has a lot of parameters, which helps to process them through machine learning algorithms. If they do not have any parameters which can be used for getting the predictions or to get some classification models, we can use some estimation techniques to get some insights from the data. For example, a Dublin bikes dataset has a variable named "available bikes," which can be predicted using time-series analysis as the dataset has the hourly bike availability data with timestamps. Another example would be weather prediction based on temperature and wind. You can use a machine learning model to get output suitable for decision-making. In each of the above cases, one or the other variables in the dataset had a pattern that would help the machine learning model give predictions. These models also have some assumptions for the data, such as a linear relationship between dependent and independent variables, there should be no auto-correlation in the dataset, etc Kharwal (2021). Such assumptions do not apply to some datasets. For such datasets, we use non-parametric estimation. For example, in the Ozone layer depletion dataset or any astrological dataset where you have coordinates and the respective depletion level or some other parameters, in such cases, these coordinates are randomly placed, which is not feasible for the parametric algorithms. In this dissertation, we are going to discuss one such non-parametric estimation technique, which is kernel density estimation on spherical domains.

1.1 Motivation

In recent years, many of the deadliest earthquakes have occurred all around the globe, where none of the countries were prepared for them. This is a natural calamity; it is not that easy to predict which tectonic plate will have an earthquake in upcoming years. The data of such incidents do not even have a parameter that can be predicted, or it does not even have any following pattern that can help to predict it via any machine learning algorithm; it just has random coordinates. This randomness of the data can be used as an estimation parameter, which is a well-known statistical problem of solving a probability density function with values ranging from X_i to X_n.This randomness and the problem of getting insights from the earthquake dataset motivated me to do research on it by using the statistical approach of non-parametric estimation, which is Kernel Density Estimation.

1.2 Objectives

This research aims to test the kernel density estimation function developed in the paper Cleanthous et al. (2022) on the real-time dataset. This research also tries to tweak the kernel's parameters and get insights and the best possible parameters to get an accurate estimation from this non-parametric estimation technique.

1.3 Challenges

There were many challenges faced during the research. The initial challenge was to figure out how to generate the sphere efficiently. Initially, we were able to plot it with the help of normal arrays, due to which the execution time increased as the kernel code has quite heavy calculations. Due to this sphere code, the execution time for 10,000 coordinates in the dataset was more than 24 hours, which was another challenge to handle. For solving this execution time, we also used High-Performance Computing provided by the IT Research department, which helped to cut down the execution time to 18 hours due to the high specification of the machines, though it was a lot to handle. To solve the sphere problem, we changed the code and used longitude and latitude ranges to plot it, which helped to decrease the execution time from 24 hours to 1 hour.

1.4 Dissertation Overview

1. **Chapter 1 - Introduction**

   This chapter gives the initial understanding of the motive behind the research as well as the motivation for the research. Further, it also provides the objectives and contribution of the research. And it ends with the challenges faced during the research.

2. **Chapter 2 - Background and Literature Review**

   This chapter discusses kernel density estimation and the statistics used to get the estimated values. Further, it discusses the authentic Legendre Polynomial, which helped the kernel to detect the randomness. Finally, it ends with the literature review, where we discuss how different researchers used KDE for estimation.

3. **Chapter 3 - Methodology**

   This chapter gives the gist of what dataset was used in this research. Further, it discusses the significant packages used in the research, which made it possible to generate results. And at the end, it describes the overall flow of implementation.

4. **Chapter 4 - Implementation**

   This chapter first sheds light on the data and how it was processed to pass it through the kernel in three dimensions, as the research is based on the sphere. Further, it discusses how the kernel code was written and each module of it. In the end, it discusses how the final results were plotted using various packages in modular form.

5. **Chapter 5 - Results and Discussion**

   This chapter discusses the final results generated for both the data chunks. Then gives the comparative analysis for the plots, which were plotted for different h and k value combinations.

6. **Chapter 6 - Evaluation**

   In this chapter, we discuss how the estimations made by the kernel were evaluated using Theoretical Upper bound and out-of-sample Log-loss. It also discusses the h-value and truncation value, finalized on the basis of higher accuracy and were further used to plot the estimation for the magnitude 7 dataset.

7. **Chapter 7 - Conclusion**

   This chapter is the end of this thesis, where we conclude the writing, and it also discusses the future scope of the research.

Chapter 2

Background and Literature Review

This chapter gives an insight into what Kernel Density Estimation is and the statistics behind it.

2.1 Background on kernel density estimation

This section describes Kernel Density Estimation (KDE) and the parameters of KDE that play a significant role in making estimation more accurate and better.

Kernel Density Estimation is a nonparametric estimation technique used to figure out the probability density function of dataEbrahim (2008). It helps to detect the randomness of the data. To get estimated values from it, there are no assumptions considered in parametric estimation or any other machine learning algorithmChen (2017). It learns the shape of the data itself and adapts the shape of the data and (1994). For example, in figure 2.1 you can see it has a histogram plotted for data, and for the same data, a kernel density estimator is plotted, which adapts the shape of the histogram in the form of a smooth curve. Here, for getting the estimate, the kernel used was sin-wave as it had a 2-dimensional data of histogram. However, for three-dimensional dimensional or n-dimensional data such as Earth, we need to use a sphere as a kernel. In the next section, we will discuss the KDE in metric spaces.

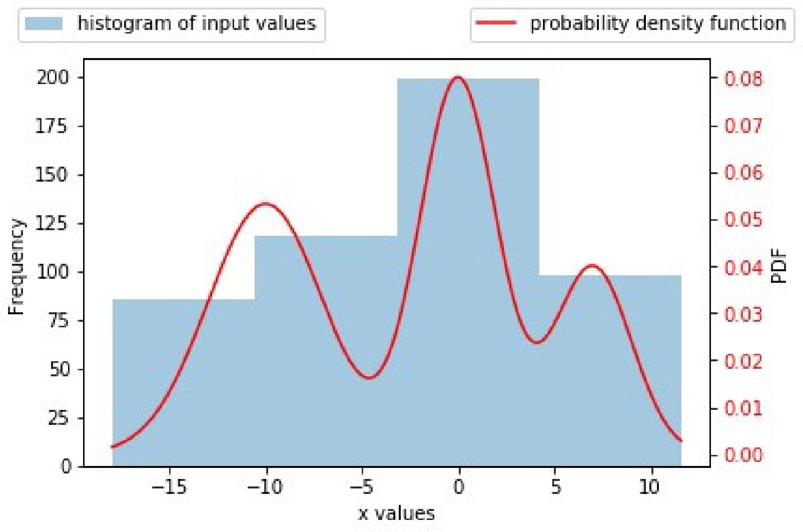


Figure 2.1: Plot of KDE for a histogramCiortan (2019)

2.1.1 Kernel density estimators

The equation 2.1.1 gives the equation used for estimating the values of the spherical domain. Where h is the smoothness parameter that decides the radius of the estimate, the effect of this h parameter is discussed further in 2.1.1. Here, P_ν is the Legendre polynomial function of degree m. However, in this case, we are taking the Legendre polynomial of ξ and η. Where ξ would be the three-dimensional coordinates on the sphere and η would be the matrix form of the sphere. The dot product of this is calculated, and then the Legendre polynomial of degree v is calculated. The formula 2.1.1 is for just a single coordinate if we remove the first summation. Further, we have the formula for multiple coordinates of KDE 2.1.1. Here k is the fast decaying function which can be observed in example 2.3.9

$$\hat{f}_{n,h}(\xi) = \frac{1}{n}\sum_{i=1}^{n}\sum_{\nu=0}^{\infty}\frac{1+2\nu}{4\pi^2}k\big(h\sqrt{\nu(\nu+1)}\big)P_\nu\big(\langle\xi, X_i\rangle\big), \quad \xi \in S^2, \tag{2.1.1}$$

2.1.2 Hölder spaces

Definition 2.1.1 *Let $s > 0$ and denote by $\ell = \lfloor s \rfloor$. The Hölder space of order s, $\dot{\mathcal{H}}^s$, is defined as the set of all functions $f : S^2 \to R$ such that*

$$\|f\|_{\dot{\mathcal{H}}^s} := \sup_{x \neq y} \frac{\left|L^{\ell/2} f(x) - L^{\ell/2} f(y)\right|}{\rho(x,y)^{s-\ell}} < \infty. \tag{2.1.2}$$

2.2 Legendre Polynomial

Legendre functions are solutions to Legendre's differential equation in mathematics:

$$\frac{d}{dx}\left[(1-x^2)\frac{d}{dx}P_n(x)\right] + n(n+1)P_n(x) = 0 sci\ (2008), Weisstein\ (2015) \tag{2.2.3}$$

Adrien-Marie Legendre was the name given to them. Physics and other technical fields frequently use this differential equation. It mainly occurs while attempting to solve Laplace's equation in spherical coordinates along with any related partial differential equations Weisstein (2015). The normal power series method can be used to solve the Legendre differential problem. A series solution at the origin will often converge only for —x— less than 1 since the equation contains regular singular points at x = 1. When n is an integer, the series for this solution comes to an end because the regular solution $Pn(x) at x = 1$ is also regular at x = 1. (i.e., it is a polynomial). The Legendre polynomials, a series of orthogonal polynomials, are these solutions for n = 0, 1, 2,... (with the normalization $Pn(1) = 1$). Each Legendre polynomial Pn(x) is an nth-degree polynomial.

2.3 Kernel density estimators on the sphere

Since the Earth is spherical in shape, the unit sphere S^2 of R^3 is the most important domain in many scientific disciplines. Even though this work focuses on earthquakes as a seismological phenomenon, researchers in fields as diverse as astronomy, ecology, and geology might find the geometry presented here to be of interest G. Cleanthous (2021).

Theorem 2.3.1 *Let $s > 0$, $f \in L^\infty \cap \dot{\mathcal{H}}^s$ and a symbol $k \in \mathcal{C}^\tau(R_+)$ for some $\tau > 2 + s$, satisfying: $k(0) = 1$,*

$$k^{(\nu)}(0) = 0, \quad \textit{for every } 1 \leq \nu \leq \tau, \tag{2.3.4}$$

$$|k^{(\nu)}(\lambda)| \leq C_\tau(1+\lambda)^{-r}, \quad \textit{for every } \lambda \geq 0,\ 0 \leq \nu \leq \tau \tag{2.3.5}$$

for some $r > \tau + 2$, and

$$k^{(2\nu+1)}(0) = 0, \quad \textit{for every } 1 \leq 2\nu + 1 \leq \tau. \tag{2.3.6}$$

We pick $h = h_n = n^{-\frac{1}{2s+2}}$. Then for every $n \in N$ the corresponding kde $\hat{f}_{n,h}$ satisfies

$$\sup_{x \in \mathcal{M}} E\big[\big(\hat{f}_{n,h}(x) - f(x)\big)^2\big] \leq cC(f)n^{-\frac{2s}{2s+2}}, \tag{2.3.7}$$

where the constant $c > 0$, depends only on τ, s, c_τ while $C(f)$ is given by

$$C(f) := \max\big(\|f\|_\infty, \|f\|^2_{\dot{\mathcal{H}}^s}\big). \tag{2.3.8}$$

We consider the symbols

$$g_\sigma(\lambda) := (1 + |\lambda|^\sigma)^{-1},\ \lambda \in R, \tag{2.3.9}$$

for $\sigma \in N$, with $\sigma > 1$. Evidently for every $\sigma > 1$, the symbol g_σ is an even function such that $g_\sigma \in \mathcal{C}^{\sigma-1}(R)$, $g_\sigma(0) = 1$, $g_\sigma^{(\nu)}(0) = 0$, for every $1 \leq \nu \leq \sigma - 1$ and presents the decay as in (2.3.6) for $r = \sigma$.

Let $s > 0$ and denote by $\lceil s \rceil$ the smallest integer strictly greater than s. The symbols (2.3.9) for $r = \sigma := 5 + \lceil s \rceil$ satisfy the assumptions of theorem2.3.1 for densities on $\dot{\mathcal{H}}^s$.

From G. Cleanthous (2021) R or Python could use the expression (2.1.1) after the infinite series is truncated until some certain integer $N \in N$. Then:

$$g_r(h\sqrt{\nu(\nu+1)}) < (h\sqrt{\nu(\nu+1)})^{-r} < h^{-r}\nu^{-r}.$$

For the Legendre polynomials it is well-known that $|P_\nu(u)| \leq 1$, for every $u \in [-1, 1]$ and of course $\frac{1+2\nu}{4\pi^2} \leq 0.51\frac{\nu}{\pi^2}$, for every $\nu \geq 25$.

Then the error (absolute value of the difference) because of the truncation of (2.1.1) until the order $N \geq 24$ can be safely bounded from above by

$$\begin{aligned}\text{error} &\leq \sum_{\nu>N} \frac{0.51\nu}{\pi^2} h^{-r}\nu^{-r} = \frac{0.51}{\pi^2} h^{-r} \sum_{\nu=N+1}^{\infty} \nu^{-r+1} \\ &\leq \frac{0.51}{\pi^2} h^{-r} \int_N^{\infty} x^{-r+1} dx = \frac{0.51 h^{-r} N^{-r+2}}{\pi^2 (r-2)}. \qquad (2.3.10)\end{aligned}$$

$h = n^{-1/(2s+2)}$, where n is our datasize.

Expression (6.1.1) asserts that an effectively large N could provide a certain error-bound.

Let us take for instance $s \in (0, 1]$ (the less restrictive range) which corresponds to the value $r = 6$. In this case, the error is at most

$$\frac{0.51 * n^{3/(s+1)}}{4\pi^2} N^{-4}. \qquad (2.3.11)$$

In a specific data analysis, with a given data size n, we bear in mind to respect (2.3.11) for the hypothetical smoothness' level s and obtain appropriate values for the error that we pre-define as "suitable." For a specific data analysis, the reader is referred to Section 4.1.

2.4 Literature Review

2.4.1 KDE using GIS

Paper by Forlin et al. (2016) discusses the analysis of medieval period earthquake datasets using kernel density estimation as a function of GIS(Geographic Information System). The author plotted KDE plots using the KDE functionality of GIS, where you can pass the radius of the estimation, and it decides the epicenter for the same. The author has also plotted some graphs based on the districts for Europe and the Mediterranean region, showing the active seismic regions in the eastern European region. The methodology used in this paper was quite promising as it could handle a vast dataset and provide results. This could be a better tool for seismologists to detect and get more insights from the data recorded by multiple organizations such as USGS, AHEAD, and EMEC.

2.4.2 Use of KDE as parameter

KDE(Kernel Density Estimator) is generally used for the estimation of the probability density function of the data. However, in the research by Yousefzadeh et al. (2021) the author uses this to generate a new parameter to make a prediction on the random earthquake coordinates. The methodology used by the author to get the predictions were to test multiple machine learning algorithms such as SVM(Support Vector Machine), DNN(Deep Neural Network), DT(Decision Tree), and SNN(Shallow Neural Network) with a new parameter which was generated using kernel density estimation with Bivariate Moran's I function. The combination of Moran's I and KDE helped to calculate the error of the predictions and helped to improve the model. The new parameter helped to increase the accuracy of the prediction, according to the author. The dataset used for this research was collected from USGS(U.S. Geological Survey) and IIES. According to the author, the algorithms that worked the best were SVM and DNN for higher order magnitudes, whereas DT worked for both levels. This research code is combined with the current research to generate the FD parameter even more accurately than the previous one.

2.4.3 KDE in Community Gardens

As we discussed in the introduction that KDE is a must for any dataset with randomness. The paper by Ding et al. (2022) proves it by implementing KDE to analyze the Community Gardens in China. This research makes the real-time application of the KDE. The author used KDE with ArcGIS to plot the estimated values of KDE. China consists of many community gardens, both formal and informal, as mentioned in the research. KDE here helped to get insights into how many of the CGs are near urban infrastructures and how many are near green infrastructures. With the help of the surveys and questionnaires, the author was also able to investigate from the people which area would be better for upcoming gardens. The author used Rosenblatt-Parzen kernel estimation for KDE, which is as follows Ding et al. (2022):

$$fn(x) = \frac{1}{nh} \sum_{i=1}^{n} k(\frac{x - x_i}{h}) \tag{2.4.12}$$

Chapter 3

Methodology

This chapter discusses the flow of the research. It starts with a dataset discussion, and further, we discuss data analysis. Then we discuss the data pre-processing and conversion to a suitable format. Furthermore, this chapter talks about the kernel density estimation and then finally about plotting the absolute estimated values into the graph.

3.1 Dataset

We have used U.S. Geological Survey's publicly accessible data. The data is an earthquake dataset from the USGS Earthquake Hazards Program, which is part of the National Institute of Standards and Technology-led National Earthquake Hazards Reduction Program abo (2017). The dataset provided by USGS has a total count of 961560 rows consisting of loads of information from the Year 1900 for every possible magnitude. It is a vast number to process for our kernel as it takes much time, which we discussed in the challenges section. So we have processed different chunks of data through our kernel, which has different magnitudes and a different number of years. In addition, there are multiple variables in the dataset, which we will discuss in the data pre-processing section 4.2. Finally, the data chunks are discussed in section 4.1.

3.1.1 Dataset Summary

This segment discusses what values the dataset consists of. There are around 22 columns in the dataset. The table 3.1 shows the variable name and its description. (1990)

Variable name	**Description**
time	Time when earthquake occurred
longitude	Decimal degrees ranging from -180 to 180 which is the number of degrees to east or west. Negative values are for west
latitude	Decimal degrees from -90 to 90, which is the number of degrees to north or south. Negative values are for the southern part.
depth	Distance underground in kilometres at which an earthquake was felt.
mag	It is the magnitude at which earthquake occurred.
magType	Method used to calculate the magnitude of earthquake
nst	The total number of seismic stations used to determine earthquake location.
gap	The largest azimuthal gap between azimuthally adjacent stations (in degrees). In general, the smaller this number, the more reliable the calculated horizontal position of the earthquake.
dmin	Horizontal distance from the epicenter to the nearest station (in degrees). 1 degree is approximately 111.2 kilometers.
rms	The root-mean-square (RMS) travel time residual, in a sec, using all weights. This parameter provides a measure of the fit of the observed arrival times to the predicted arrival times for this location.
net	The ID of a data contributor. Identifies the network as considered the preferred source of information for this event.
id	Unique identifier for each earthquake
updated	time when the earthquake data was most recently updated.
place	Textual description of named geographic region near to the earthquake.
type	A comma-separated list of product types associated with this event.
horizontalError	Uncertainty of reported location of the earthquake in kilometers.
depthError	Uncertainty of reported depth of the earthquake in kilometers.
magError	Uncertainty of reported magnitude of the event. The estimated standard error of the magnitude.
magNst	The total number of seismic stations used to calculate the magnitude for this earthquake.
status	Indicates whether the event has been reviewed by a human.
locationSource	The network that originally authored the reported location of this event.
magSource	Network that originally authored the reported magnitude for this event.

Table 3.1: Dataset variable and description

3.2 Basemap

Python's matplotlib basemap toolkit is a package for visualizing two-dimensional data on maps. It functions similarly to the MATLAB mapping toolbox, IDL mapping facilities, GrADS, and generic mapping toolscre (2008). PyNGL and CDAT are more Python packages with comparable functionality. Basemap does not perform any plot on its own, but it gives the means to convert coordinates to one of 25 map projections (using the PROJ.4 C library). Then, Matplotlib is used to plot contours, pictures, vectors, lines, and points in the converted coordinates. In addition to the shoreline, river, and political boundary datasets from Generic Mapping Tools, techniques for visualizing them are supplied. The GEOS library is employed internally to trim the coastline and political boundary features to the required map projection region. Figure 3.2 shows a 3d sphere example of plotting map contours for continents with the help of basemap. To read more about this example, you can refer (2015). Further, in section 4.4, we have discussed how we have used Basemap at the implementation level.

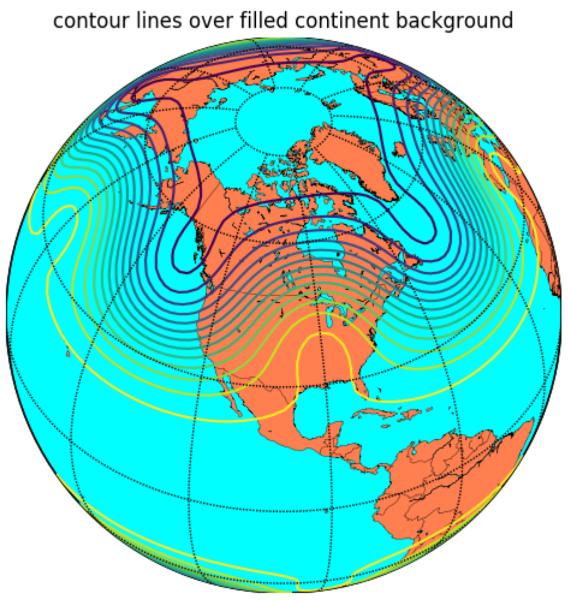


Figure 3.1: 3d plot using Basemap (2015)

3.3 Plot.ly

Plotly is a Python and Django Framework-based library on top of Plotly.js. It has the ability to visualize and examine data in order to draw conclusions from it. Although it is free to use, only a limited number of features are available. You must buy a professional version to receive all functions. It creates charts and dashboards online even if it may be an offline service inside an Ipython notebook, Jupyter notebook, or Panda. Over 40 different chart styles are available in the open-source plotting toolkit Plotly for usage in a variety of statistical, financial, geographic, scientific, and three-dimensional use cases. Plotly uses a tool called Web Plot Digitizer (WPD) to automatically extract data from static images Plotlyblog (2013). In addition to its cloud service, Plotly also provides an on-premises option that lets you host data in your own private cloud behind a firewall. For those who are worried about the privacy of their data, this is for you. Accessible APIs for these include those for Python, R, MATLAB, and Julia. For data visualization in this dissertation, Python's Plotly will be used. In this dissertation, we will use Plotly in Python to plot data. In 4.4, more information about how we applied it at the implementation level.

3.4 Sphere generation

We will require a 3D sphere for the spherical domain, which we have generated using the range of longitude and latitude of -180 to 180 and -90 to 90, respectively. First, we convert this range to radians and then use the trigonometric formula to convert it into x,y, and z-axis to plot it in a 3-dimensional graph. You can check the code snippet 3.1 for the trigonometric formula used. In the snippet 3.1 as you can see in the sphere generate function, we have first shifted the 0 to 360 to -180 to 180. Further, we pass these values of the lat long array to the mapping map to the sphere function, which helps to convert it to three-dimensional space with the help of trigonometric functions. After we have this three-dimensional space or coordinates, they are ready to be processed by the kernel for estimation.

```python
import numpy as np
def mapping_map_to_sphere(lon, lat, radius=1):
    #this function maps the points of coords (lon, lat) to points onto the
     sphere of radius radius

  lon=np.array(lon, dtype=np.float64)
  lat=np.array(lat, dtype=np.float64)
  lon=degree2radians(lon)
  lat=degree2radians(lat)
  xs=radius*cos(lon)*cos(lat)
  ys=radius*sin(lon)*cos(lat)
  zs=radius*sin(lat)
  return xs, ys, zs

def sphere_generate():
  # Shift 'lon' from [0,360] to [-180,180]
  tmp_lon = np.array([lon[n]-360 if l>=180 else lon[n]
                    for n,l in enumerate(lon)])  # => [0,180]U[-180,2.5]

  i_east, = np.where(tmp_lon>=0)  # indices of east lon
  i_west, = np.where(tmp_lon<0)   # indices of west lon
  lon = np.hstack((tmp_lon[i_west], tmp_lon[i_east]))
  clons=np.array(lon.tolist()+[180], dtype=np.float64)
  clats=np.array(lat, dtype=np.float64)
  clons, clats=np.meshgrid(clons, clats)
  XS, YS, ZS=mapping_map_to_sphere(clons, clats)
  return XS,YS,ZS
```

Listing 3.1: Sphere Generation

3.5 Design Overview

This section describes the overview of the implementation. In figure 3.2 we have a flow diagram that shows the overall structure of the implementation.

This is the very first step of this dissertation. Here we collected data from the USGS earthquake hazards website abo (2017). Since the data was too long to be processed, it was extracted into chunks. The chunks were of magnitude 6 4.1.1 and magnitude 7 4.1.2 with a different range of years which is discussed in detail in the section 4.1. After the collection of this dataset, the data was pre-processed. In dataset pre-processing, all the unwanted columns were removed from the dataset, and only three columns were used from the dataset: ' longitude,' 'latitude,' and 'mag.' Where mag was the magnitude level, longitude and latitude were the coordinates of where the earthquake occurred. The detailed description of all the columns in the dataset is given in 3.1. After pre-processing the dataset, we checked for the density and analyzed the data on a 2d map. In this step, the data was plotted on the 2d map with the help of Basemap 4.1. The 2D illustrations of both the datasets are 4.1 and 4.2 about which in detail description is in 4.1.1 and 4.1.2 respectively. Further, in the next step, the data was converted from 2d to 3d. This thesis focuses on the spherical domain, but the data extracted was 2d in terms of longitude and latitude; it was converted into x,y, and z coordinates so that they can be plotted on the 3-dimensional graph. The conversion formula of data and the code snippet for it is explained in 4.2.1. After converting the data, the only remaining part for being input to kernel processing was Sphere which we discuss in 3.4. The sphere generated by the previous module is further passed to the kernel processing model, which uses dataset points and processes all of them through the sphere and provides the estimated value with the help of packages Legendre and NumPy which in detail is discussed in section 4.3.In the end, all the pieces are brought together to get the plot generated for the final results, which are done by using packages plotly, basemap, and NumPy. By the pieces, we mean kernel's estimated value and the sphere values with basemap traces the results are plotted. More about this is discussed in the section 4.4

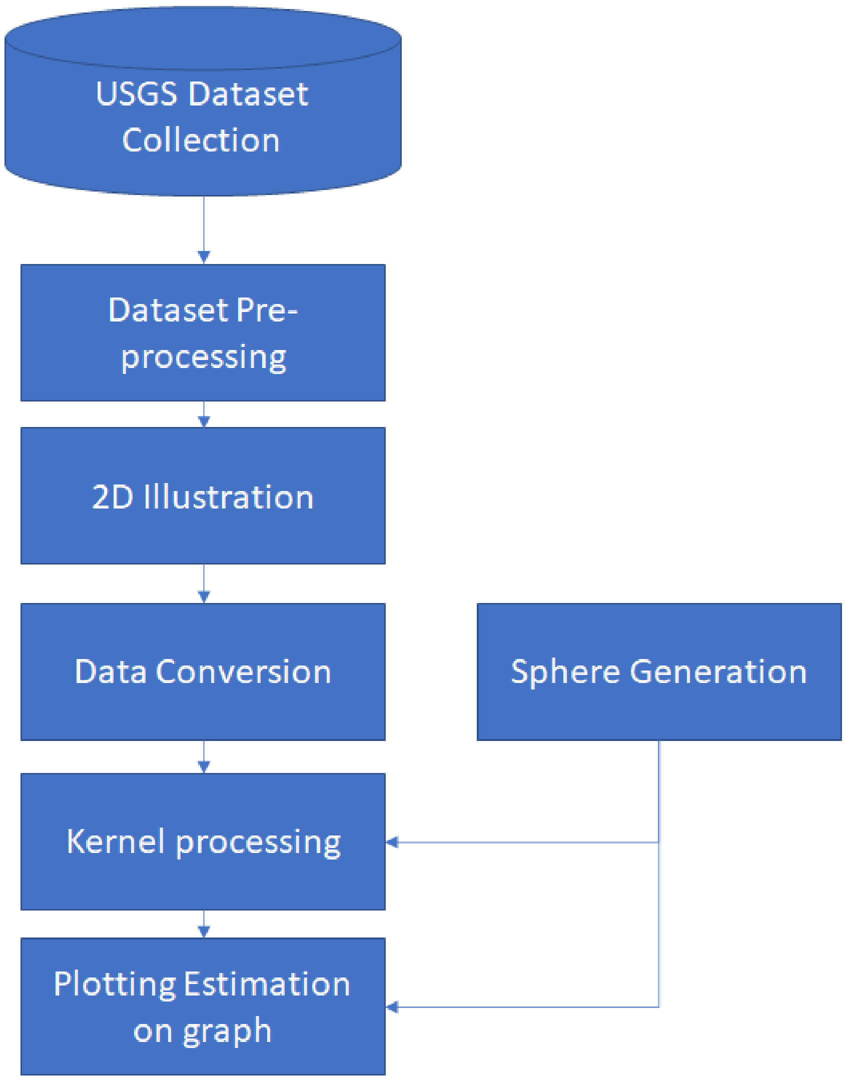


Figure 3.2: Overall flow of the implementation/dissertation

Chapter 4

Implementation

This section discusses how the dissertation implementation was carried out and what important code snippets were used in this thesis to get the expected outcome. There were multiple Python libraries used in the implementation of this thesis. The very first package was the Legendre package by scipy, about which the usage is further discussed in the section 4.3. The other package which we used was numpy. This package helped do matrix multiplications and other matrix operations in just a line of code. It also helped to do trigonometric calculations with its allowance of using different types of trigonometric functions such as sin, cos, and the degree to radians. Further pandas package was also used during data collection and extraction. Pandas helped to read the CSV file into a data frame, and it also helped to clean the dataset using its default functions. The Plotly package was also used to plot the estimated values of the kernel on a three-dimensional sphere. Plotly also helped plot the 2D map version of the dataset to carry out the data illustration process, which helped to get insights into the density of the earthquakes in a specific tectonic plate without actually passing it to the kernel. Furthermore, the Basemap package was used to plot the 3D traces of coastal lines and the country's borders with continental borders on the sphere's surface.

4.1 Data Illusttration

In this section, we are going to discuss more what is the essence of data. This section will also give you a gist of the data chunks used in this dissertation. So main chunks being used in this dissertation are given below.

4.1.1 Magnitude 6 2002-2022

This subsection discusses the dataset in which we have selected a magnitude of 6 or above from the previous 20 years' data. The dataset consists of 3138 rows and multiple columns provided by the USGS. As you can see in figure 4.1, we have plotted the dataset in the form of a 2D map with each point showing the location of earthquakes that happened between the years 2002 and 2022. Most earthquakes in North America occur on Alaska's central coast, which extends north to Anchorage and Fairbanks, and along the coasts from British Columbia to the Baja California Peninsula, where the Pacific plate collides with the North American plate Atkinson and Boore (2006). South American earthquakes extend throughout the Pacific coastline of the continentHoltkamp et al. (2011). Most earthquakes in the Asia region have happened where the Australian plate wraps around the Indonesian archipelago and in Japan.Molnar and Tapponnier (1975).

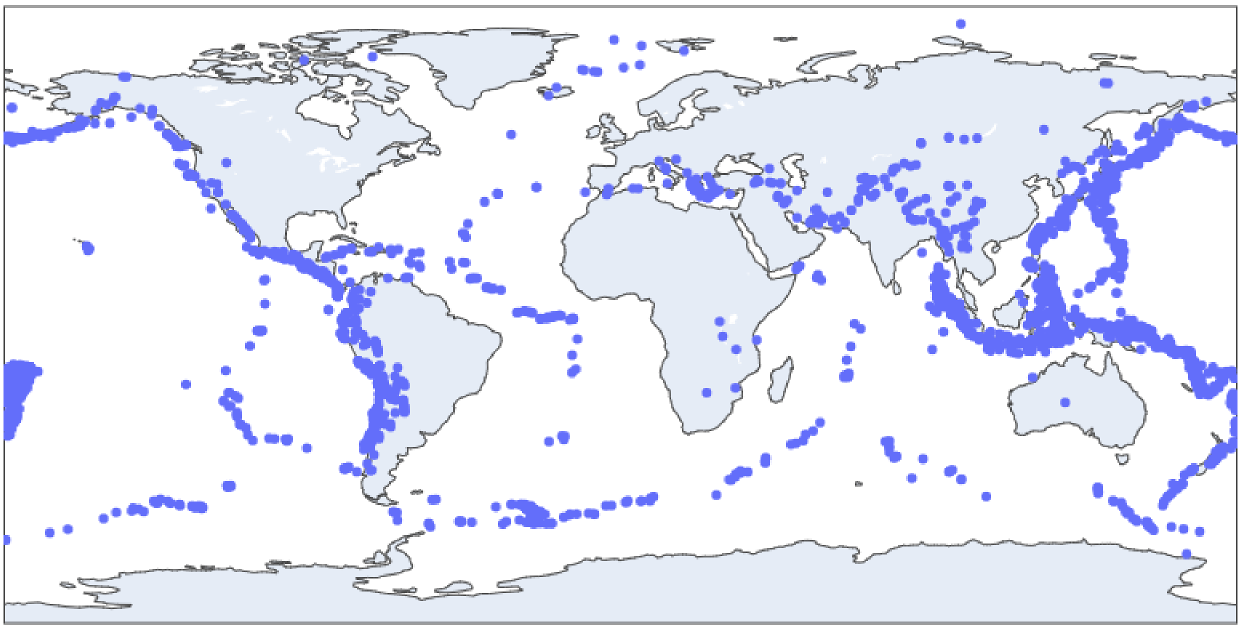


Figure 4.1: Map View of Dataset with magnitude six and from the year 2002 to 2022

4.1.2 Magnitude 7 1950-2022

This subsection discusses the dataset in which we have selected a magnitude of 7 and above for the previous 72 years' data. The dataset consists of 936 rows and multiple columns provided by the USGS. As you can see in 4.2 we have plotted the dataset in the form of a 2D map with each point showing the place of earthquakes that happened between the years 1950 and 2022. In this case, higher magnitude earthquakes are being recorded. Compared to 4.1 4.2 has a lower density near the pacific plate adjacent to the Eurasian plate due to the higher magnitude earthquakes being recorded in this dataset. If we rank in terms of density in this figure 4.2 the second rank for the density would be to the pacific rim that is near the north and south American plate.

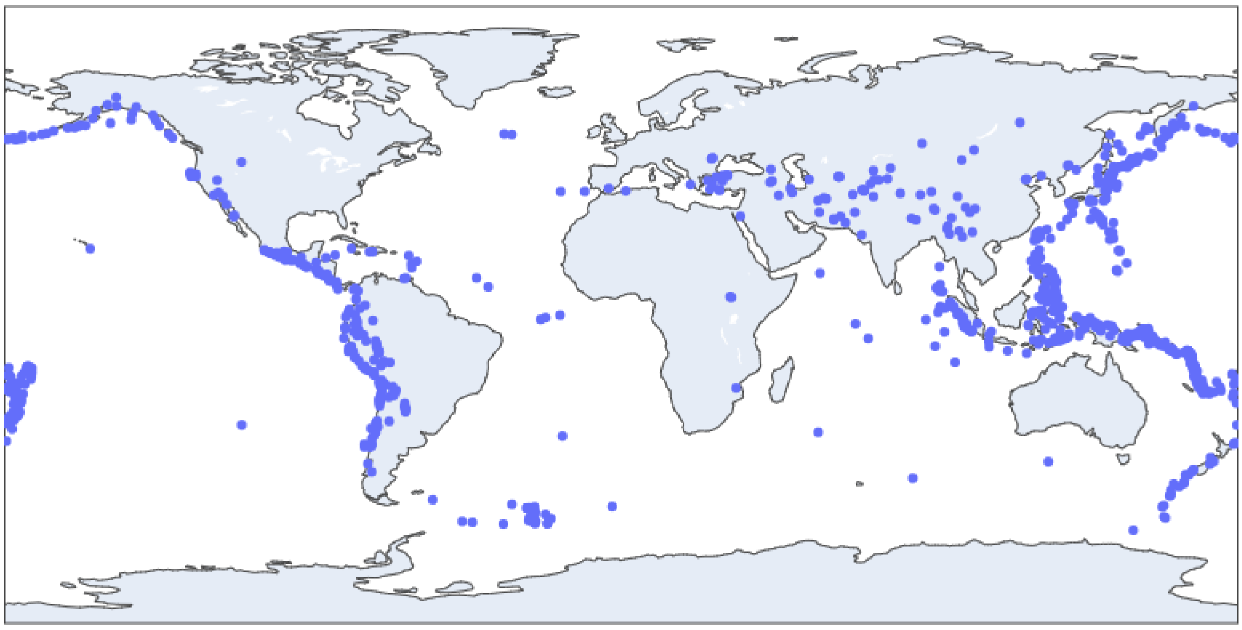


Figure 4.2: Map View of Dataset with magnitude seven and from the year 1950 to 2022

4.2 Data Pre-Processing

As we discussed in the previous section, the original data is quite heavy to use, so we decided to use it in multiple chunks. But as it has 22 columns, not every column will be needed for plotting it on the kernel. So in data preprocessing, we only extracted three columns from 22: latitude, longitude, and the time at which the event occurred. Since the longitude and latitudes were already in the form of -180 to 180 and -90 to 90, there was no issue with using them as it is for further process. Further, we will discuss how we converted these coordinates into a three-dimensional coordinate which will be feasible for plotting it on a 3d sphere.

4.2.1 Latlong to sphere

```python
import numpy as np
def degree2radians(degree):
    #convert degrees to radians
    return degree*pi/180

def convertLatLon(lat,lon):
  import math
  lon=np.array(lon, dtype=np.float64)
  lat=np.array(lat, dtype=np.float64)
  lon=degree2radians(lon)
  lat=degree2radians(lat)
  x=1.01*cos(lon)*cos(lat)
  y=1.01*sin(lon)*cos(lat)
  z=1.01*sin(lat)
  return [x,y,z]
```

Listing 4.1: Data Conversion

4.3 Kernel Processing

This section discusses the main part of this thesis which is an implementation of the kernel which gives the estimated output to the plotting function. For the implementation of this kernel processing, multiple python packages were used. The very first package is Legendre by scipy. This package has multiple functionalities, but we used only one to get the Legendre polynomial of a specific degree. Thus the kernel follows the formula 2.1.1. For applying data to this formula, the very first step is to get the dataset in a 3-Dimensional format and a matrix with three-dimensional sphere coordinates. For converting data from 2D coordinates to 3, dimensional trigonometric functions were used with the help of converting degree to radians before passing data to trigonometric functions. After the conversion of the data and generation of the sphere, we pass them to the kernel function. The code snippet 4.2 has three functions that do the complete processing of the kernel using the formula.

```python
def legendre1(val,deg):
  leg =legendre(deg)
  P_n = leg(val)
  return P_n

def dot_prod(x,y,z,eta):
  tmp =[]
  x =np.multiply(eta[0],x) + np.multiply(eta[1],y) + np.multiply(eta[2],z)
  dot_prd = []
  dot_prd.append(x)
  return dot_prd

def kn(x,k):
  cal = 1/(1+x**(6+k))
  return cal

def process_kernel(d,h,n1,k):
  x,y,z = sphere_generate()
  j=0

  for dti in d:
    # print(d[dti])

    dprod = dot_prod(x,y,z,d[dti])
    s= 1/(4*(pi**2))
    s= np.array(s)
    m=100
    for n in range(m):
      en1 = (kn(h*math.sqrt(n*(n+1)),k))*(2*n+1)
      legendre_pl = legendre1(dprod[0], n)
      legendre_pl = legendre_pl/(4*(pi**2))
```

```
32         en_final = en1 * legendre_pl
33         s = en_final+s
34       j=j+s
35     j=j/len(d)
36     return x,y,z,j
```

Listing 4.2: Kernel

4.3.1 Process kernel

The first function to be called by the code is process_kernel which is from lines number 17 to 36. This function has four input parameters d,h,n1, and k. Where d is the dataset in the form of a dictionary, h is the smoothness parameter in the kernel, which is mentioned in 2.1.1 as well, n1 is the number of rows in the dataset, and k is the kernel level. First, the sphere matrix values are stored in the variables, then we loop around the data set and get a dot prod of every coordinate with the sphere matrix, which is the basic step of the kernel. After getting dot prod, we calculate the "s" value, which is the first part of the formula, after which we calculate the sum of kernel level value with the Legendre polynomial. Where the parameter k is passed to the kernel function "kn." For kernel function, we pass two parameters first is the formula which consists of the smoothness parameter h with the square root of the product of n with (n+1), and the second parameter is k which is kernel level. The output returned by the kernel function is multiplied by (2n + 1). Further, we call Legendre polynomial function to which we pass two parameters as well. First is the dot-product of the coordinate with the sphere and the next parameter is m which is the level of the polynomial we want to produce. The output of this function is divided by $(4 * pi^2)$ then multiplied to en value which was the product of kernel output and (2n+1). Finally, it is added to the output of the previous coordinate. Finally, after adding all the coordinate data for n times, we take an overall average of the sphere matrix to achieve some realistic value from the kernel. The kernel returns the x,y, and z coordinate values of the sphere and s, which is the estimated value by the kernel. These values are passed to the plotting function to plot it on the three-dimensional graph.

4.3.2 Dot product

This subsection discusses the dot-product function. This function is in code snippet 4.2 from line number 6 to line number 11. The input parameters in this code are x,y,z, and eta. Where x,y,z can be considered as ξ, from the formula 2.1.1. In this function, first, a temporary list is initialized, after which the dot product of every dimension of ξ, i.e., x,y, and z are multiplied with eta[0], eta[1], and eta[2], respectively, where eta[0], eta[1], and eta[2] are the data coordinates that are being converted from latitude and longitude to 3-Dimensional values. Whereas x,y, and z are the matrix values or array values for the sphere. So the output of this matrix multiplication is then returned to the process kernel function to carry out the further process.

4.3.3 Legendre polynomial

Legendre Polynomial function is the most important part of this kernel processing. To make calculations possible for Legendre polynomial Legendre package from scipy.special. This package has multiple functionalities, but the only use of it in this thesis was to generate the Legendre polynomial for the data of a specific degree. This function is the code snippet 4.2 from line number 1 to 4. This function has two parameters first is val which is the data of which we need to generate the Legendre polynomial, and the second parameter is deg which accepts the value of what degree of Legendre polynomial is to be generated by the function. So in the first line, we generate the degree of Legendre polynomial with the help of the package, and further in the next line, we pass the data to that degree of polynomial to get the exact value for it. Then this calculated output is returned to the process kernel function. The value passed to this function which is the first parameter, is the output of the dot product function. The formula used by the library is 2.2.3. Where $P_n(x)$ is is a polynomial of degree n sci (2008).

4.3.4 Kernel level

Kernel level function is written in the code snippet 4.2 from line number 13 to 15. The main functionality of this function is to return the kernel level with the help of the kernel formula. This function gets two parameters as its input. The very first parameter is x which is the data that is to be passed via kernel function; the next parameter is k which is the kernel level itself. Here from the kernel process function, the square rooted value as discussed in 4.3.1 is passed as data, and k is passed as the second parameter to calculate the kernel value for it. The formula for kernel function is:

$$kn(x, k) = \frac{1}{1 + x^{(6+k)}} \tag{4.3.1}$$

4.4 Plotting Graph and Basemap

This section describes the final part of the implementation, which is plotting the estimated value on a sphere with the map traces. For plotting the whole estimated values on a graph, multiple packages were used. Firstly, the Basemap package was used to generate coastlines and borders to be plotted on a 3D sphere. The other package used to plot the data whole together was plotly.

4.4.1 Coastline and Borders generation

The code snippet4.3 shows the code for coastline generation and border generation to be plotted on the sphere. The package used for this was Basemap and plotly as mentioned above. Initially, in the code snippet, we initialize an object of Basemap. The first function called by the main is generate borders function, which is from lines number 29 to 46. This function first calls the generate coastline traces function, which is from lines number 47 to 51. This get coastline function extracts coastline pathways from the basemap package using its drawcoastline(), which gives coordinates from the package of coastline for every ocean on earth. Further in this function, the coordinates extracted from the draw coeastline method are passed through the polygon to traces function to retreive the exact line traces for the waterbodies. This function returns all the coastline boundaries to generate borders function. Then generate borders functions calls the get country traces function, which further provides the line traces of all the countries via using the polygon to trace function and draw countries function from basemap. After receiving both the latitude and longitude values for country and coastlines, the function concatenates both the lat-longs and passes them to the mapping map to sphere function, which can be referred to in the snippet 4.1. Map to sphere function provides the three-dimensional points with the help of converting these values from degree to radians and passing them through the trigonometric function. These three-dimensional coordinates are then passed into a dictionary to return to the main function to plot them on the estimated values. The figure 5.2 shows the line traces generated by this function without placing them on the sphere as well as on estimated values. As you can see in the figure that it shows line traces for each and every country and any piece of land on earth, including the Bermuda triangle as well. The country coastal line distribution helps to differentiate country borders from oceans which can be clearly seen. The figure 5.2 shows the North and South American continents with all the country border traces. This image is the 2D version of the 3D plot.

```python
# not specifying projection type for this example
m = Basemap()


# Functions converting coastline/country polygons to lon/lat traces
def polygons_to_traces(poly_paths, N_poly):
  lons=[]
  lats=[]

  for i_poly in range(N_poly):
      poly_path = poly_paths[i_poly]

      # get the Basemap coordinates of each segment
      coords_cc = np.array(
          [(vertex[0],vertex[1])
            for (vertex,code) in poly_path.iter_segments(simplify=False)]
      )

      # convert coordinates to lon/lat by 'inverting' the Basemap
    projection
      lon_cc, lat_cc = m(coords_cc[:,0],coords_cc[:,1], inverse=True)


      lats.extend(lat_cc.tolist()+[None])
      lons.extend(lon_cc.tolist()+[None])


  return lons, lats

def generate_borders():
```

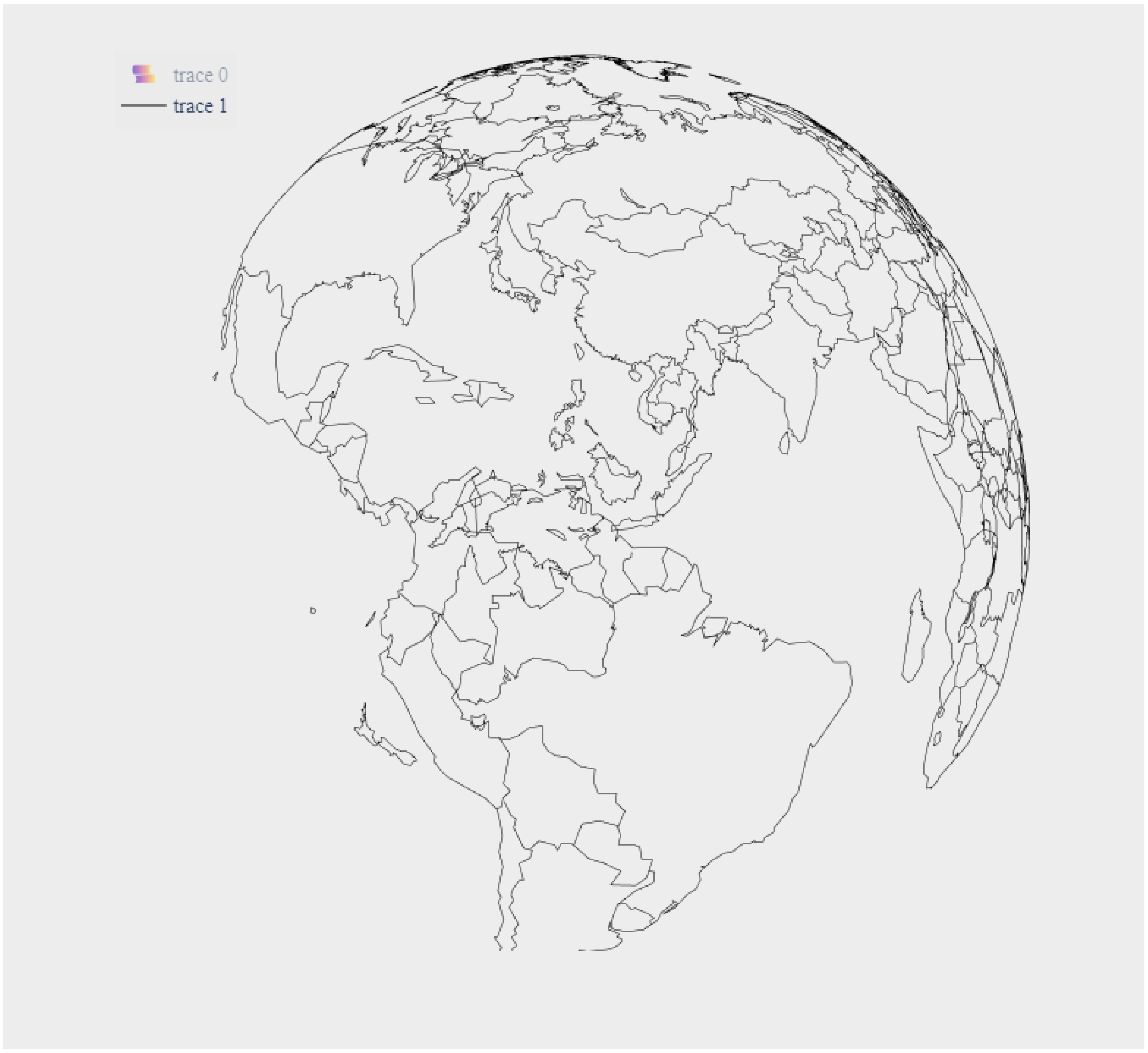


Figure 4.3: Line traces without sphere

```python
  cc_lons, cc_lats=get_coastline_traces()
  country_lons, country_lats=get_country_traces()

  #concatenate the lon/lat for coastlines and country boundaries:
  lons=cc_lons+[None]+country_lons
  lats=cc_lats+[None]+country_lats
  xs, ys, zs=mapping_map_to_sphere(lons, lats, radius=1.01)# here the radius is slightly greater than 1
                                                         #to ensure lines visibility; otherwise (with radius=1)
                                                         # some lines are hidden by contours colors
  boundaries=dict(type='scatter3d',
                  x=xs,
                  y=ys,
                  z=zs,
                  mode='lines',
                  line=dict(color='black', width=1)
                  )
  return boundaries
def get_coastline_traces():
  poly_paths = m.drawcoastlines().get_paths() # coastline polygon paths
  N_poly = 91  # use only the 91st biggest coastlines (i.e. no rivers)
  cc_lons, cc_lats= polygons_to_traces(poly_paths, N_poly)
  return cc_lons, cc_lats

# Function generating country lon/lat
def get_country_traces():
  poly_paths = m.drawcountries().get_paths() # country polygon paths
  N_poly = len(poly_paths)  # use all countries
  country_lons, country_lats= polygons_to_traces(poly_paths, N_poly)
  return country_lons, country_lats
```

Listing 4.3: Border Generation

4.4.2 Plotting sphere

This subsection discusses about the implementation of plotting spheres without any traces or estimated values. Code snippet 4.4 has the code of plotting sphere without any estimated values. In the very first line of the snippet, we create a dictionary where we pass the x,y, and z values of the sphere matrix, which was generated 3.4. Further, the background attributes are set so that axis lines are not visible, which are available in the default layout. On line number 23 plotly figure object is created to plot the sphere on the graph. The figure 4.5 shows the sphere plotted by using the snippet 4.4. By default, the plotly package gives color to the plots, which can be changed by making small tweaks in the code.

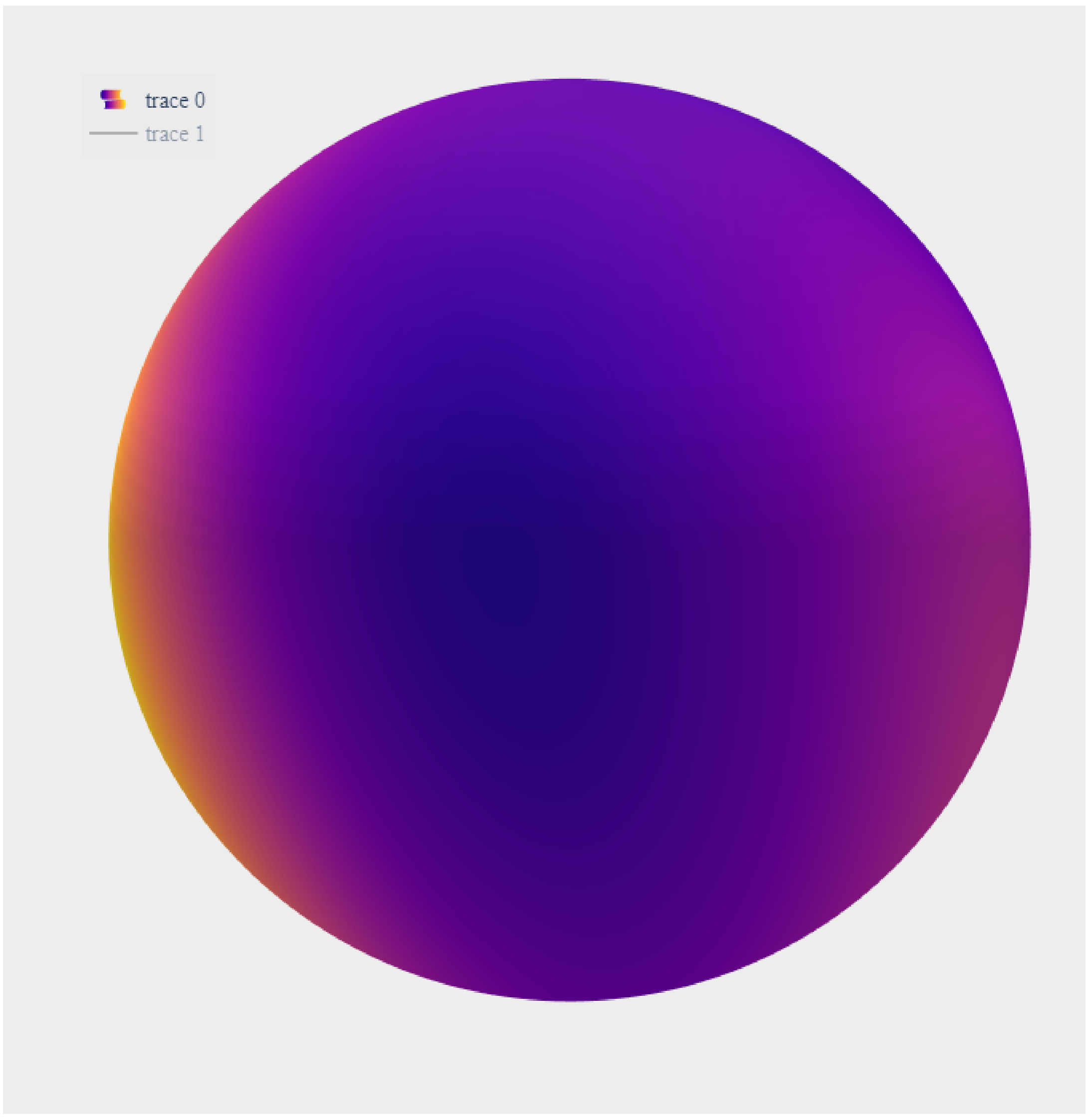


Figure 4.4: Sphere without Line traces

```
sphere = dict(type='surface',
        x=x,
        y=y,
        z=z)
noaxis=dict(showbackground=False,
showgrid=False,
showline=False,
showticklabels=False,
ticks='',
title='',
zeroline=False)
layout3d = dict(font=dict(family='Balto', size=14),width=800, height=800,
    scene=dict(xaxis=noaxis,yaxis=noaxis, zaxis=noaxis,aspectratio=dict(x
    =1,
                                      y=1,
                                      z=1),
                    camera=dict(eye=dict(x=1.15,
                                y=1.15,
                                z=1.15)
                                )
        ),
        paper_bgcolor='rgba(235,235,235, 0.9)'
      )
fig=go.Figure(data=[sphere, ],layout=layout3d)
fig.update_layout(legend=dict(
yanchor="top",
y=0.99,
xanchor="left",
x=0.01
))
fig.update_traces(showlegend=True)
fig.show()
```

Listing 4.4: Border Generation

4.4.3 Plotting Estimated values on the sphere with coastline and borders

For plotting estimated values on the sphere with all the coastlines and borders, both the previous subsection code snippets 4.3 and 4.4 are merged, and a few changes are made in the plotting sphere code which is in the code snippet 4.5. First, the generated sphere and the estimated values from the kernel are initialized in the dictionary, which is from line number 2 to 6 in the snippet where x,y, and z are three-dimensional coordinates of the sphere and s is the estimated values from the kernel. The array s is passed to surface color so that the estimated values can be represented in the form of a heat map. Further, the boundaries formed from code snippet 4.3 are given to the Plotly figure object, which is on line number 94.5. Figure 4.5 shows the 3D graph plotted by the code 4.5. As you can see, the map surface and the sphere surface have a bit gap between them; this is kept to make the country borders and coastline visible. For map traces, the radius was kept at 1.01, whereas, for the sphere, the radius was 1. The color bar on the right side shows the heat level or the density level from 0.02 to 0.14, where 0.14 is the highest density of earthquakes, and 0.02 is no earthquakes. As Japan is the highest active area for earthquakes, it has the highest density. More plots for the dataset are discussed in section 5.

```
1
2  sphere = dict(type='surface',
3                x=x,
4                y=y,
5                z=z,
6                surfacecolor=s)
7  fig=go.Figure(data=[sphere, boundaries],layout=layout3d)
8    fig.update_layout(legend=dict(
9    yanchor="top",
10   y=0.99,
11   xanchor="left",
12   x=0.01
13 ))
14   fig.update_traces(showlegend=True)
15   fig.show()
```

Listing 4.5: Upgraded code for Sphere plotting

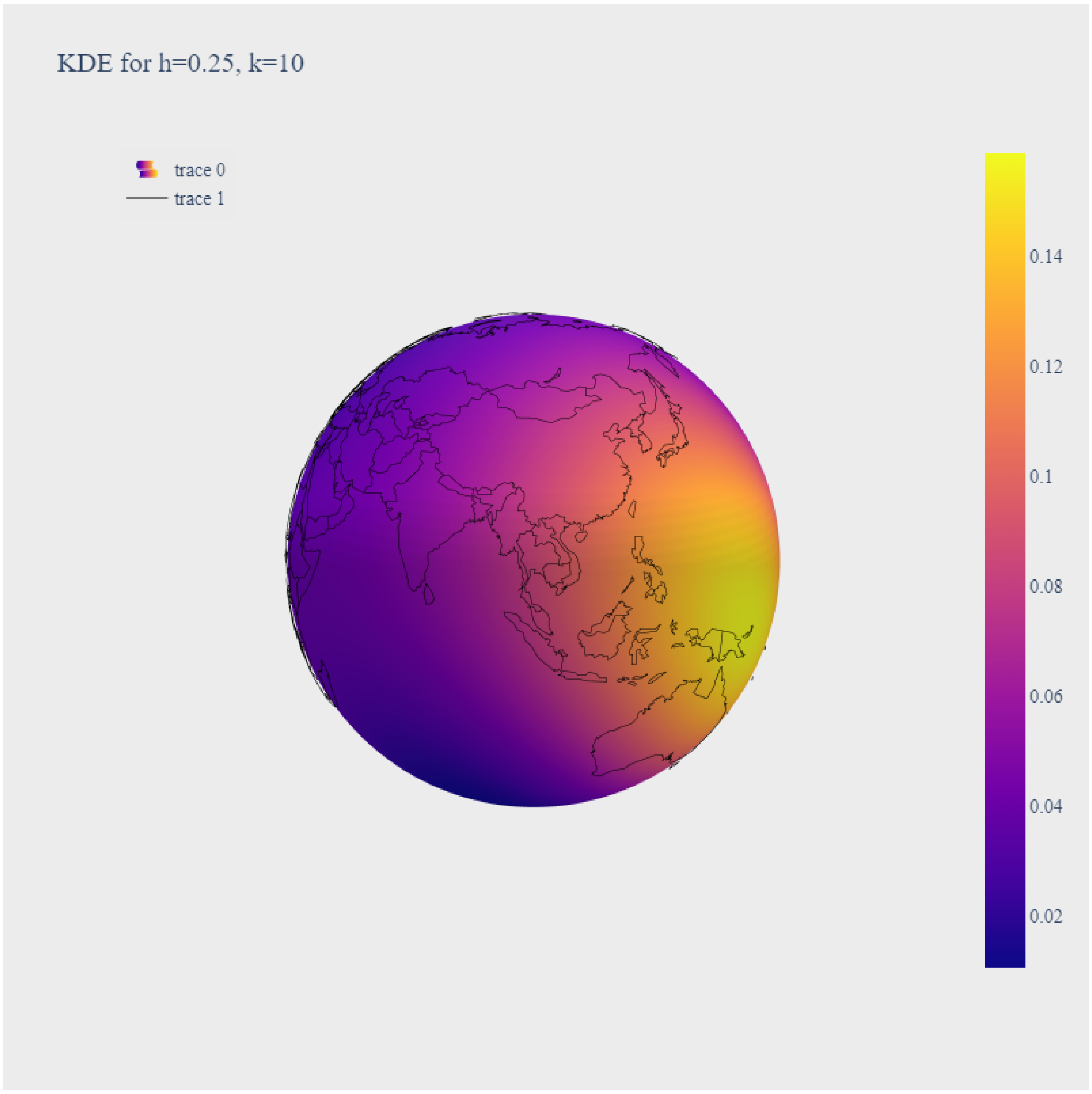


Figure 4.5: Example of Sphere plot with estimated values with coastline and borders

Chapter 5

Results and Discussion

In this section, we are going to discuss the results generated for each dataset with different parameters and k values applied to it. This will help get a clear idea of what kernel density estimation output looks like. Each output has sphere, estimated values plotted on it, Map of the world and the colorbar. In the color bar the lowest number or the color assigned to it indicates that the earthquake never occured in that region or the number of earthquakes were it less comparatively. Further after the results, we have a discussion subsection in which we will shed more light on comparing all the output.

5.1 Magnitude 6

The results here are generated by making combination of h and k selected h-value from $0.01, 0.25, 0.5$ and k value from $1, 10$. The very first output is figure5.1 which has a combination of h=0.25, k=1.

5.1.1 Estimation for h=0.25, k=1

As you can see in figure4.1 has a higher density on the Pacific plate which is adjacent to Eurasian plate near Japan, Indonesia and Australia which means they have higher count of earthquakes in that area since year 2002 of magnitude equal to or greater than 6. In the plot 5.1 the plot has a color-bar ranging form 0.02 to 0.14 where blue assigned to lowest number and yellow being assigned to the highest number. If we compare the density estimation plot generated from kernel given in figure5.1 with the 2d map plot in figure 4.1 it is showing a proper coloring for the higher density of earthquakes near pacific ocean towards Eurasian tectonic plate which is accurate in terms of density. More about comparing this plot with other combinations is discussed in section 5.2.

5.1.2 Estimation for h=0.5, k=1

This subsection discusses about the estimation for the combination h=0.5,k=1. In this as the h value is higher the radius of the estimated value is higher compared to combination $h = 0.25, k = 1$ which can be observed by checking figures 5.1 and 5.3. In figure 5.3 the color-bar has the highest value of 0.08 and minimum value to be 0.03.The color distribution is similar to previous plot lowest value has navy blue color and changing from blue to yellow for highest value. The plot of density is quite inaccurate in terms of showing the areas which are affected more about the comparison of the plots with other combination is discussed in 5.2.

5.1.3 Estimation for h=0.01, k=1

This subsection discusses about the estimation for the combination h=0.01,k=1 plot for this combination is given in the figure 5.4. This plot has the highest range of estimated values which can be observed on color-bar. Highest value being 1,00,000 and lowest being 0. The color is also assigned accordingly navy blue to 0 and yellow to the highest number which is 1,00,000. This plot shows the most accurate plot observed as it does not gives higher color or higher estimated values for the parts which have not even faced a single earthquake. As you can see the Australian plate has the lowest estimated value which means there were no earthquakes faced by the country which is actually true if we compare it to the original 2d plot 4.1. Comparison of this plot with the other two plots is discussed in the section 5.2.

5.2 Discussion

This section discusses about the comparison of the plots of different parameter combinations as well as of different magnitudes.

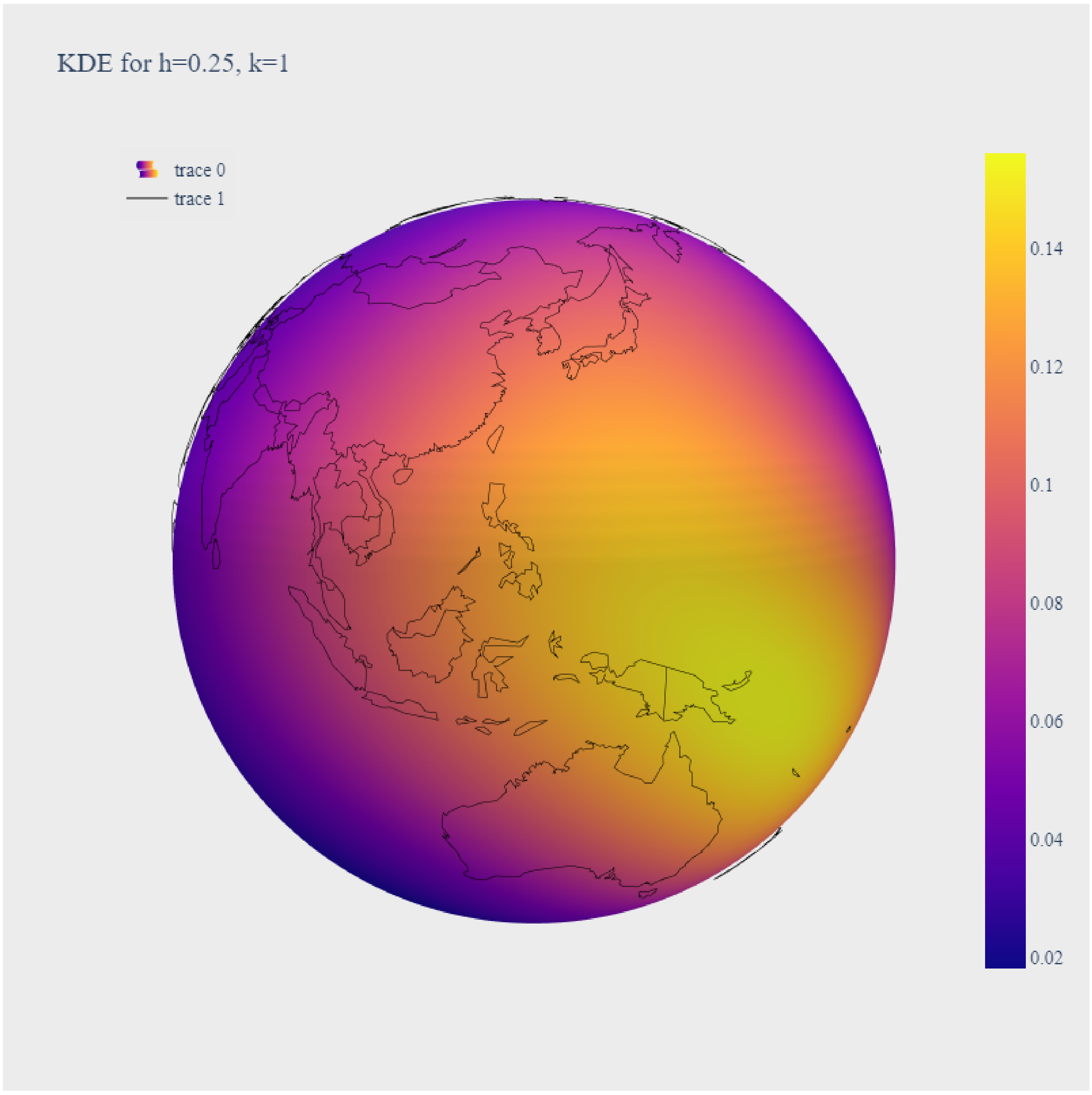


Figure 5.1: Estimated Values for magnitude 6 for k=1 and h=0.25

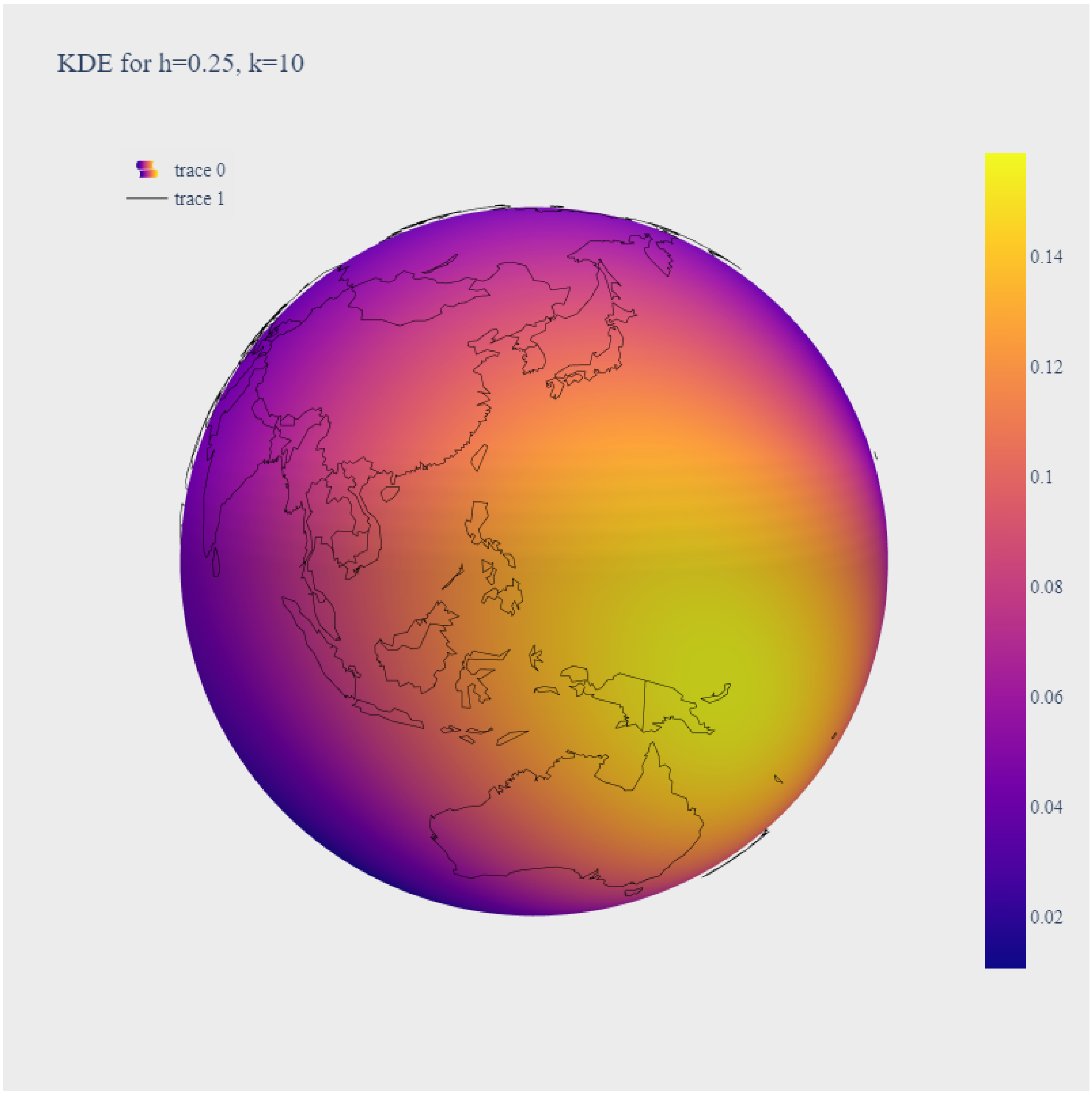


Figure 5.2: Estimated Values for magnitude 6 for k=10 and h=0.25

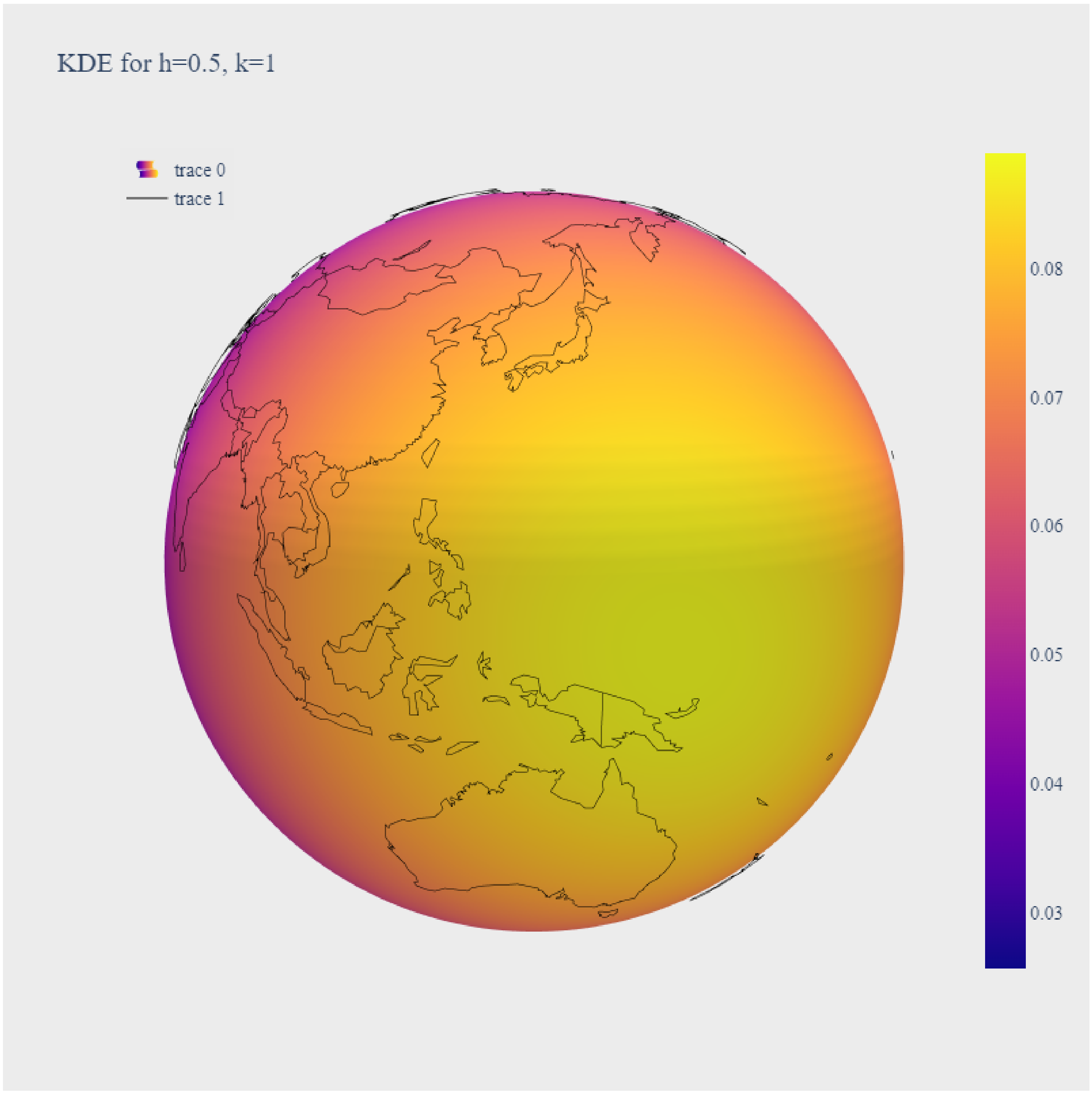


Figure 5.3: Estimated Values for magnitude 6 for k=1 and h=0.5

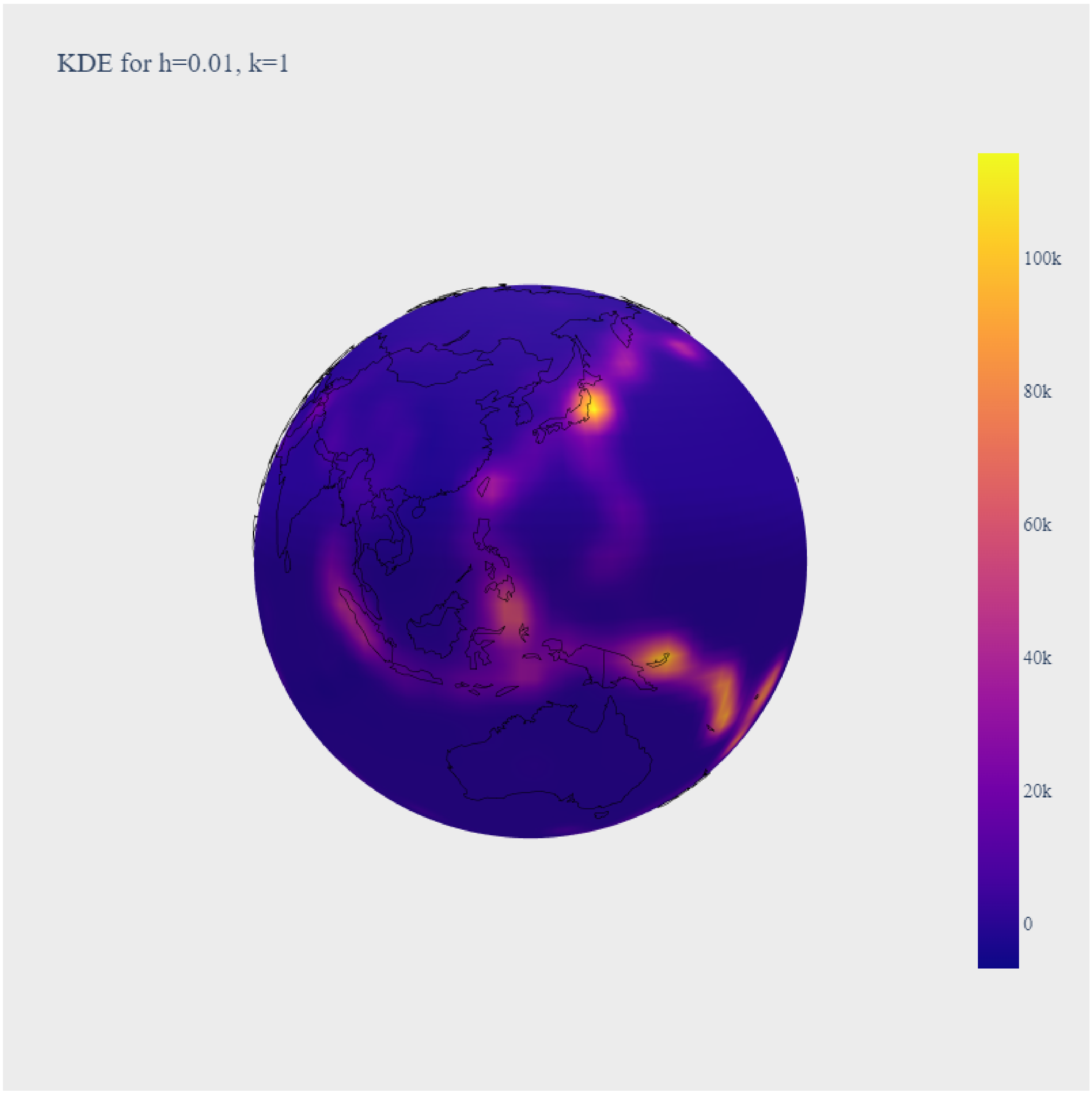


Figure 5.4: Estimated Values for magnitude 6 for k=1 and h=0.01

5.2.1 Magnitude 6

If we compare the plots 5.4,5.1 and 5.3 the plot 5.4 shows the most accurate estimation. This is because the h value is low which directly affects the radius of the estimated value. As you can see the highest density is seen in the Japan where the color is the highest shade of color-bar(yellow) which is quite accurate as Japan has seen the highest number of earthquakes. But the sharing the same rank in number of earthquakes Indonesia also has the higher number of earthquakes; Which also has the higher density estimated can be seen but as the area is quite bigger compared to japan and the earthquakes are quite well distributed all around the country so the color is quite light but well distributed. Since it is well distributed and the radius is low which is the reason to get a lower shade color from the color bar for Indonesia's estimated value by the kernel. If we compare 5.1 and 5.3 the only difference is the radius of estimation as the h value goes higher the radius increases and the estimation starts to be inaccurate because it also gives a higher feedback for the places which have never faced a earthquake. To prove this we can check the part of Australia in all the three plots 5.4,5.1, 5.3, For Australia in plot 5.4 we have the lowest shade from the color-bar which is for the areas which have less likely faced any earthquakes but in the plot 5.1 and 5.3 we can see they have the second highest color-shade and the highest color shade for the Australia respectively. The value distribution of the color-bar decreases as the h level increases which can be seen the plot

Thus, we can infer that the lower h value is feasible for seismologists to get the country-wise kernel estimation plot for the earthquakes.

Chapter 6

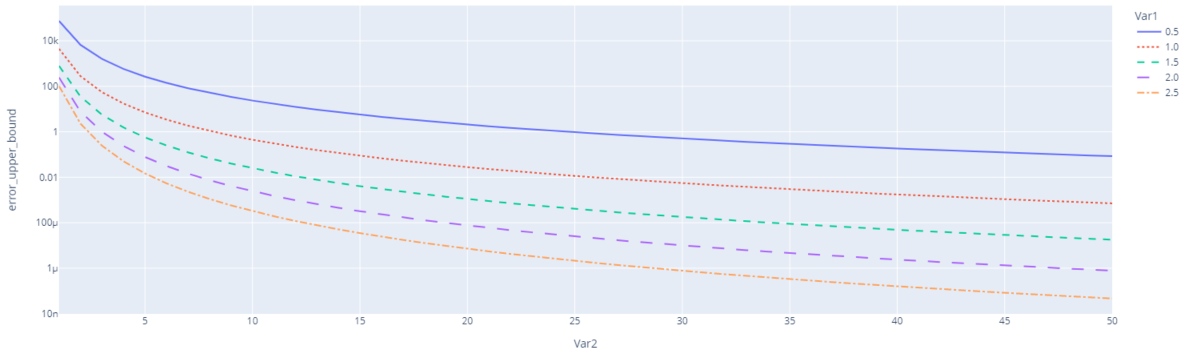
Evaluation

In this section we discuss about the accuracies of the multiple combinations of

6.1 Theoretical error and log-loss

The dataset of magnitudes 6 and 7 are widely disseminated and indexed on the sphere which can also be observed in figure4.2. To get a better estimation of earthquakes we approximate the density estimator in 2.1.1 by choosing a certain truncation point N. While analysing this data we studied how the bandwidth and truncation point of the 6.1.1 affects the estimation of earthquake location by the kernel. Instead of considering bandwidth explicitly the smoothness of the kernel is decided on the basis of the $h = n^{-1/(2s+2)}$ where s is in $0.5, 1, 1.5, 2, 2.5$. We estimate that lower s-value gives better estimation compared to larger values. We also modify the the truncation point of the density estimator N in $1, 5, 10, 15, 20, 30, 40, 50$. We encountered numerical difficulties in evaluating legendre polynomials when $N > 50$. Since truncation introduces uncertainty into the estimation, we predict that smaller values of N will result in decreased precision. We plot a graph of theorotical upper bound for both the datasets mag6 as well as for mag 7 in figure 6.1. Where we can observe the upper bound of truncation error reduces for all values of s as N grows. When $N = 50$, we observe that for all s values, the truncation errors are less than 0.01. Further we use out of sample performance to find out the bandwidth and truncation point we use 20:80 train test split to validate the kernel density estimator. Using log-loss sample method is feasible as the log-score is a scoring rule for evaluation Lindley (1992),Gneiting and Raftery (2007). We select the s value and the N value which are smoothness and truncation point on the basis of the log-loss plotted for the datasets. The lower the log-loss sample the better the higher the accuracy. In figure 6.2 as you can see we have the curve shows the higher log-loss for s-values 2,2.5,1.5,1. But for s-value 0.5 we are getting the lowest log-loss curve; especially for truncation point $N = 50$. Hence, we select $s = 0.5andN = 50$ as the values for accurate estimation. In figure 6.3 you can see the plot for mag7 database for the parameters finalized using log-loss score. As you can observe it is giving the perfect estimation which the h= 0.01 was estimating. The estimation for North American and South American plate are very accurate.

$$\begin{aligned}\text{error} &\leq \sum_{\nu>N} \frac{0.51\nu}{\pi^2} h^{-r}\nu^{-r} = \frac{0.51}{\pi^2} h^{-r} \sum_{\nu=N+1}^{\infty} \nu^{-r+1} \\ &\leq \frac{0.51}{\pi^2} h^{-r} \int_N^{\infty} x^{-r+1} dx = \frac{0.51 h^{-r} N^{-r+2}}{\pi^2 (r-2)}. \end{aligned} \tag{6.1.1}$$



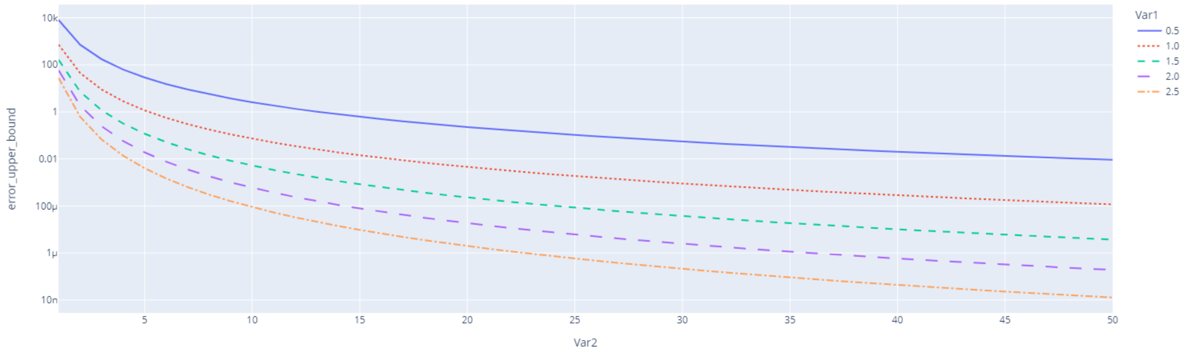
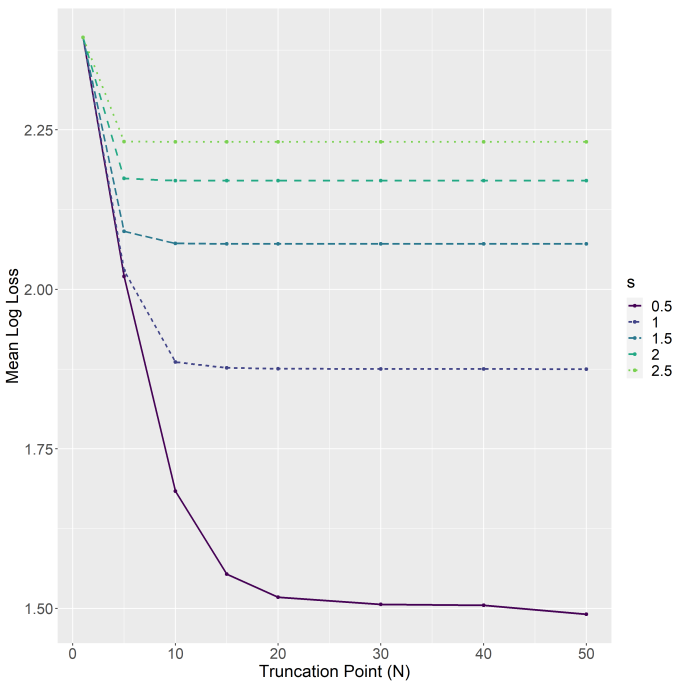


Figure 6.1: The mean out-of-sample log loss (negative log-score) for various combinations of s and N for both datasets.



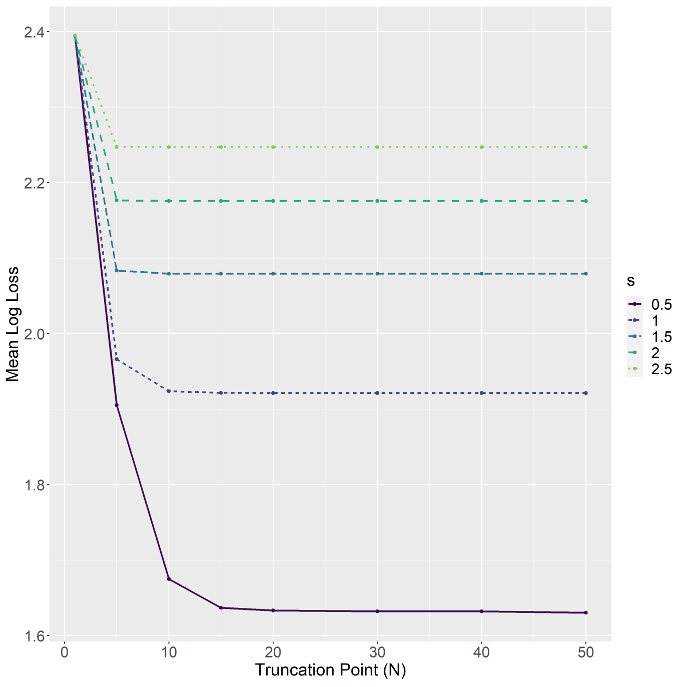


Figure 6.2: The theoretical upper bound on the truncation error from (6.1.1) for combinations of s and N. The dashed line indicates an error of 0.01 for both the datasets.

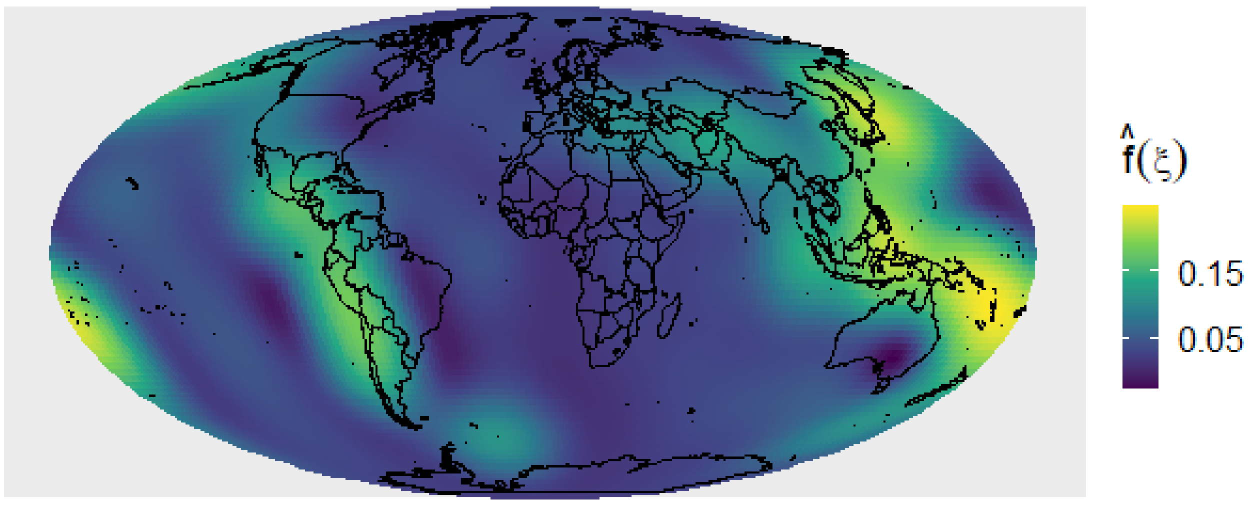


Figure 6.3: Kernel density estimate for the 1950-2022 earthquake data for mag = 7, s = 0.5 and N = 50. The colors of the density are presented on the log scale.G. Cleanthous (2021)

Chapter 7

Conclusion

7.1 Conclusion

The main purpose of the dissertation was to take a step further in the field of non-parametric estimation in the spherical domain. The data was first converted from 2 to 3-dimensional space to pass through the kernel. After conversion, it was processed through the kernel to generate estimated values for multiple combinations of h-value (smoothness parameter or radius of estimated values) and k-value (kernel level). After getting results for multiple combinations of h and k, We made the comparison of the plots generated for estimated values, which helped us to realize that the lower the h-value, the more accurate the result would be. The reason behind the lower h-value or bandwidth is that if it is higher, it gives higher estimated values for the areas which did not even face the earthquake. For example, in plot 5.3 you can notice there is a higher estimated value for Australia, but it has never faced any earthquakes in history since 1950, and the 2d plot also shows the same. So the lower h-values provide an accurate estimation of the data. The smoothness value and the truncation value for both the dataset were selected after applying log-loss out of sample method which was $s = 0.5$ and $N = 50$.

7.2 Future Scope

In current scenario for datasets with randomness there is only use of non-parametric estimation to get the value for probability density function. Further we can try to predict the earthquake using the combination of machine learning algorithm with this estimation which will help to get more accurate predictions.

Bibliography

(2008). Create Simple Maps Using worldmap- MATLAB Simulink- MathWorks United Kingdom. [Online; accessed 2022-08-18].

(2008). scipy.special.legendre — SciPy v1.9.0 Manual. [Online; accessed 2022-08-18].

(2017). What We Do - Earthquake Hazards Program | U.S. Geological Survey. [Online; accessed 2022-08-18].

, M. (2015). Plotting data on a map (Example Gallery) — Basemap Matplotlib Toolkit 1.2.1 documentation. [Online; accessed 2022-08-18].

, M. W. and , M. J. (1994). Kernel Smoothing | M.P. Wand, M.C. Jones | Taylor Francis eBooks, Re. [Online; accessed 2022-08-16].

, U. (1990). Anss Comprehensive Earthquake Catalog (ComCat) Documentation. [Online; accessed 2022-08-18].

Atkinson, G. M. and Boore, D. M. (2006). Earthquake ground-motion prediction equations for eastern north america. *Bulletin of the seismological society of America*, 96(6):2181–2205.

Chen, Y.-C. (2017). A tutorial on kernel density estimation and recent advances. *Biostatistics & Epidemiology*, 1(1):161–187.

Ciortan, M. (2019). Modality tests and kernel density estimations | by Madalina Ciortan | Towards Data Science. [Online; accessed 2022-08-16].

Cleanthous, G., Georgiadis, A. G., and Porcu, E. (2022). Oracle inequalities and upper bounds for kernel density estimators on manifolds and more general metric spaces. *Journal of Nonparametric Statistics*, pages 1–24.

Ding, X., Zhao, Z., Zheng, J., Yue, X., Jin, H., and Zhang, Y. (2022). Community gardens in china: Spatial distribution, patterns, perceived benefits and barriers. *Sustainable Cities and Society*, 84:103991.

Ebrahim, G. J. (2008). Statistical analysis and modelling of spatial point patternsIllian J., Penttinen A., Stoyan H., Stoyan D. (eds). *Journal of Tropical Pediatrics*, 55(1):69–69.

Forlin, P., Gerrard, C., and Petley, D. (2016). Exploring representativeness and reliability for late medieval earthquakes in Europe - Natural Hazards. [Online; accessed 2022-08-16].

G. Cleanthous, A. G. Georgiadis, P. W. (2021). Density estimation on metric spaces associated with operators and applications in seismology. *Journal of Nonparametric Statistics.*

Gneiting, T. and Raftery, A. E. (2007). Strictly proper scoring rules, prediction, and estimation. *Journal of the American Statistical Association*, 102(477):359–378.

Holtkamp, S. G., Pritchard, M., and Lohman, R. (2011). Earthquake swarms in south america. *Geophysical Journal International*, 187(1):128–146.

Kharwal, A. (2021). Assumptions of machine learning algorithms.

Kusiak, A. (2009). Innovation: A data-driven approach. *International Journal of Production Economics*, 122(1):440–448. Transport Logistics and Physical Distribution Interlocking of Information Systems for International Supply and Demand Chains Management ICPR19.

Lindley, D. V. (1992). Introduction to good (1952) rational decisions.

Molnar, P. and Tapponnier, P. (1975). Cenozoic tectonics of asia: Effects of a continental collision: Features of recent continental tectonics in asia can be interpreted as results of the india-eurasia collision. *Science*, 189(4201):419–426.

Plotlyblog (2013). Automatically grab data from an image with webplotdigitizer.

Weisstein, E. W. (2015). Legendre polynomial. *Mathematical Methods*, 43.

Yousefzadeh, M., Hosseini, S. A., and Farnaghi, M. (2021). Spatiotemporally explicit earthquake prediction using deep neural network. *Soil Dynamics and Earthquake Engineering*, 144:106663.