Universitatea din București
Facultatea de Fizică

Gabriel Tiberiu Pană

Properties of seismic scale-free networks

applications for Romania, United States of America, Italy and Japan

Master Thesis

Scientific Advisers
Conf. Univ. Dr. Alexandru Nicolin
Prof. Univ. Dr. Virgil Băran

Bucureşti, 2021

Contents

Introduction

In this work we wish to analyze various seismic regions around the globe from the perspective of complex networks. We expect that this would help give some insight into the mechanism of large events creation, avalanches of aftershocks and spatio-temporal span of these events and it would help us prove that earthquakes behave as a complex, self-organized critical system.

This report is organized in five chapters:

Chapter 1 is dedicated to familiarizing with terms such as complex systems, critical systems, self-organization and self-organization to criticality. A short history and introduction in network theory and graph theory is also presented.

Chapter 2 deals with the most known system that shows self-organization to criticality behaviour, the classical sandpile model. The simplified and the original model help us understand this peculiar characteristic that appears in some complex systems in nature. Also, a staple model of seismicity is presented in this chapter: the Olami-Feder-Christensen slider-blocks model.

Chapter 3 is dedicated to presenting the databases from which we extract information about earthquakes and how we manipulate these databeses in our analysis. Also, we present how we split a seismic region into small cubes, which will become the nodes of our seismic network.

Chapter 4 is the main focus of our work. Here we explain how we construct our seismic network and the centrality measures we use to analyze it. We present results regarding distribution of connectivity, weighted and unweighted. We define the building blocks of networks - graph motifs and we apply motif discovery tools to extract 3 and 4 nodes motifs from our seismic networks. Distributions of surfaces and volumes of these motifs, weighted by mean and total magnitude are presented. Finally, suggestive visualization of the real networks created, for a small part of the network are introduced.

Chapter 5, the final one of our report, deals with spatio-temporal autocorrelations in our seismic networks. The goal is to give an undertanding into the seemingly chaotic behaviour of earthquakes.

In the end, we conclude the report with a summary of results and insights obtained that point towards the self-organized to criticality nature of seismic zones.

Chapter 1

Complex Systems

1.1 What makes a System Complex

Complexity, together with **diversity**, are two properties that coexist when dealing with a complex system and a first step to understanding such a type of system would be to understand and differentiate between these two concepts [1].

Diversity applies to a collection of entities or a population; it requires a multitude of objects that are to be analyzed. For example, cities, ecosystems, molecular matter, plasma are diverse. When speaking of diversity, we can mean a few different characteristics of a population. For example you can mean a *variation* of some attribute, such as difference of height in a group of people. We can also talk about *diversity* of types, such as different ethnicities in a social group. Finally we can talk about differences in *configuration* such as different arrangments in housing units of a group of people.

Complexity, on the other hand, proves to be a much more difficult concept to define. Complexity can be thought of as particular structures and patterns that cannot be easily described or predicted. A system becomes complex when *diverse* rule-following entities behave in an *interdependent* way. These entities interact over a *contact structure* or *network*. Furthermore, these entities can also *adapt*. For example, in a social system, individuals can learn, or in an ecosystem natural selection can take place. Another characteristic of these systems is robustness, meaning that they exhibit a certain behaviour at any spatio-temporal scale you can consider and also they can produce *large* events, which is a concept that we will tackle at large in the following. Finally, they can attain equillibrium states, fixed points or patterns and they can produce long random sequences (chaotic behaviour).

As an interdisciplinary domain, complex systems draws contributions from many different fields, such as the study of **self-organization** and **critical phenomena** from physics, that of spontaneous order from the social sciences, chaos from mathematics, adaptation from biology, and many others. "Complex systems" is therefore often used as a broad term encompassing a research approach to problems in many diverse disciplines, including statistical physics, information theory, nonlinear dynamics, anthropology, computer science, meteorology, sociology, economics, psychology, and biology.

In our studies, two concepts that we need to be familiarized with are *criticality* and *self-organization* and finally, we can combine these two under the umbrella of **Self-Organization Criticality**, a concept which we use to theorise and try to explain the driving mechanism of seismic activity, the ultimate goal of this review.

1.2 What is a Critical System

1.2.1 Critical phenomena

The term **critical phenomena** refers to peculiar behaviour of a system when it is near or at the point of a continuous-phase transition, also called a *critical* point [2]. A continuous-phase transition can be defined as a point at which the system changes from one state to another without a jump or discontinuity in it's properties such as internal energy, density or magnetization. In contrast to the critical point (or interchangeable, the continous-phase transition) we can talk about the familiar case of the first-order phase transition where the properties jump discontinously as a parameter of the system (pressure, or temperature for example) passes through the transition point.

The peculiarities of the critical point arise because there are certain degrees of freedom of the system that show anomalously large fluctuations on a long spatio-temporal scale, compared to a normal system far from criticality. These large fluctuations cause a breakdown of the usual macroscopic laws that characterize the system in dramatic or subtle ways and it is very challenging to learn what are the new special laws that describe the system at it's critical points.

The physicist's attraction to the study of these peculiarities arise both in *theorists*, whom struggle with the computationally intensive interconnected laws that describe each interacting part of the system and in *experimentalists* whose challenge lies in making measurements close to the critical point, which requires extremely precise control of the paramater of the system (temperature, for example in condensed matter) which is to be modified in order to attain said point. But as it is with the minds of physicists, where there is complexity, there is also the fascination and drive to understand it, so this is the moving factor in the interest of studying these behaviours. The understanding gained has proven useful in a plethora of other systems, including quantum field theory in elementary-particle systems, analyses of phenomena in long polymer chains, description of percolation in macroscopically inhomogenous systems.

1.2.2 Examples of phase transitions and critical points

The following examples help us differentiate between first-order transition and a continuous transition (or critical point).

First-order transition: the boiling transition from liquid to vapor is a first-order transition: when the water at 1 atmosphere pressure boils at a temperature of 100° Celsius there is a sudden decrease in density by a factor of 1700.

Continuous transition: the Curie point of a ferromagnetic substance is a critical point. At temperatures below T_C, a sample of iron has a net magnetization $\boldsymbol{M}$ that points along one of several directions , in the absence of an external magnetic field. The magnetization strength decreases with increasing temperature until T_C is

reached. Above this temperature, the magnetization is zero and the material is in a paramagnetic state. So we can say that at T_C there is a continuous tranisition, or a critical point.

In order to facilitate the comparison between phase transitions it is convenient to introduce the concept of *order parameter*. For the ferromagnet, this would be the magnetization $\boldsymbol{M}$ which measures the amount of broken symmetry in the system. When there is no broken symmetry, the order parameter is a quantity sensitive to the difference between the two phases, below or above the transition temperature. For the boiling water example, the order parameter would be the actual density of the fluid.

1.3 Self-Organized Criticality

1.3.1 Self-Organization

Self-organization is the process by which individual components of a system organize their communal behaviour to create global order by interactions amongst themselves rather than through external influence or instruction [3]. Complex dynamic systems which have many and diverse elements interacting with each other, may display features of self-organization. This phenomena can be triggered by seemingly random fluctuations, amplified by positive feedback.

Fundamental characteristics of this organization are:

1. **decentralization** : the organization is distributed over all the components of the system

2. **robustness** : the system is able to survive or self-repair perturbations

"In chaos theory, self-organizations represent "islands" of predictability in a sea of chaotic unpredictability".

1.3.2 Self-Organized Criticality

Self-organized criticality (SOC) has been a controversial and complicated topic to properly define in literature. Authors considered different definitions for SOC.

In his 1996 paper "Simplest Possible Self-Organized Critical System" [4], Flyjberg explains that a SOC system is a driven dissipative system consisting of:

1. a *medium* which has:

2. *disturbances* propagating through it, causing

3. a *modification* of the medium, such that eventually

4. the medium is in a *critical state* and

5. the medium is *modified no more*

Per Bak, Chao Tang and Kurt Wiesenfeld [5] formulated the followinng key points when they defined Self-Organized Criticality (SOC):

- Spatial and temporal scaling must usually be unavoidably connected.
- In contrast to phase transitions (or chaos) seen as a certain fixed point in the order parameter space, there must be a robust, widespread new kind of spatio-temporal critical behaviour which arises from self-organization.
- The conditions for SOC to be seen in a system are slow driven interaction and existence of a threshold. Also, dissipation has a crucial role in maintaining this kind of state.
- Spacetime fractals are snapshots of the SOC state.

In SOC, criticality has a particular consequence regarding continuous transitions. At the critical point, correlations become long ranged (meaning they follow a power law) and, equivalently, fluctuations occur on all length scale, meaning that the fluctuations also display a power law dependence, with non-zero exponent.

In a 2015 comprehensive review of 25 years of Self-Organized Criticality [6], the authors argued further points that help define this concept:

The necessary conditions to observe a SOC state:

1. Non trivial scaling.
2. Spatio-temporal power law correlations.
3. Apparent self tuning to the critical point.

The sufficient conditions to generate a SOC state:

1. Non-linear interaction, usually as thresholds.
2. Avalanching (intermittently, expected when the system is slow driven and has thresholds).
3. Separation of time scale (to sustain avalanches).

Importance in seismicity modeling In the second chapter of this report, the main models of seismicity are presented, which posses a great deal of insight into the possible mechanism of earthquake generation and especially the self-organized criticality state that a seismic zone may reach before triggering an avalanche of earthquakes, comprising of a big event, followed by its aftershocks.

1.4 Network Theory

As mentioned earlier, a great tool of studying complex systems is through Network Theory [7]. Initially born out of graph theory from mathematics, it has become a great instrument in analyzing complex interacting systems by being able to keep track of their parts and the relationships between them in a simple and intuitive manner. Complex networks are networks whose structure is irregular, complex and dynamically evolving in time.

1.4.1 Historical View on Networks

The six degrees of separatiom The concept of networking has been born much earlier than one would think [8]. Although it has been mathematically defined in recent history, the most common network one turns to when speaking of this subject, the social network we all are a part of, has been brilliantly identified in 1967 by Stanley Milgram, a Harvard professor, who made a groundbreaking study on our interconnectivity.

Milgram's goal was to find the average "distance" between two random people in the United States, i.e. how many acquaintances would it take to link two randomly selected persons. He first chose a target person (for example, a broker in boston), then sent letters to random people across the United States with the target's picture and information. When a person receives the letter, they first have to record their name on a list in the letter to keep track of the path. Then send the letter to the target only if they knew them on a first name basis, or to a person that they think would be "closer" to the targets. The process continues until the target is reached. The result: 42 of 160 letters reached the targets, and the average distance was 5.5, thus the famous "six degrees of separation".

The discovery was intriguing because it suggests that despite the vast size of our social network, we live in a world in which no one is more than a few handshakes away from anyone else. Our society is a small world because it is a very dense web.

Figure 1.1: Stanley Milgram's Small Worlds Experiment: A letter randomly sent to a citizen in Nebraska starts on a 6 person's journey to it's target in Boston. Each person mailed the letter to an acquaintance that they thought would be closer to the target. The second to last person, mails the letter to the target because they knew him personally.

Euler's bridges Even earlier than Milgram's experiment, the problem that is thought to be the birth of graph theory is the Seven Bridges of Königsberg [9].

The problem is formulated as follows: is there a possible walk through the city that would cross each of the seven bridges once and only once (the configuration of land-bridges is depicted below)? "No" deduces Leonhard Euler in 1736 and gives it's solution, along with other representations which many consider today the first conjuncture of graph theory in mathematics.

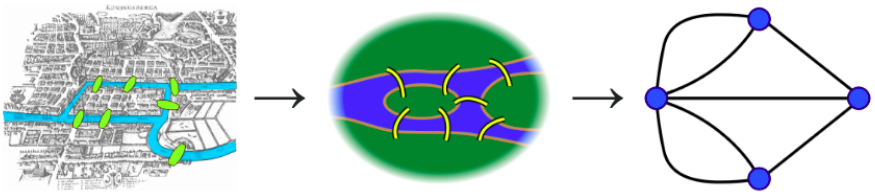


Figure 1.2: A depiction of what the problem looks like then and now. On the left, a map of Seventeenth-centuri Königsberg with the bridges in question highlighted. In the middle, a visualization of how Euler graphically represented the problem and on the right, how we represent the issue today, in modern graph theory.

Euler's solution came from the observation that by walking in the graph, except for the start and finish nodes, one must enter a node as many times as he exists, so the nodes must be touched by even numbers of bridges. This made the connection between walks and node degrees in a graph, meaning that the necessary and sufficient condition for the desired walk is that the graph may have exactly zero or two nodes of odd degree. Since in the problem, all land masses are connected by odd number of bridges, the proposed walk is impossible.

Euler's contribution was two-fold: firstly, it is considered to be the first theorem in graph theory and the theory of networks and secondly, the realisation that the information in the problem was the number of bridges and their endpoints represented the beginning of the development of a new area of mathematics named topology.

1.4.2 Graph Theory

Today, networks have applications in a vast number of domains: from sociology, economics, finance to statistical physics, electrical engineering, climatology and neuroscience. Graph theory is the framework for the exact mathematical treatment of complex networks. An undirected (or directed) graph $G = (\mathcal{N}, \mathcal{L})$ consists of two sets $\mathcal{N}$ such that $\mathcal{N} \neq \emptyset$ and $\mathcal{L}$ represents the set of unordered (or ordered) pairs of elements of $\mathcal{N}$:

- $\mathcal{N} \equiv \{n_1, n_2, ..., n_N\}$ = the nodes (or vertices, points) of G
- $\mathcal{L} \equiv \{l_1, l_2, ..., l_K\}$ = the links (or edges, lines) of G

$G(N, K) = (\mathcal{N}, \mathcal{L})$ represents a graph with N nodes and K edges.

The nodes are reffered to by their indices, their order i in the set $\mathcal{N}$ and the edges, by the two indices of the nodes (i, j) they join together. The usual graphical

representation is composed of dots for the nodes and lines between the nodes which are connected through an edge. If the graph is directed, then the line transforms into an arrow. The links can also be weighted, meaning there are more connections between the same nodes and this would be graphically represented by making these lines thicker.

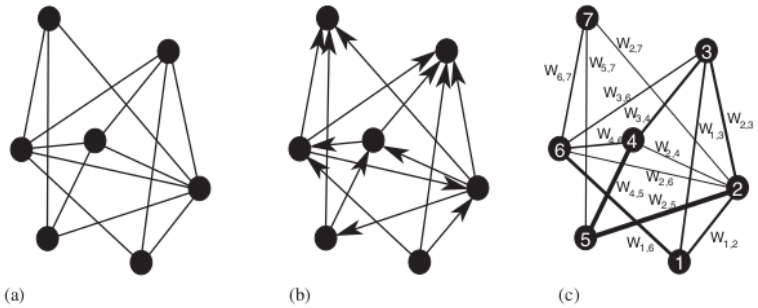


Figure 1.3: Graphical representation of a few types of graphs, each with $N = 7$ nodes and $K = 14$ edges: (a) undirected, (b) directed, the arrows showing the direction of each link and (c) weighted undirected, each link's weight $W_{i,j}$ reported on the respective line.

A few fundamental elements of a graph:

- a *walk* fom node i to node j = alternating sequence of nodes and edges that begins in i and ends in j;
- a *trail* = a walk in which no edge is repeated;
- a *path* = a walk in which no node is repeated;
- the *shortest path* = the walk of minimal length between two nodes;
- a *cycle* = a closed walk of at least three nodes, usually defined by it's length k or C_k (example: C_3 is a triangle, C_4 is a quadrilater).

Finally a graph is mathematically described by it's *adjacency matrix* $\mathcal{A}$, a $N \times N$ matrix whose entries a_{ij} are either 1 if link l_{ij} exists or 0 otherwise.

The properties and characteristics of the graphs that interests us will be defined when we construct and analyze seismic networks later in Chapter 4.

Chapter 2

Models that show SOC behaviour

In this chapter a few models are presented in order to better understand Self-Organized Criticality and to introduce the reader into the space of seismic models.

2.1 Flyvbjerg's Simplified Sandpile Model

In his work [4], Flyvbjerg illustrates, as the name suggests, the simplest possible Self-Organized Critical system. His model is a simplification of the usual sandpile model, which we will describe in detail in the next section, which he calls the "random neighbor model".

Consider N dynamical sites each capable of containing z_{max} grains of sand. The simplest case, each can contain 1 or 0 sand grains, so $z_{max} = 2$. With discrete time, at each step a random site is chosen and a sand grain is dropped on it. If it already contained a grain, the site topples and the grains redistribute to neighbors. These neighbours can be of two types: normal N, or absorbing M:

- $N = N_0 + N_1$ = total number of sites; N_0 = unoccupied, N_1 = occupied
- M = absorbing sites, having the purpose of eliminating sand from the model, i.e. keeping it's dissipative characteristic.

When a topple happens there are 3 outcomes:

1. Grain falls on empy site with probability $N_0/(N + M)$.
2. Grain falls on occupied site with probability $N_1/(N+M)$, resulting in another toppling.
3. Grain falls on absorbing site with probability $M/(N+M)$, exiting the system.

The phase space of this system is thus fully characterized by 2 integer variables, i.e. having only 2 degrees of freedom:

1. $N_1(\tau)$ = the number of particles in the sandpile = the number of sites occupied, since at $z_{max} = 2$, the site topples.
2. $n(\tau)$ = the number of sand particles in the avalanche.

Flyvbjerg proceeds to show that in mean field approximation, $N \to \infty$, $M \to \infty$ and $M/N \to \infty$, the model is gradually driven to a state which is critical, indicated by the absence of a characteristic scale in avalanche sizes. Furthermore, if you consider the number of occupied sites $N_1 = N/2$, it acts as an attractor point for

the medium's dynamics up to errors and fluctuations of order $\sqrt{N}$. The fluctuations are crucial for the self-organizing characteristic of the model.

By describing the evolution of the system using the master equation involving probability

$$\begin{aligned} P(N_1, n; \tau + 1) = &\frac{N_1 + 1}{N + M} P(N_1 + 1, n - 1; \tau) \\ &+ \frac{N - N_1 + 1}{N + M} P(N_1 - 1, n + 1; \tau) \\ &+ \frac{M}{N + M} P(N_1, n + 1; \tau). \end{aligned} \tag{1}$$

By introducing scaling variables x, y and t and scaling function $f(x, y; t)$:

$$x = (N_1 - N/2)/\sqrt{N}, \tag{2}$$
$$y = n/\sqrt{N}, \tag{3}$$
$$t = \tau/N, \tag{4}$$
$$\mu = M/\sqrt{N}, \tag{5}$$
$$f(x, y; t) = NP(N_1, n; \tau). \tag{6}$$

We get the equation for the probability density $f(x, y; t)$ that the medium is in state x and the avalache has size y at time t:

$$\partial_t f = [\underbrace{\frac{1}{2}(\partial_x - \partial_y)^2}_{\text{diffusive term}} + \underbrace{2(\partial_x - \partial_y)x}_{\text{describes SOC}} + \underbrace{\mu \partial_y}_{\text{dispersive term}}]f. \tag{7}$$

Using these variable transformations to reach the equation above, it is shown that an increase of x by infinitesimal amount dx initiates an avalanche in $y = 2/\sqrt{N}$ with probability $\sqrt{N}/2dx$.

Also the distribution $p(x)$, i.e the state x of the system between avalanches changes with each step and as expected through the central limit theorem is a normal distribution. However, the mean is not zero, showing that the mean field result for the average state $N_1 = N/2$ has a positive correction $\sqrt{N}$.

The underlying conclusion of this model is that Flyvbjerg proves that a system with only two independent degrees of freedom can be SOC and that there isn't a simpler possible SOC.

2.2 The Sandpile Model

2.2.1 Bak, Tang and Wiesenfeld's Explanation of 1/f Noise

In their work [5], Bak, Tang and Wiesenfeld (BTW) demonstrate that dynamical systems with spatial degrees of freedom naturally evole into a self-organized critical point. This analysis was fueled by two problems which were unexplained before:

- the anomalous flicker noise, or "$1/f$" noise found for transport in systems like resistors, flow of rivers, luminosity of stars. The low-frequency power spectra of these systems display a very robust power-law behaviour.

- the self-similiar fractal structures of spatially extended objects, like cosmic strings, coastal lines, mountain ranges. Again, a power-law is detected in the analysis of spatio-temporal correlations extending over large periods.

In self-organization, criticality loses it's meaning from phase transitions, where a parameter (temperature for example) needs to be tuned so that the system reaches the critical point; it now represents an attractor whose scaling properties are insensitive to such parameter changes. The system, starting far from equillibrium arranges itself into a state that is barely stable.

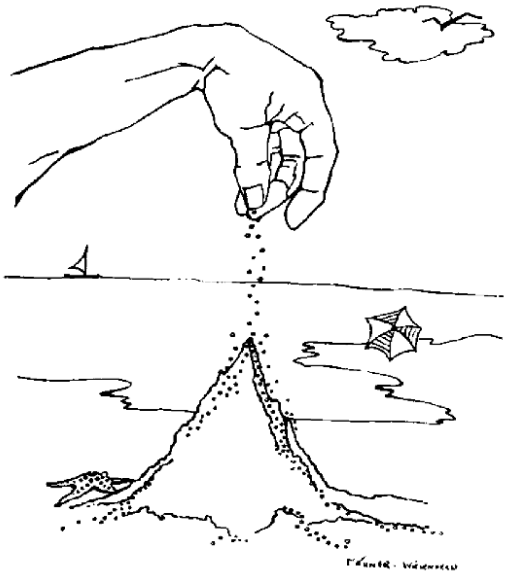


Figure 2.1: An illustration from "How Nature Works" [10], a drawing by Elaine Wiesendfeld in which the dropping of grains of sand on a little pile on the beach is pictured.

The model is described as a two-dimensional sandpile, a cellular automaton comprising of interactions of an integer variable z (number of sand grains in a site) with it's nearest neighbours. A grid of sites is created, with boundary conditions $z = 0$. The sites are updated at random by dropping sand on them, and when

$z > K$ (a threshold value) redistribution occurs:

$$z(x, y) \to z(x, y) - 4, \tag{8}$$
$$z(x \pm 1, y) \to z(x \pm 1, y) + 1, \tag{9}$$
$$z(x, y \pm 1) \to z(x, y \pm 1) + 1. \tag{10}$$

The site is initiated with random values for each site $z >> K$ and it evolves until it stops $z < K$. Simulations show that the distribution of cluster sizes and time scales follow power-law distributions.

The self-organized critical state of minimally stable clusters is very robust on all length scales and they ceate fluctuations on all time scales.

2.2.2 Bak's Bureaucrats Model

In his famous book, How Nature Works [10], Per Bak describes a re-imagination of the sandpile model, called the bureaucrats model. The principle is similar and it is illustrated in the following picture:

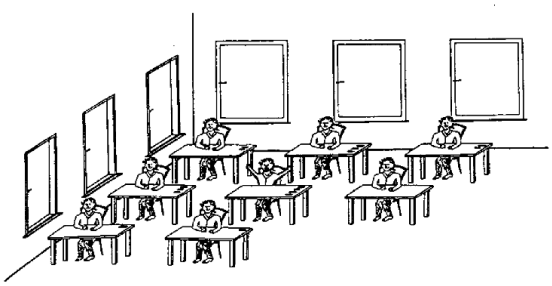


Figure 2.2: The "Office" version of the sandpile model. At each timestep a document is placed on the desk of a bureaucrat. When he finds four or more documents on his desk, he redistributes them, one to each of his neighbours, or he throws it out the window if he is at the edge of the room (the boundary conditions).

2.3 Olami-Feder-Christensen Slider-Blocks Model

In their paper [11], Olami, Feder and Christensen (OFC) argue that earthquakes are probably the most relevant paradigm of SOC. One of the first realisations in this space is the Gutenberg-Richter law which states that the rate of occurence of earthquakes of magnitude M greater than m is given by:

$$log_{10}N(M > m) = a - bm. \tag{11}$$

with the parameter b having a wide range of values for different faults. The energy released during the earthquake is thought to increase exponentially with the magnitude:

$$log_{10}E = c + dm \tag{12}$$

Thus, the law transforms into a power law for the number of observed earthquakes with energy greater than E:

$$N(E_0 > E) \approx E^{-b/d} = E^{-B} \tag{13}$$

OFC devised a 2D generalization of the Burridge-Knopoff [12] 1-dimensional spring-block model for earthquakes. Their model have some interesting qualities:

- displays robust SOC behaviour over a number of conservation levels;
- the amount of conservation impacts the obtained power-laws;
- as conservation increases, the behaviour transitions from localized to nonlocalized;
- this conservation dependence explains the variance of the parameter in the Gutenberg-Richter law.

The model is constructed as follows :

1. consider a 2D lattice of blocks;
2. each block is connected with springs to its four nearest nighbours and to a rigid moving plate that connects all the blocks;
3. each block interacts frictionally to a fixed plate that they move on;
4. blocks are driven by the continuous displacement of the moving plate;
5. when the force of one of the blocks reaches a threshold value F_{th}, the block slips, redefining the forces of it's neighbours;
6. the slip of one block may result in an avalanche of slips by the nearest blocks resulting in a chain reaction.

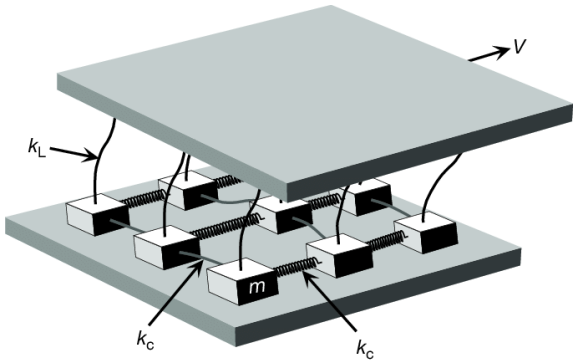


Figure 2.3: Slider Blocks Model: identical blocks of mass m are linked with each other by identical springs with elastic constant k_c and to a moving plate in constant motion with velocity v by springs with elastic constant k_L. Blocks are in contact with a fixed plate, with friction.

The total force exerted by the springs on a block (i, j) is:

$$F_{i,j} = K_1[2dx_{i,j} - dx_{i-1,j} - dx_{i+1,j}] \tag{14}$$
$$+ K_2[2dx_{i,j} - dx_{i,j-1} - dx_{i,j+1}] + K_L dx_{i,j} \tag{15}$$

When the plates move relative to eachother and the force exerted on a block reaches the threshold, it redistributes as follows:

$$F_{i\pm1,j} \to F_{i\pm1,j} + \delta F_{i\pm1,j}, \tag{16}$$
$$F_{i,j\pm1} \to F_{i,j\pm1} + \delta F_{i,j\pm1}, \tag{17}$$
$$F_{i,j} \to 0, \tag{18}$$

with the increases in forces given by:

$$\delta F_{i\pm1,j} = \frac{K_1}{2K_1 + 2K_2 + K_L} F_{i,j} = \alpha_1 F_{i,j}, \tag{19}$$
$$\delta F_{i,j\pm1} = \frac{K_2}{2K_1 + 2K_2 + K_L} F_{i,j} = \alpha_2 F_{i,j}. \tag{20}$$

For simplification, the isotropic case is considered: $K_1 = K_2 = k_c$, subsequently the elastic ratios become $\alpha_1 = \alpha_2 = \alpha = \frac{k_c}{4k_c + k_L}$. The model evolves as follows:

1. initialize all sites' forces to a random value between 0 and 1;
2. check if any $F_{ij} \geq F_{th}$;
3. if yes, redistribute the force according to:

$$F_{n,n} \to F_{n,n} + \alpha F_{i,j}, \tag{21}$$
$$F_{i,j} \to 0. \tag{22}$$

4. repeat from step 2 until the earthquake fully evolves;
5. locate the block with F_{max} and add $F_{th} - F_{max}$ to all blocks and return to the second step.

The probability distribution of the total number of relaxations of the earthquakes is measured. This quantitiy is proportional to the energy release during an earthquake. This cellular automaton model is found to show SOC behaviour for a large number of α values.

Chapter 3

Seismic Database

For each seismic region we wish to study, the first step is to collect the earthquakes databases available online, published by the respective region institute:

- **Vrancea(Romania)** - National Institute for Earth Phyiscs [13].
- **California(USA)** - Southern California Earthquake Center [14].
- **Italy** - National Institute for Geophysics and Vulcanology [15].
- **Japan** - Japan Meteorological Agency [16].

3.1 What is a Seismic Database

From the available data we select the event date, latitude, longitude, depth and magnitude and store the information on our machines so that it is easily accesible through an mySQL script in Python used to call the database, selecting all the events, or only a certain amount of events based on conditions put upon the date, magnitude or geographical position.

For the four regions mentioned earlier, the total data available spans the following timeframes and coordinates:

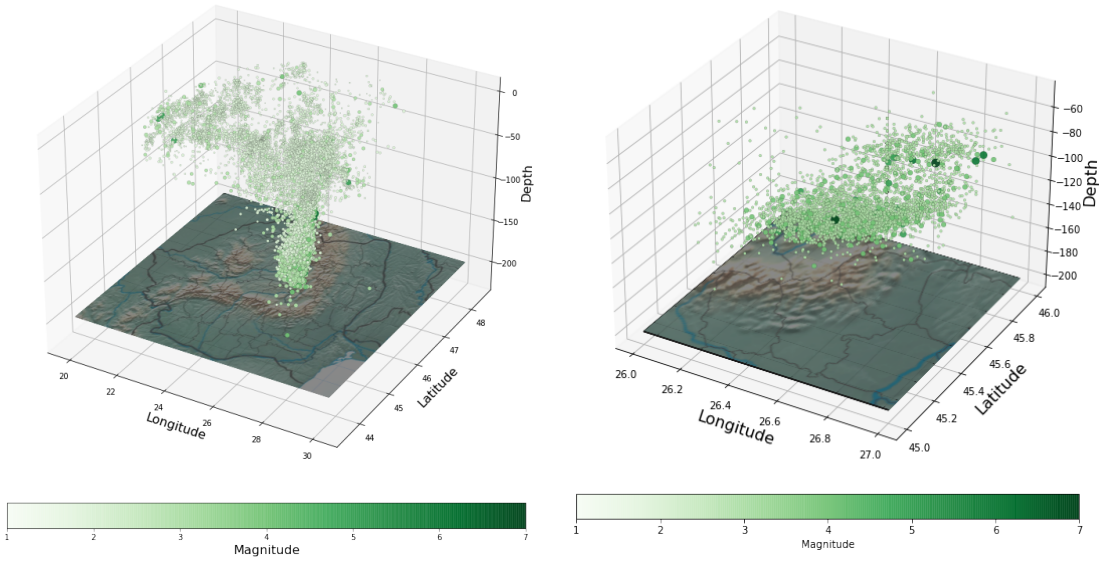
Seismic Databases				
Seismic Zone	Timeframe	Latitude	Longitude	Depth
Romania	0984-01-01 2021-02-28	43.594°N 48.23°N	20.1°E 26.14°E	0 218.4
California(USA)	1932-01-02 2020-12-31	32°N 37°N	-114°W -122°W	0 51.1
Italy	1986-01-01 2020-12-31	30.61°N 47.998°N	-6.08°W 36.02°E	0 644.4
Japan	1919-01-11 2019-08-31	17.41°N 54.97°N	114.78°E 160.17°E	0 698.4

For our analysis it is best to apply some restrictions when extracting data from the databases, in order to have a proper statistical relevance. For example, in Romania, the primary seismic zone, where most large magnitude earthquakes occur, is in Vrancea county, so in order to select only the earthquakes in this seismic region we would restrict the geographical coordinates as such: $45°2'N - 46°N$ latitude, $26°E - 27°E$ longitude and the depth between 50 km and 197 km.

Also, for each sesimic zone, we restrict the timeframe by extracting data starting with a year when proper data collection started. For example, in Romania, the rigurous study of the Vrancea seismic zone started after the 7.4 magnitude earthquake of 1977-03-04, so this is why we restrict our statistic for earthquakes starting from the year 1976. For the other regions, the timeframes of our analysis are presented below.

In the following we present earthquakes distributions as scatterplots with size and colour of the points varying with the magnitude of the earthquake, for the four seismic zones analyzed. The magnitude scale is present along with the plots and the timeframe, coordinate restrictions and number of events are presented in the description of each graph:

Romania and Vrancea - Earthquakes Distribution



(a) Romania: 29186 earthquakes,
from 1976-02-03 13:29:16 to 2021-02-28 16:57:29.

(b) Vrancea: 7512 earthquakes,
from 1976-08-19 19:03:01 to 2021-02-28 00:11:55.

Figure 3.1: Earthquakes distribution for Romania (left) and Vrancea (right) seismic zones with magnitude > 1. Largest magnitude recorded: 7.4 on Richter Scale.

California(USA) - Earthquakes Distribution

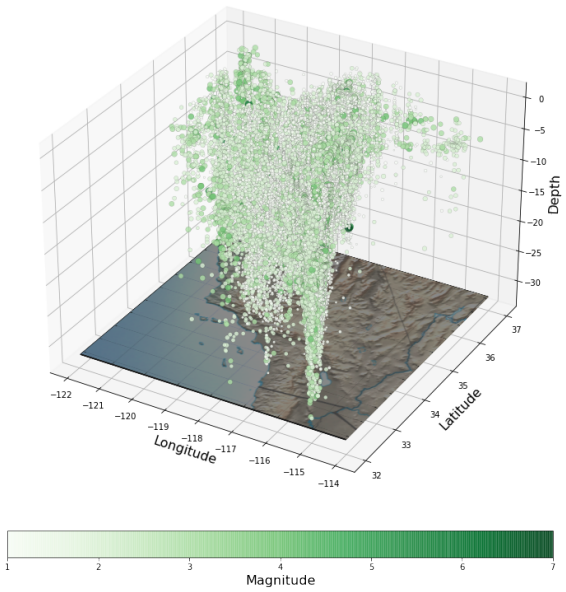


Figure 3.2: Earthquakes distribution for California(USA), seismic zone: 221113 events with magnitude > 1, from 1984-01-01 18:27:55 to 2020-12-31 23:04:53. Largest magnitude recorded: 7.2 on Richter Scale.

Italy - Earthquakes Distribution

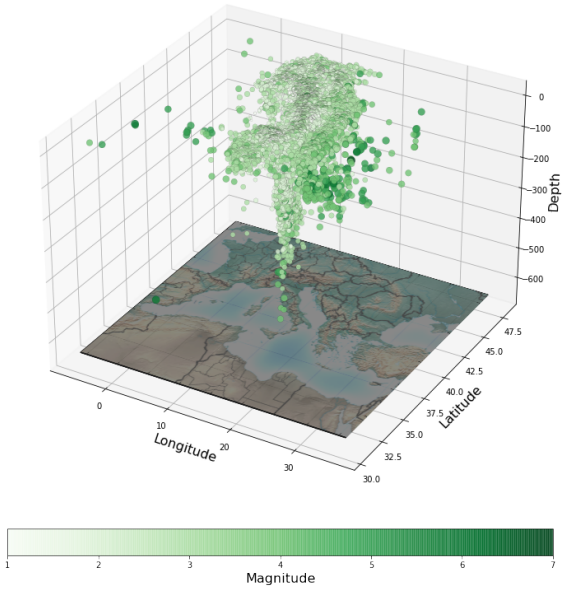


Figure 3.3: Earthquakes distribution for Italy seismic zone: 319567 events with magnitude > 1, from 1986-01-01 17:22:53 to 2020-12-31 23:41:18. Largest magnitude recorded: 7 on Richter Scale.

Japan - Earthquakes Distribution

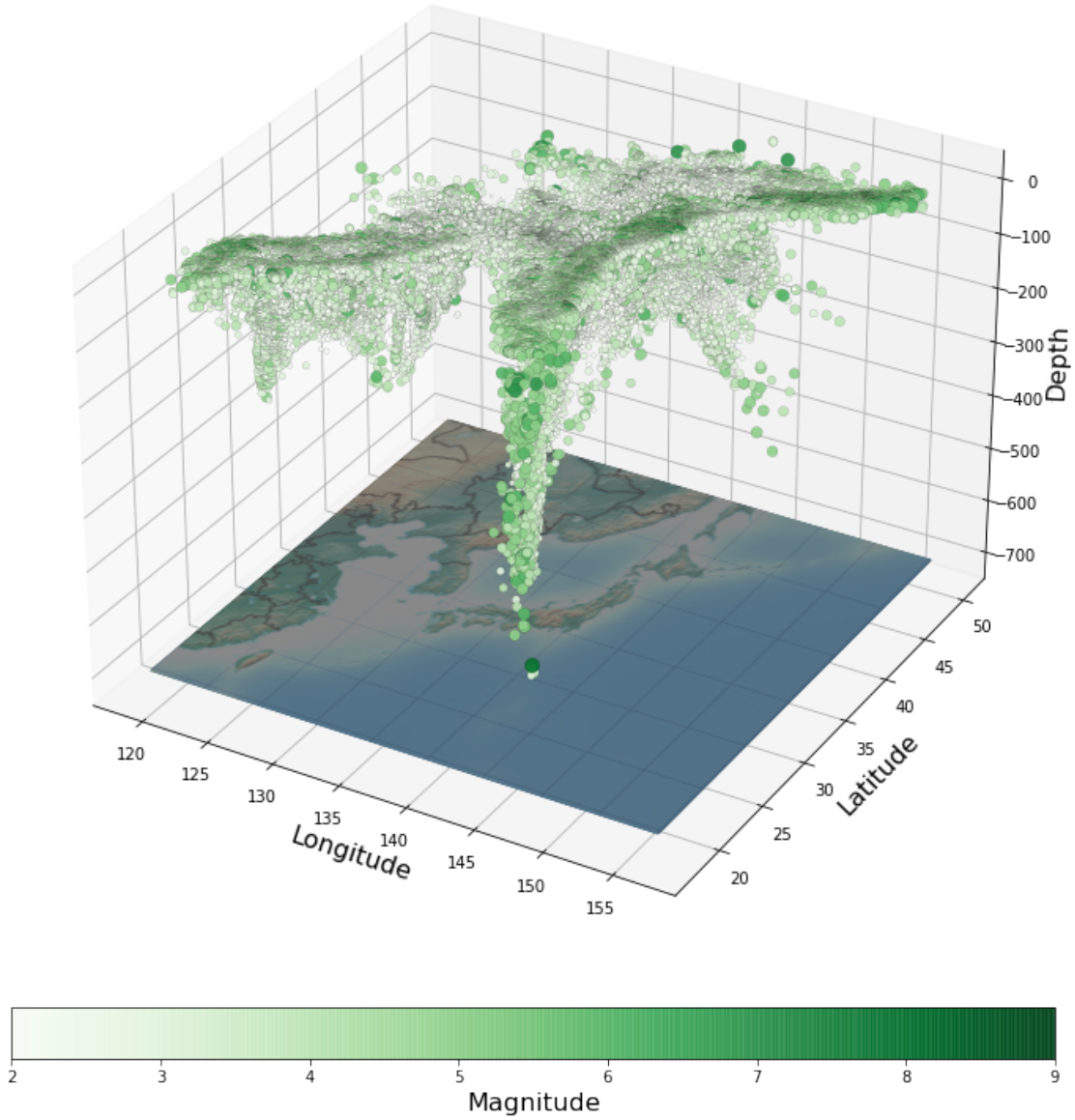


Figure 3.4: Earthquakes distribution for Japan seismic zone: 595713 events with magnitude > 2, from 1992-01-01 00:54:03 to 2019-08-31 23:54:24. Largest magnitude recorded: 9 on Richter Scale.

3.2 Seismic Table Construction

This section explains how to construct the seismic table containing the relevant information used in our upcoming computations. Into the same Python script used when calling the database using mySQL we implemented two aditional computations:

Firstly, we can roughly estimate the energy release of each event by converting the moment magnitude M_W to energy using the equation $logE = 5.24 + 1.44M$ where M is the magnitude.

Secondly, in order to build our earthquakes network, we need to divide the spatial region selected into cubes and place each event in it's respective cube. The split is done as follows:

- A side length for the cube is chosen : $sideLength = 5$ / 10 km, depending on how much granularization of the network you wish to have.
- For each dimension, the total number of kilometers is computed[1], then this total is divided to the length of the cube side, identifying the total number of cubes that fit in each dimension. In the following, the examples are for latitude:

$$\begin{aligned} &X = round(\frac{(maxLatitude - minLatitude)111}{sideLength}), \\ &X = \text{Total number of cubes on the latitude direction,} \\ &maxLatitude = \text{The maximum latitude at which an event is located,} \\ &minLatitude = \text{The minimum latitude at which an event is located.} \end{aligned} \tag{1}$$

- Then for each event, it's latitude index is assigned by converting it's dimension to the corresponding index:

$$\begin{aligned} &x = floor(\frac{(i - minLatitude)X}{maxLatitude - minLatitude}) + 1, \\ &i = \text{The latitude of the event,} \\ &x = \text{The latitude index of the event.} \end{aligned} \tag{2}$$

- After doing the same for Longitude and Depth, a *cubeIndex* is assigned to each event; this indexing taking values from 1 to the total number of cubes that we split the seismic zone into.
- Finally, for each cube, the real coordinates of it's center are also presented (*cubeLatitude*, *cubeLongitude* and *cubeDepth*).

[1] for each region, 1° latitude corresponds to 111 km, whereas for longitude, it changes depending on position on the globe. 1° longitude in Vrancea(Romania) is 79 km, in California(USA) 94 km, in Italy 84 km and in Japan 91 km.

This is an examle of how our earthquakes table would look like:

	date	latitude	longitude	depth	magnitude	energyRelease	x	y	z	cubeIndex	cubeLatitude	cubeLongitude	cubeDepth
0	1976-08-19 19:03:01	45.5400	26.3700	162.0	2.3	3.564511e+08	8	6	23	6434	45.5378	26.3481	162.5
1	1976-09-07 17:38:08	45.6200	26.5000	155.3	3.6	2.654606e+10	10	9	21	5914	45.6279	26.5380	152.5
2	1976-10-01 17:50:43	45.6800	26.4900	146.0	6.0	7.585776e+13	11	8	20	5609	45.6730	26.4747	147.5
3	1977-03-04 19:21:54	45.7700	26.7600	94.0	7.4	7.870458e+15	13	13	9	2533	45.7631	26.7911	92.5
4	1977-03-04 21:21:01	45.2200	26.6500	141.0	3.0	3.630781e+09	1	11	19	5365	45.2225	26.6646	142.5
...	...	...	...	...	...	...	...	...	...	...	...	...	...

Figure 3.5: The Earthquakes Table - our fundamental tool for the following analysis, containing the basic information from the databases available online (*date, latitude, longitude, depth and magnitude*) and our computations: the *energyRelease* and the "cube parametrization" with indexing both in the "cubes space": x, y, z and the real space: *cubeLatidue*, *cubeLongitude*, *cubeDepth*. The data in this example represents the first 5 events in the Vrancea seismic zone, starting with the year 1976, as a pandas DataFrame in python.

Chapter 4

Seismic Network

In this chapter we explain how the seismic network is constructed, based on all the data processed from the seismic database into our earthquakes table and we show two methods of analyzing this network and the results obtained, mainly the scale-free nature of these characteristics through suggestive visualizations.

4.1 How to construct a Seismic Network

After splitting the region into cubes and assigning each earthquake into it's respective cube we can begin building our network. Similar to [17], this is done by iterating chronologically through the earthquakes table as follows:

1. When you encounter an event create a node with the label $cubeIndex$ defined earlier in our table. Our graph nodes represent the cubes that we split our region in, so when we say cube/node we refer to the same thing.
2. Pick the next event chronologically and create a node for it.
3. Now we have two options, to consider edge weights[1] or not:
   - If we consider edge weights, first check if there is already an edge between the two nodes that we placed. If there is, add 1 to the weight of this edge; if there isn't, create the edge.
   - If we don't consider edge weights, simply create the edge between the nodes; if there is already an edge, creating a new one on top of the old changes nothing, at the end the edge still counts as a single one.
4. Two nodes may sometimes coincide with each other (succesive events in the same cube), forming a loop.
5. Repeat from step 2.
6. At the end, after the network is complete, for each node we can add different attributes based on the task we wish to perform on the graph. We can add the cube coordinates (x, y, z) or the real cube coordinates (cubeLatitude, cubeLongitude, cubeDepth). Also we can add the list of all the events that occured in that specific cube.

[1] Considering edge weights ultimately changes the connectivity distribution of the network (because certain nodes degrees will increase when taking into account the weight) so it is useful to study the network from both perspectives.

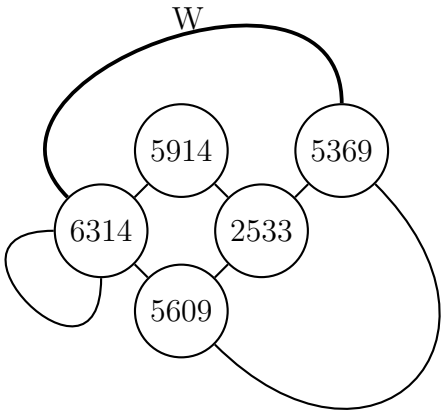


Figure 4.1: Seismic Network (Graph representation) example: Nodes are identified as $cubeIndex$. This indexing makes it easy to acces any information about events in that respective cube from our Earthquakes Table. Also, if the network is weighted, this can be represented on the edge as W, where each additional link between two nodes increases this value by 1.

So now we posses the two fundamental tools used in our following analysis, the earthquakes table and the seismic network

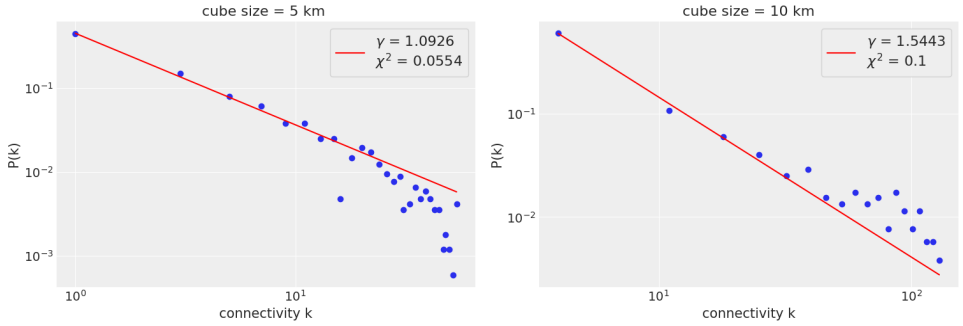
4.2 Centrality Measures of a Network

The fundamental measure to describe our seismic network is the connectivity distribution $P(k)$ [7]. This distribution comes naturally by realising the histogram of the nodes degree and computing the distribution. We also fit the log-log plot of this distribution and show that it is scale-free.

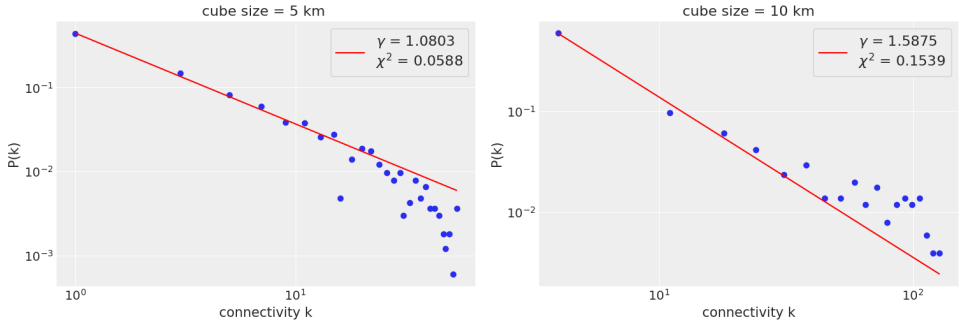
$$P(k) \sim k^{-\gamma} \tag{1}$$

The connectivity computations are made for various seismic networks (Vrancea(Romania), California(USA), Italy and Japan), with different magnitude restrictions, for 2 cube sizes: $5 \times 5 \times 5$ km and $10 \times 10 \times 10$ km, with and without edge weights.

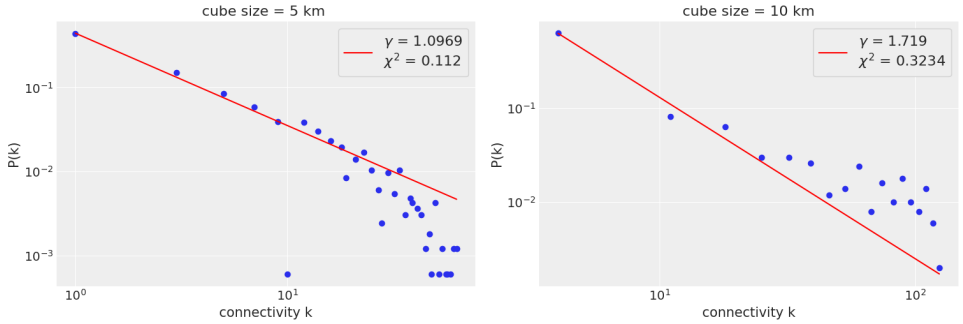
Vrancea - Connectivity



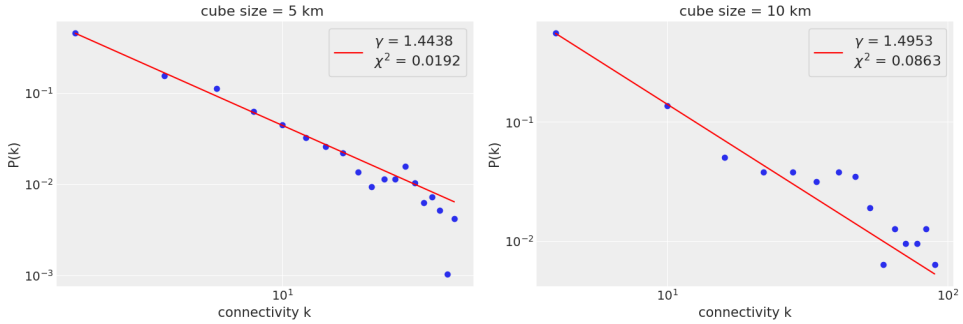
(a) $1 \leq magnitude$



(b) $2 \leq magnitude$



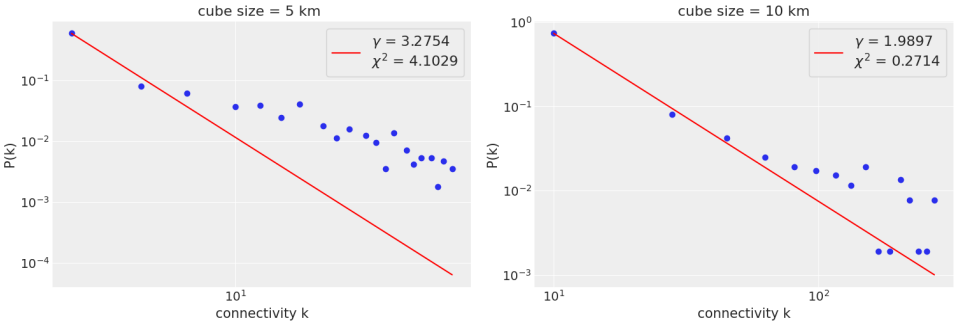
(c) $2 \leq magnitude \leq 4$



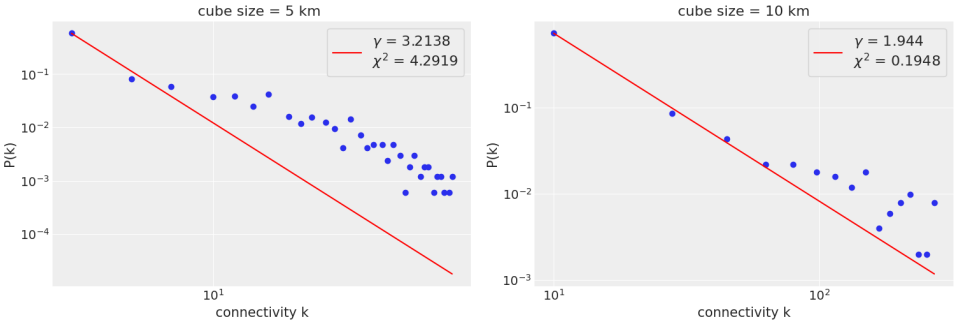
(d) $3 \leq magnitude \leq 7$

Figure 4.2: Connectivity distribution $P(k)$ for Vrancea in log-log and linear interpolation of the results, for a number of magnitude ranges. For each range, two granularization sizes of the seismic areas have been chosen: on the left $5 \times 5 \times 5$ km cubes and on the right $10 \times 10 \times 10$ km cubes. The exponent of the fit, γ ranges from ~ 1.08 to 1.756.

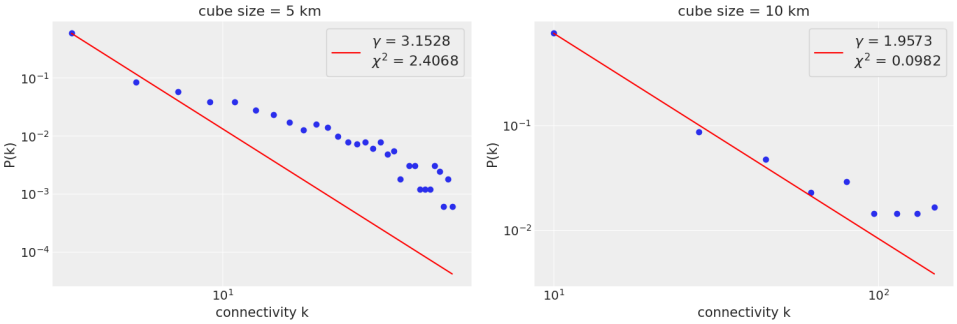
Vrancea - Connectivity Weighted



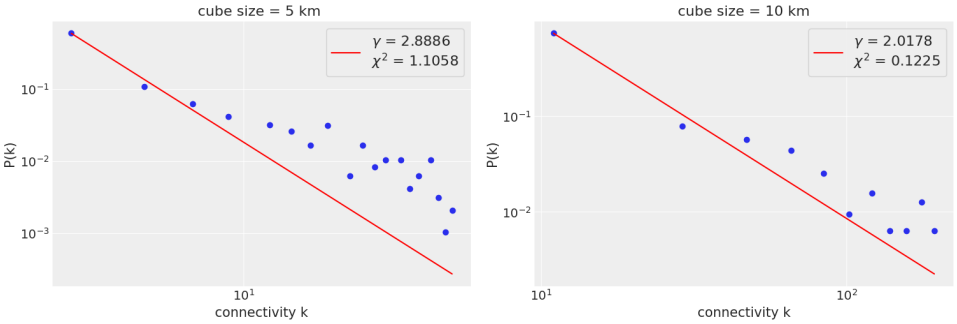
(a) $1 \leq magnitude$



(b) $2 \leq magnitude$



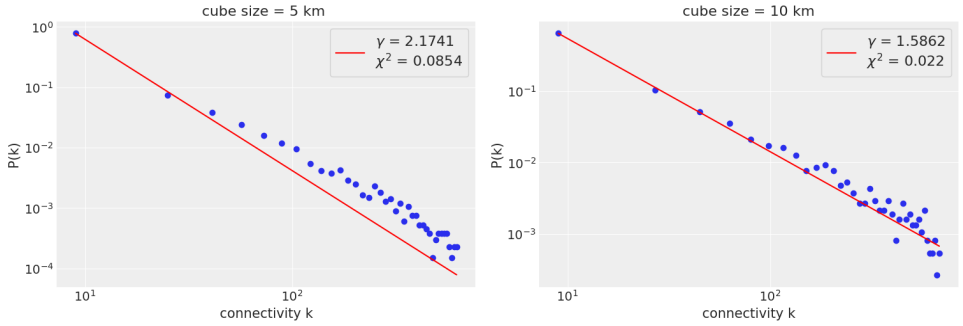
(c) $2 \leq magnitude \leq 4$



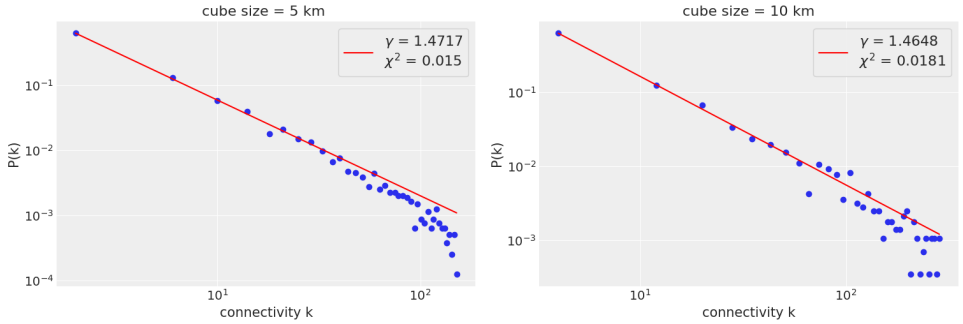
(d) $3 \leq magnitude \leq 7$

Figure 4.3: Weighted connectivity distribution $P(k)$ for Vrancea in log-log and linear interpolation of the results, for a number of magnitude ranges. For each range, two granularization sizes of the seismic areas have been chosen: on the left $5 \times 5 \times 5$ km cubes and on the right $10 \times 10 \times 10$ km cubes. The exponent of the fit, γ ranges from ~ 1.95 to 3.26.

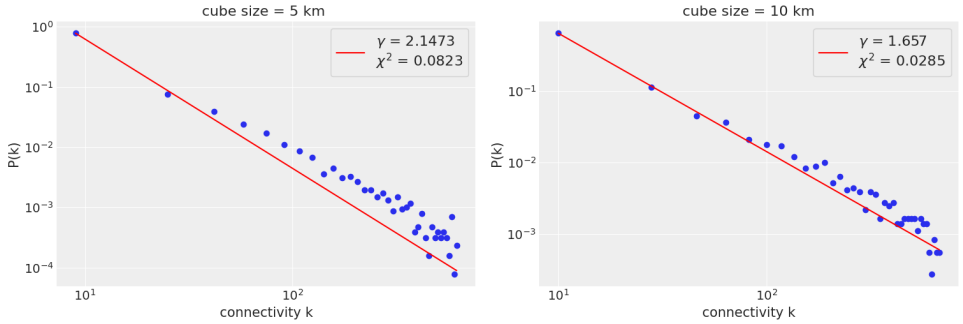
California - Connectivity



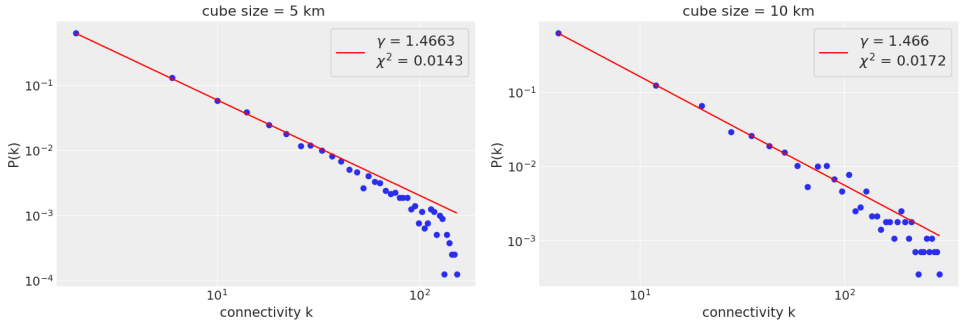
(a) $1 \leq magnitude$



(b) $2 \leq magnitude$



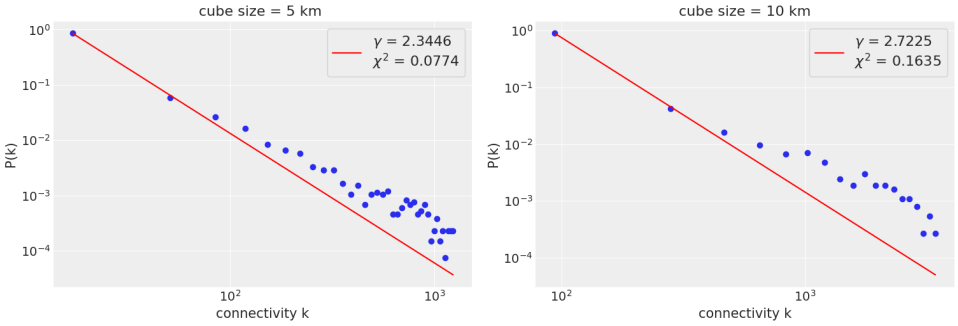
(c) $1 \leq magnitude \leq 3$



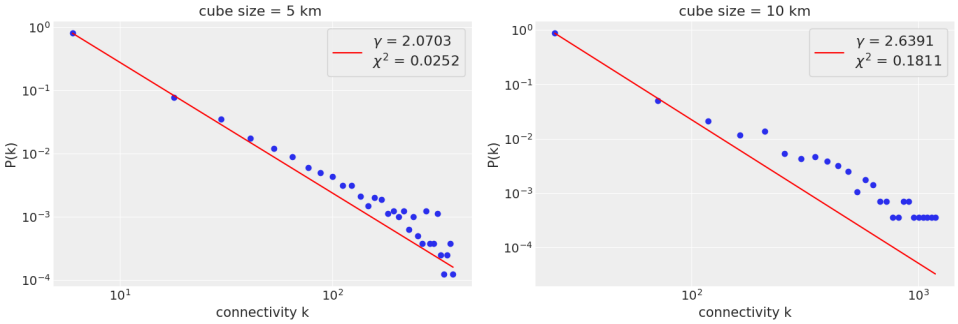
(d) $2 \leq magnitude \leq 4$

Figure 4.4: Connectivity distribution $P(k)$ for California in log-log and linear interpolation of the results, for a number of magnitude ranges. For each range, two granularization sizes of the seismic areas have been chosen: on the left $5 \times 5 \times 5$ km cubes and on the right $10 \times 10 \times 10$ km cubes. The exponent of the fit, γ ranges from ~ 1.45 to 2.16.

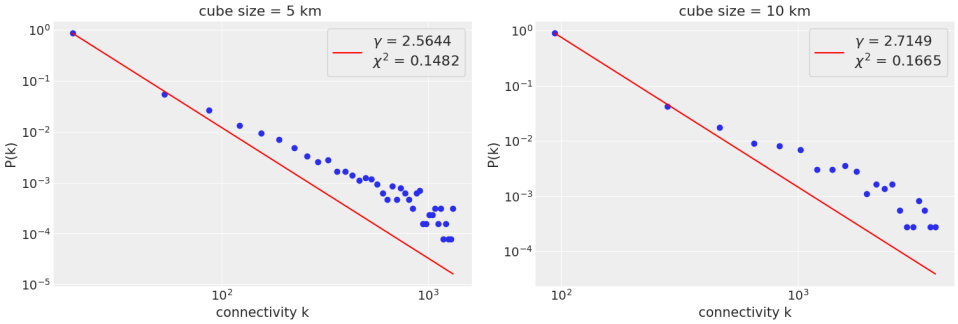
California - Connectivity Weighted



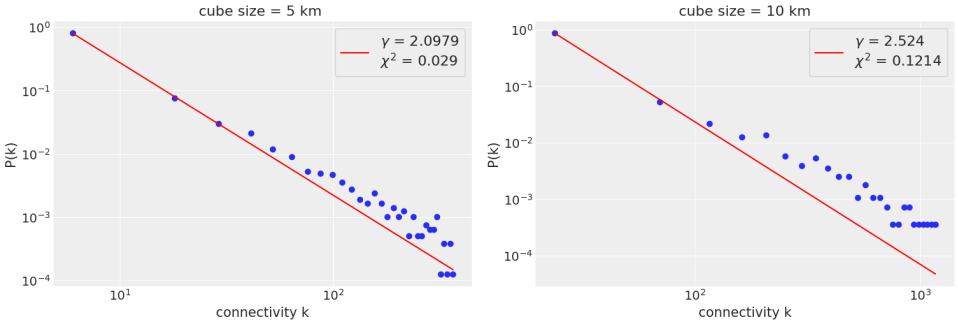
(a) $1 \leq magnitude$



(b) $2 \leq magnitude$



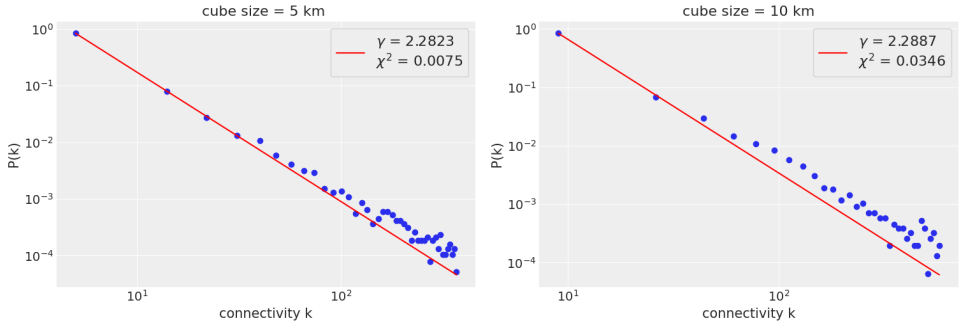
(c) $1 \leq magnitude \leq 3$



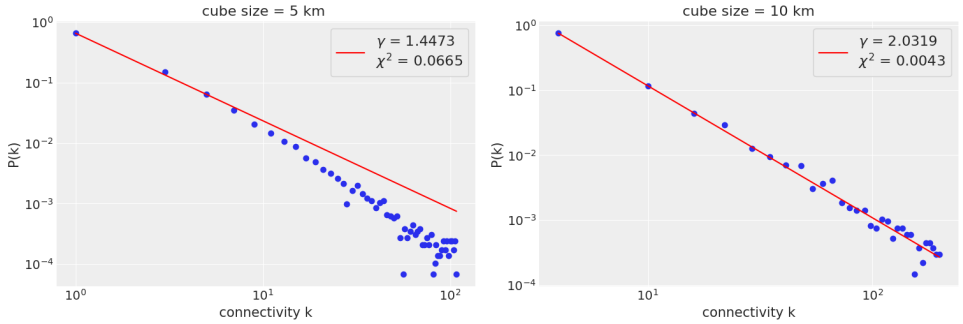
(d) $2 \leq magnitude \leq 4$

Figure 4.5: Weighted connectivity distribution $P(k)$ for California in log-log and linear interpolation of the results, for a number of magnitude ranges. For each range, two granularization sizes of the seismic areas have been chosen: on the left $5 \times 5 \times 5$ km cubes and on the right $10 \times 10 \times 10$ km cubes. The exponent of the fit, γ ranges from ~ 2.06 to 2.71.

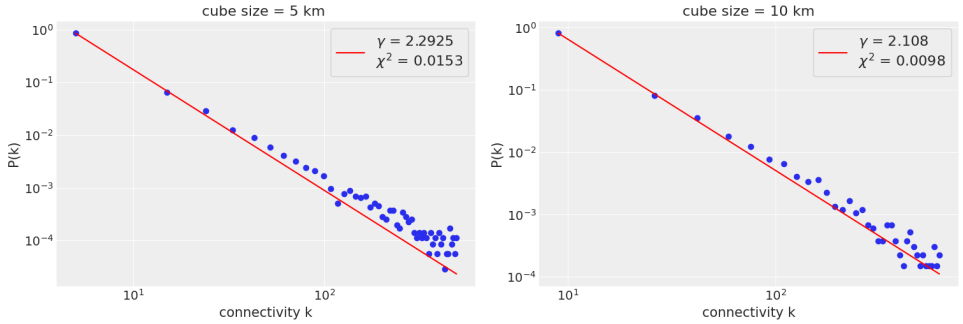
Italy - Connectivity



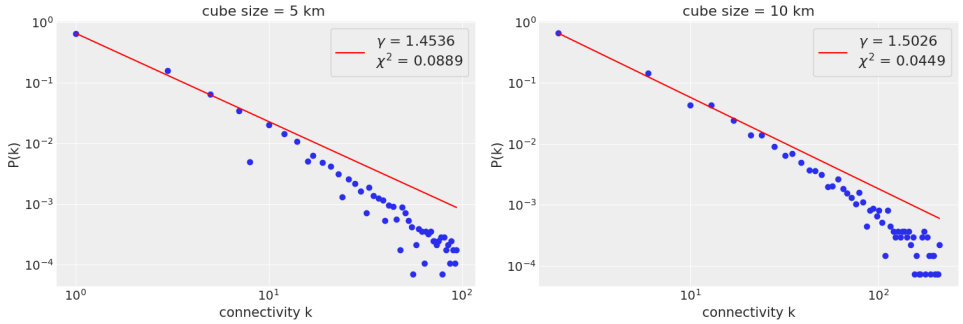
(a) $1 \leq magnitude$



(b) $2 \leq magnitude$



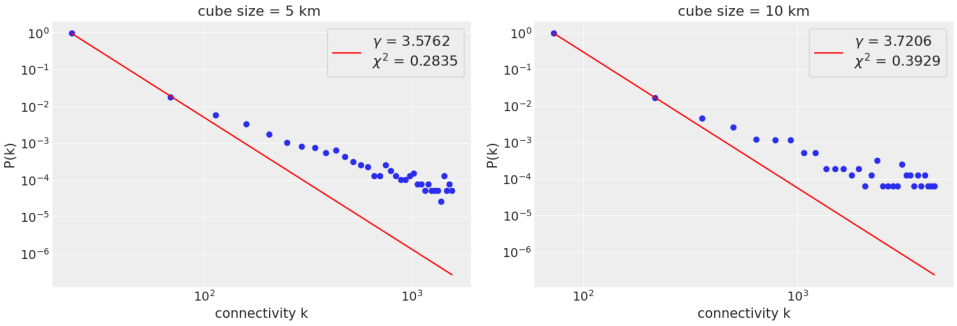
(c) $1 \leq magnitude \leq 3$



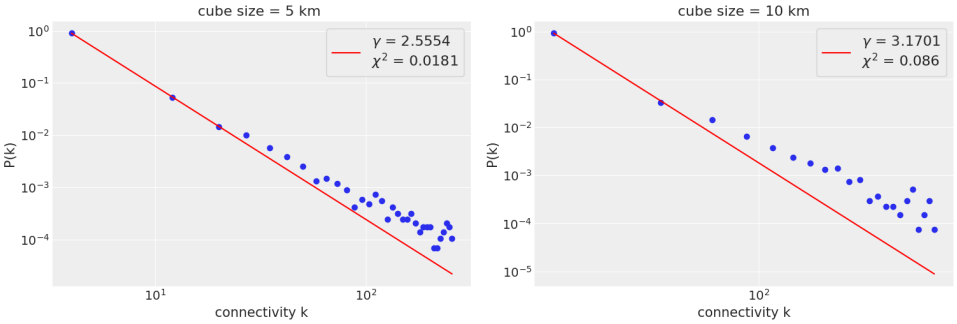
(d) $2 \leq magnitude \leq 4$

Figure 4.6: Connectivity distribution $P(k)$ for Italy in log-log and linear interpolation of the results, for a number of magnitude ranges. For each range, two granularization sizes of the seismic areas have been chosen: on the left $5 \times 5 \times 5$ km cubes and on the right $10 \times 10 \times 10$ km cubes. The exponent of the fit, γ ranges from ~ 1.44 to 2.3.

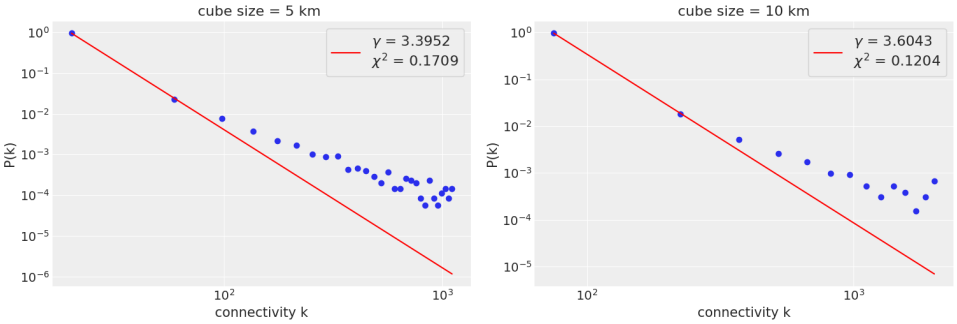
Italy - Connectivity Weighted



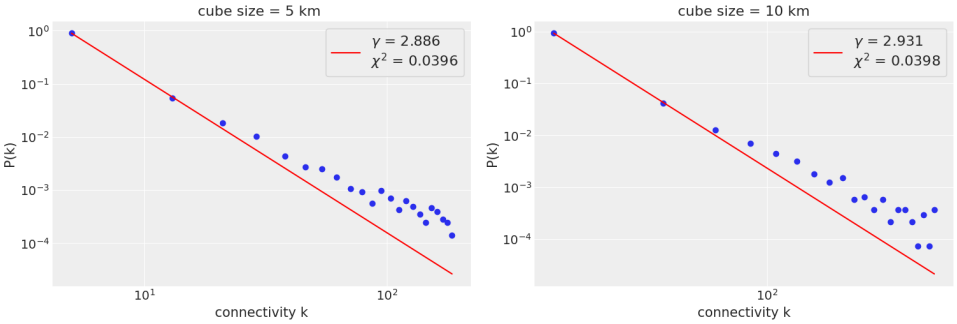
(a) $1 \leq magnitude$



(b) $1 \leq magnitude$



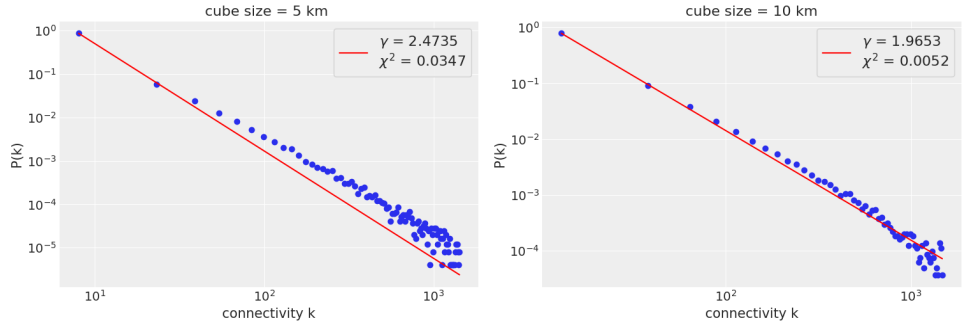
(c) $1 \leq magnitude \leq 3$



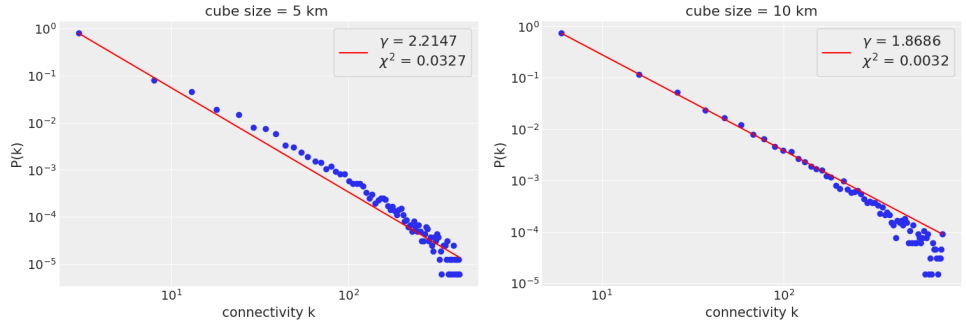
(d) $2 \leq magnitude \leq 4$

Figure 4.7: Weighted connectivity distribution $P(k)$ for Italy in log-log and linear interpolation of the results, for a number of magnitude ranges. For each range, two granularization sizes of the seismic areas have been chosen: on the left $5 \times 5 \times 5$ km cubes and on the right $10 \times 10 \times 10$ km cubes. The exponent of the fit, γ ranges from ~ 2.8 to 3.72.

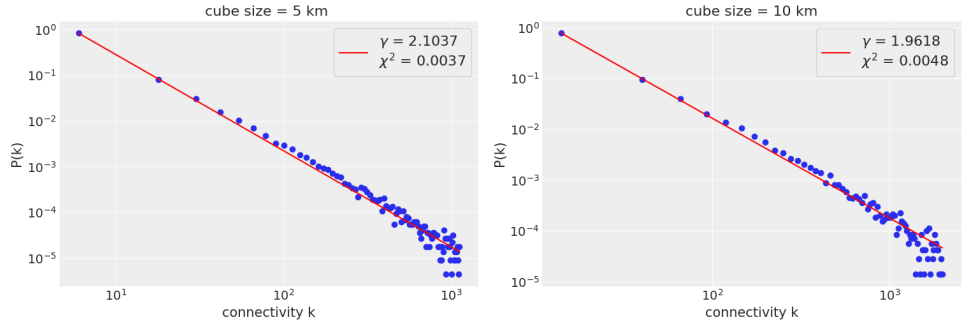
Japan - Connectivity



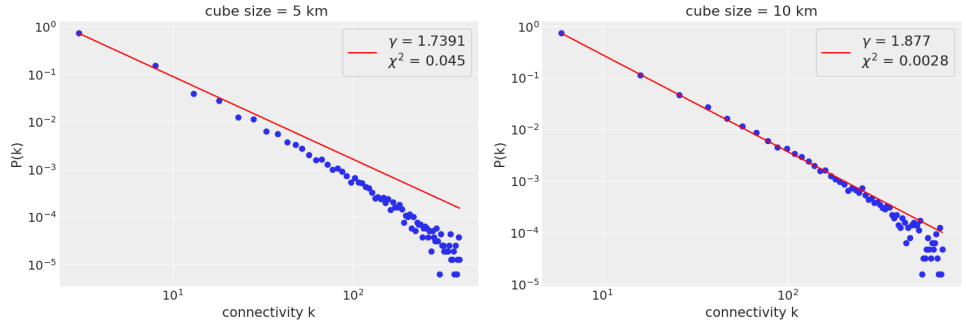
(a) $1 \leq magnitude$



(b) $2 \leq magnitude$



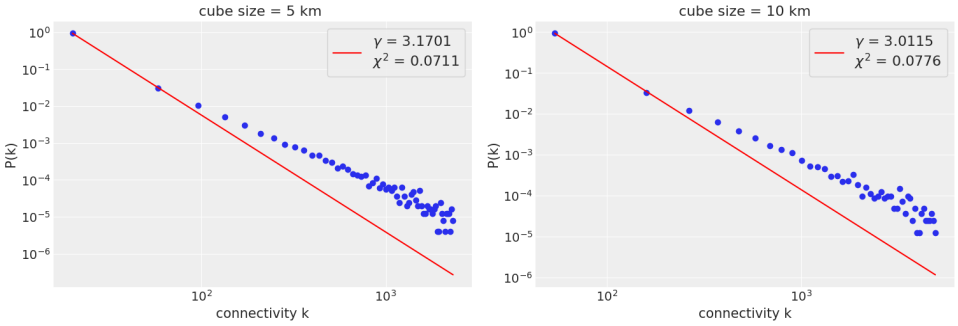
(c) $1 \leq magnitude \leq 3$



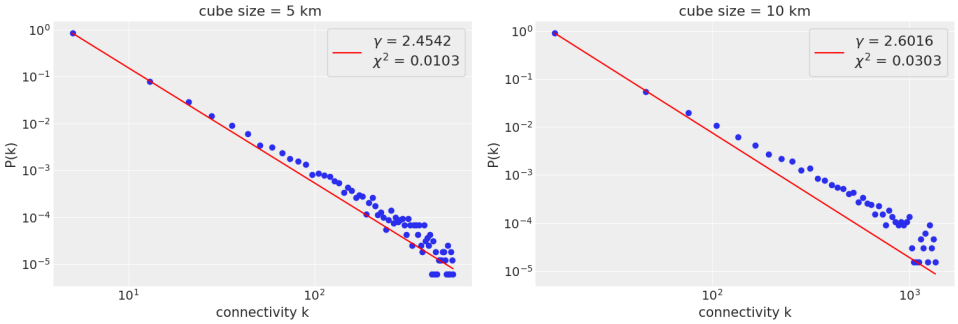
(d) $2 \leq magnitude \leq 4$

Figure 4.8: Connectivity distribution $P(k)$ for Japan in log-log and linear interpolation of the results, for a number of magnitude ranges. For each range, two granularization sizes of the seismic areas have been chosen: on the left $5 \times 5 \times 5$ km cubes and on the right $10 \times 10 \times 10$ km cubes. The exponent of the fit, γ ranges from ~ 1.87 to 2.5.

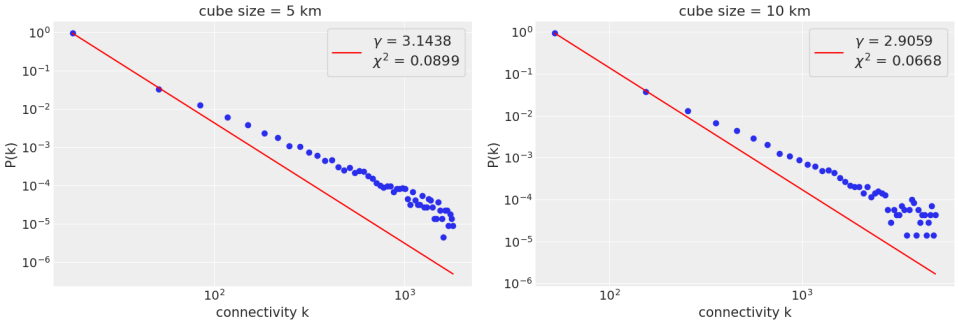
Japan - Connectivity Weighted



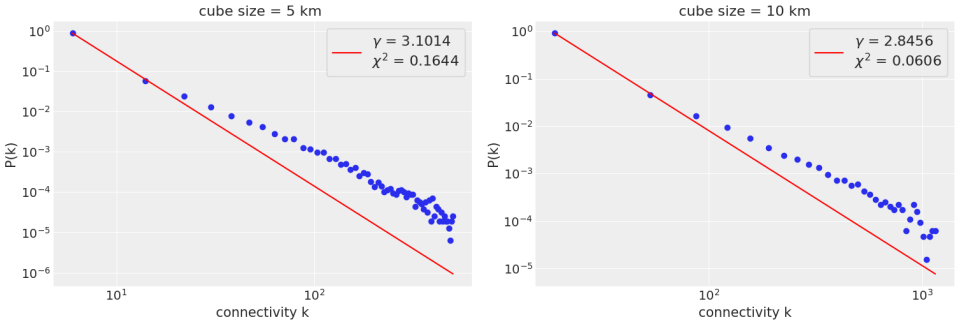
(a) $1 \leq magnitude$



(b) $2 \leq magnitude$



(c) $1 \leq magnitude \leq 3$



(d) $2 \leq magnitude \leq 4$

Figure 4.9: Weighted connectivity distribution $P(k)$ for Japan in log-log and linear interpolation of the results, for a number of magnitude ranges. For each range, two granularization sizes of the seismic areas have been chosen: on the left $5 \times 5 \times 5$ km cubes and on the right $10 \times 10 \times 10$ km cubes. The exponent of the fit, γ ranges from ~ 2.45 to 3.17.

4.3 Network Cycles and Motifs

In this section the general approach to define and detect the building blocks of a network is being described [7].

In social and biological networks it has been noticed that triangles are highly recurrent, so work has been done in identifying where and why these kind of patterns appear. They are generally called *cycles* of different lengths C_l and other subgraphs known as *motifs* $F_{n,l}$, which represent cycles that occur in *real networks* more frequently than in their corresponding random counterparts.

4.3.1 Motifs

Network motifs are sub-graphs that repeat themselves in a specific network or even among various networks. Each of these sub-graphs, defined by a particular pattern of interactions between vertices, may reflect a framework in which particular functions are achieved efficiently. They have gathered much attention as a useful concept to uncover structural and functional design principles of complex networks. In bioinformatics, network motifs have been applied to various applications including prediction of protein interactions, identifying breast cancer related genes, and detection of essential proteins. Although network motifs may provide a deep insight into the network's functional abilities, their detection is computationally challenging.

Given an undirected graph $G = (\mathcal{N}, \mathcal{L})$ and any possible small connected graph $F_{n,l}$ with n nodes and l links, we wish to find out if F is a significant subgraph of G. This is done by comparing the number of subgraphs of G isomorphic to F with the number of subraphs of a randomised network $G' = (\mathcal{N}, \mathcal{L})$ isomorphic to F.
The simplest approach to quantify the relevance of $F_{n,l}$ as a subgraph of G is based on the evaluation of the *Z-score*, defined as follows:

$$Z_F = \frac{n_f - \bar{n}_F^{rand}}{\sigma_{n_F}^{rand}} \tag{2}$$

where n_F is the number of times the subgraph $F_{n,l}$ appears in G and $\bar{n}_F^{rand}$ and $\sigma_{n_F}^{rand}$ are the mean and the standard deviation, respectively, of the number of occurences in an ensemble of graphs obtained by randomising G.

The absolute value of Z_F indicates the distance between observed number of occurences and expected number of occurences in a corresponding ensemble of randomized networks with N and K equal to those of our network G, in units of standard deviations. We are interested in values of Z_F larger than 2, meaning that the number of occurences is two standard deviations larger than $\bar{n}_F^{rand}$, which is a good indicator that the observed number of occurences are due to fluctuations only in 2.14%. Even better, when $Z_F = 3$, the percentage drops to 0.13%, so a very good indication that $F_{n,l}$ is a network motif.

4.3.2 Motifs detection using NemoSuite

NemoSuite [18] (Network Motif Analysis in a Suite) is a web program developed and hosted online by researchers at University of Washington Bothell CSSE to detect and analyze network motifs. A network motif is a frequent and unique subgraph pattern in an input network, and it is determined by P-value beings smaller than 0.05 or Z-score being larger than 2.

Network motif detection methods can be categorized into *network-centric* and *motif-centric* methods:

- **Nemo**: *Network-centric* approach searches all possible non-isomorphic subgraph patterns with a size (typically 3 to 8) and determines a pattern as a motif if the frequency of it is relatively high compared in a corresponding random graph. The network-centric motif detection program improves an ESU (Enumerate SUbgraphs) algorithm introduced by Sebastian Wernicke [19] and has more functionality and output formats (NemoCount, NemoProfile, and NemoCollect). We are interested in these for the instance collection capability.

- **NemoMapPy**: *Motif-centric* approach determines whether the given subgraph pattern is network motif by providing its frequency in an input network. Motif-centric method can determine large size of network motifs (typically larger than 8) by focusing on searching the given pattern. The program implements the NemoMap [20] algorithm developed by the same group and ported to python, which is based on Grochow and Kellis algorithm [21] and MODA program [22].

After creating the seismic network as described in 4.1, for different magnitude restrictions, we used the NemoSuite web application to identify and collect the 3 and 4 nodes motifs (triangles and tetrahedrons) in the network.

Having the motif nodes, our goal is to plot the distribution of the surfaces (for the 3 nodes motifs) and volumes (for the 4 nodes motifs) weighted by the mean and total energy released by the earthquakes in the cubes that form each motif.

4.3.3 Triangle Surfaces

For the 3 nodes motifs, calculations proceeds as follows:

- Firstly we need to calculate the mean energy in each motif. The NemoSuite application [18] extracts the node names of each motif from our network and using the "earthquakes list" attribute of each node we can identify every earthquake in each motif. Using the earthquakes table we can compute the *total energy* release of the earthquakes in each triangle, and by dividing by the total number of earthquakes in the motif we obtain the *mean energy* released.

- Then we use the motif nodes to calculate the surface of each motif. By having the geographical coordinates of each node, we use python's geopy library to calculate the exact "flat" distance along geodesic between two points using latitude and longitude in kilometers, and then we use pythagorean theorem to calculate the distance between the points, including the depth. Example for two points $X(lat, long, depth)$ and $Y(lat2, long2, depth2)$, XY being the side of the respectie triangle:

$$\begin{aligned} &flatDistance = distance(X(lat, long), Y(lat2, long2)).km, \\ &distance().km = \text{geopy distance function in km}, \\ &XY = \sqrt{flatDistance^2 + (X(depth) - Y(depth2))^2}. \end{aligned} \tag{3}$$

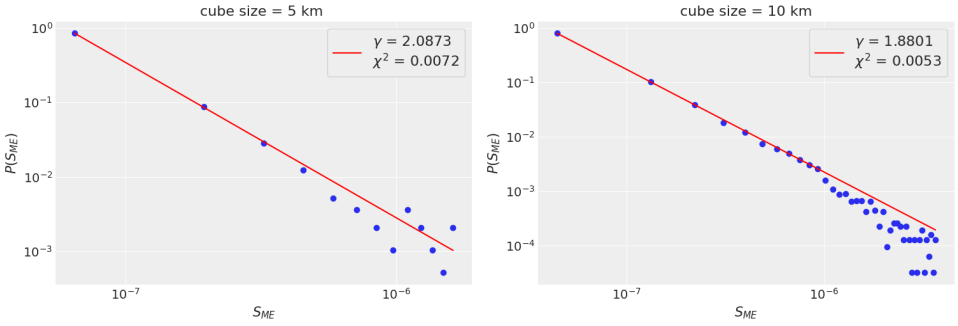
- After calculating all 3 sides of the triangle, we use Heron's formula for triangle surface:

$$\begin{aligned} &S = \sqrt{sp * (sp - a) * (sp - b) * (sp - c)}, \\ &sp = \frac{a + b + c}{2}, \\ &a, b, c = \text{the sides of the triangle.} \end{aligned} \tag{4}$$

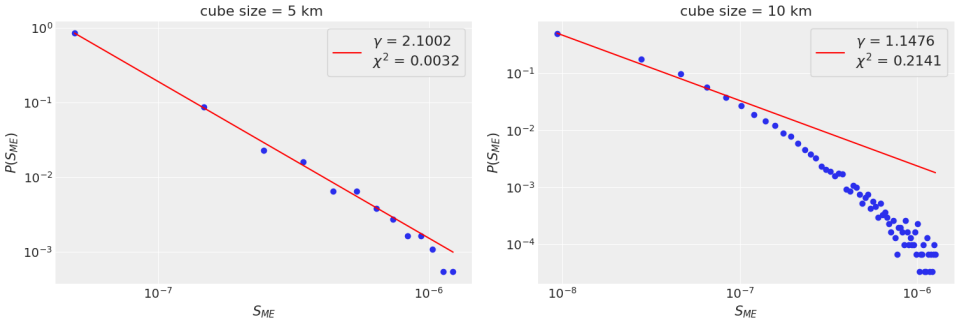
- We use the mean and total energy release per motif and the motif's surface to calculate the distribution of motifs surfaces weighted by mean or total energy.

- Compute the regression of the distribution in log-log space.

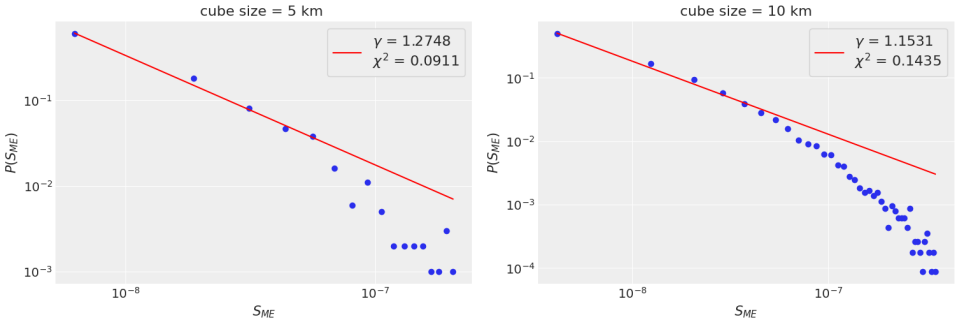
The computations are made for various seismic networks (Vrancea(Romania), California(USA) and Italy), with different magnitude restrictions, for 2 cube sizes: $5 \times 5 \times 5$ km and $10 \times 10 \times 10$ km.



(a) $1 \leq magnitude$



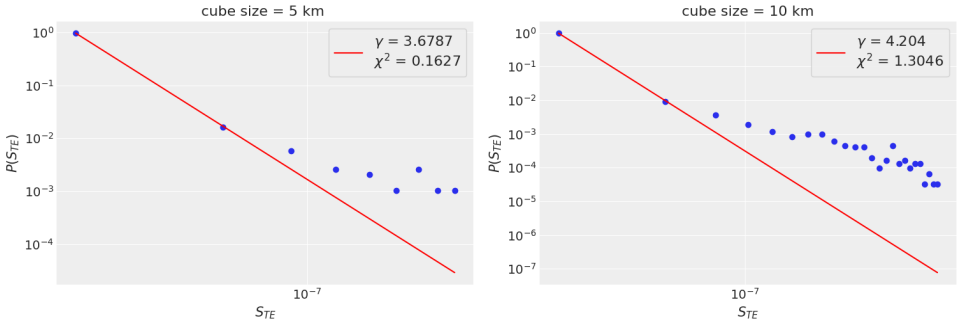
(b) $2 \leq magnitude$



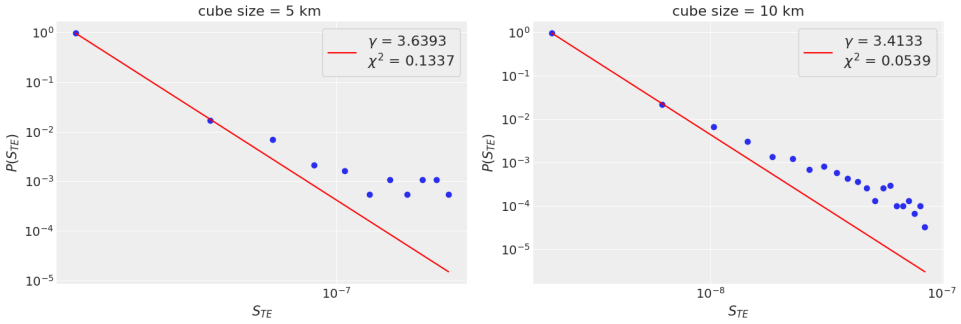
(c) $3 \leq magnitude$

Figure 4.10: $S_{ME} = Surface/Mean\ Energy$ distribution in log-log plots for triangle motifs in Vrancea for 3 magnitude restrictions. The resulting interpolation shows that the distribution appears scale-free with γ ranging from ~ 1.14 to 2.1

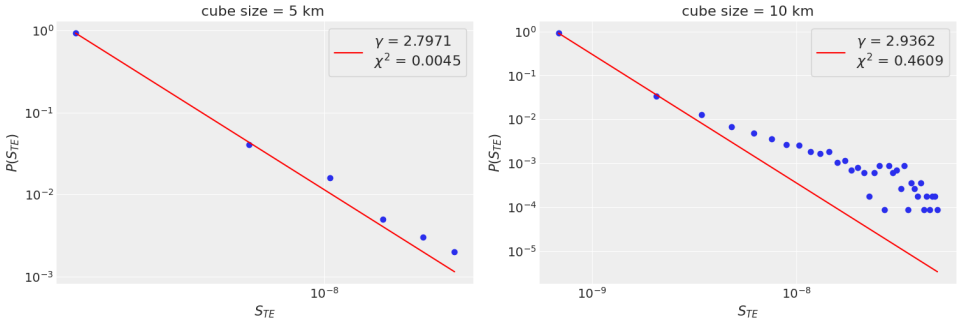
Vrancea - Total Energy Weighted Surfaces



(a) $1 \leq magnitude$



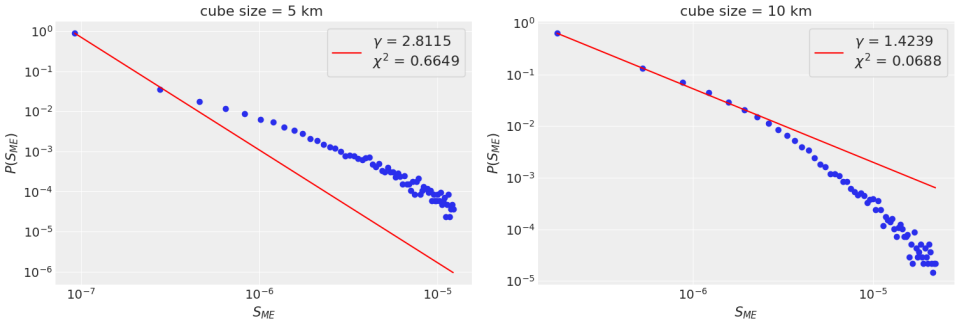
(b) $2 \leq magnitude$



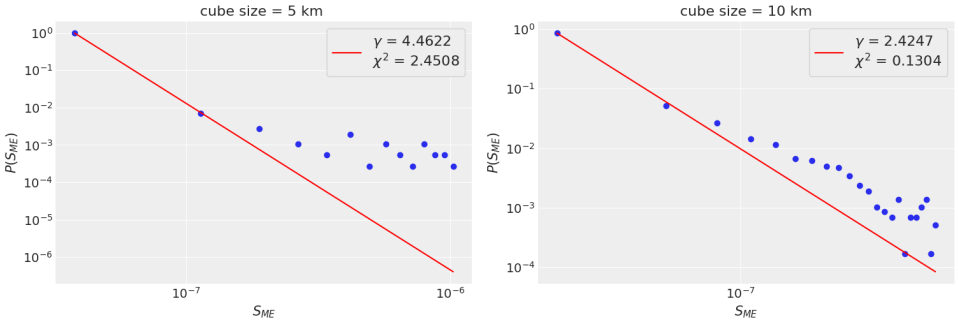
(c) $3 \leq magnitude$

Figure 4.11: $S_{TE} = Surface/Total\ Energy$ distribution in log-log plots for triangle motifs in Vrancea for 3 magnitude restrictions. The resulting interpolation shows that the distribution appears scale-free with γ ranging from ~ 2.79 to 3.67

California - Mean Energy Weighted Surfaces



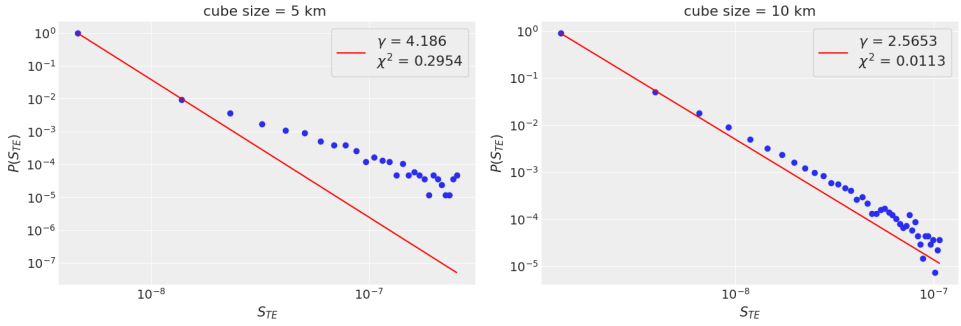
(a) $2 \leq magnitude$



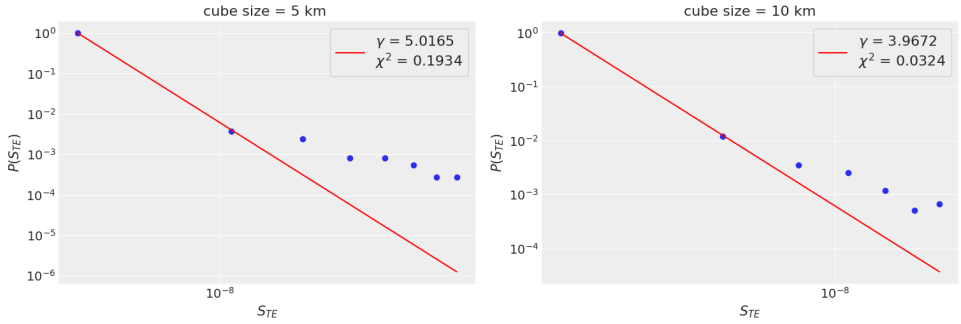
(b) $3 \leq magnitude$

Figure 4.12: $S_{ME} = Surface/Mean\ Energy$ distribution in log-log plots for triangle motifs in California for 2 magnitude restrictions. The resulting interpolation shows that the distribution appears scale-free with γ ranging from ~ 1.42 to 4.46

California - Total Energy Weighted Surfaces



(a) $2 \leq magnitude$



(b) $3 \leq magnitude$

Figure 4.13: $S_{TE} = Surface/Total\ Energy$ distribution in log-log plots for triangle motifs in California for 2 magnitude restrictions. The resulting interpolation shows that the distribution appears scale-free with γ ranging from ~ 2.56 to 5.01

Italy - Mean Energy Weighted Surfaces

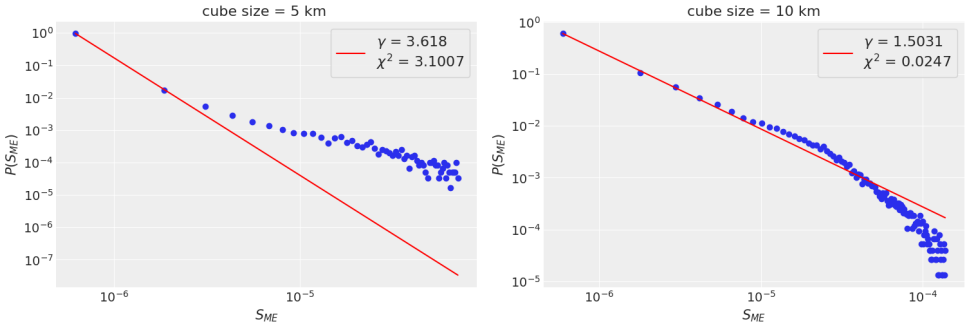


Figure 4.14: $2 \leq magnitude$

Figure 4.15: $S_{ME} = Surface/Mean\ Energy$ distribution in log-log plots for triangle motifs in Italy for $2 \leq magnitude$. The resulting interpolation shows that the distribution appears scale-free better at 10 km cube side granularization, with $\gamma = 1.503$

Italy - Total Energy Weighted Surfaces

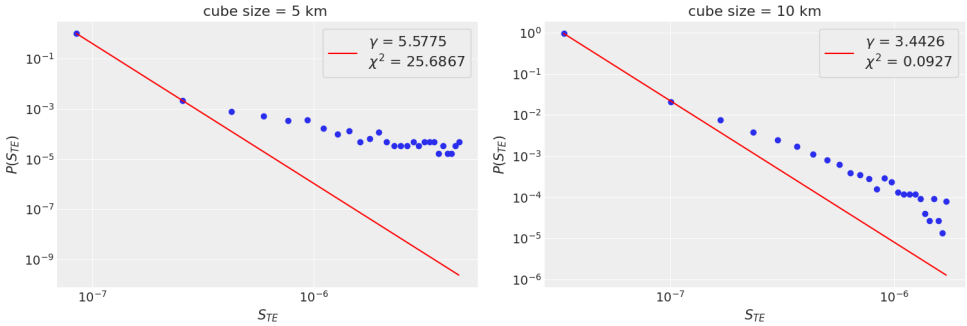


Figure 4.16: $S_{TE} = Surface/Total\ Energy$ distribution in log-log plots for triangle motifs in Italy for $2 \leq magnitude$. The resulting interpolation shows that the distribution appears scale-free better at 10 km cube side granularization, with $\gamma = 3.44$

4.3.4 Tetrahedrons Volumes

For the 4 nodes motifs, calculations proceeds as follows:

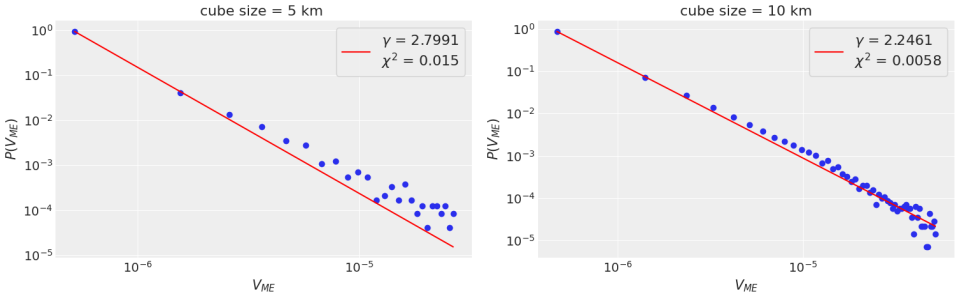
- The first two steps, calculation of mean energy per motif and all the sides of the tetrahedron are similar with the triangles
- For the volume, we use the Heron-like formula [23] for tetrahedrons:

$$
\begin{aligned}
&A = (w - U + v) * (U + v + w), a = (U - v + w) * (v - w + U) \\
&B = (u - V + w) * (V + w + u), b = (V - w + u) * (w - u + V) \\
&C = (v - W + u) * (W + u + v), c = (W - u + v) * (u - v + W) \\
&W, V, U, u, v, w = \text{the sides of the tetrahedron} \\
&p = \sqrt{a * B * C}, q = \sqrt{b * C * A}, r = \sqrt{c * A * B}, s = \sqrt{a * b * c} \\
&V = \frac{\sqrt{(-p + q + r + s) * (p - q + r + s) * (p + q - r + s) * (p + q + r - s)}}{192 * u * v * w}
\end{aligned}
\tag{5}
$$

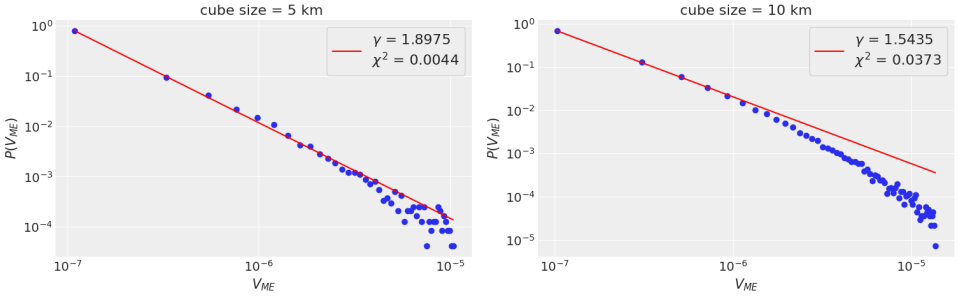
- We use the mean and total energy release per motif and the motif's volume to calculate the distribution of motifs volumes weighted by mean or total energy.
- Compute the regression of the distribution in log-log space.

The computations are made for various seismic networks (Vrancea(Romania), California(USA) and Italy), with different magnitude restrictions, for 2 cube sizes: $5 \times 5 \times 5$ km and $10 \times 10 \times 10$ km.

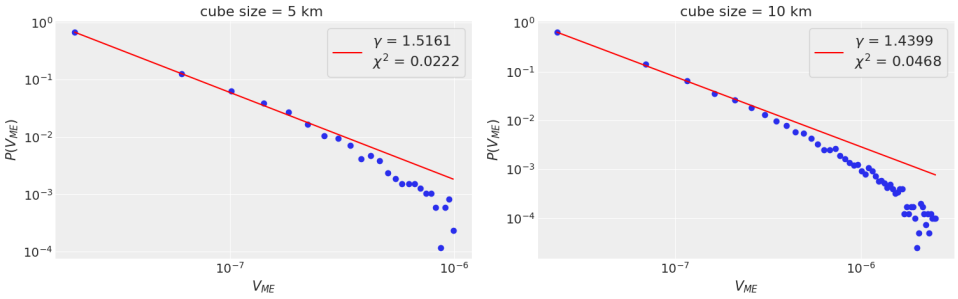
Vrancea Mean Energy Weighted Volumes



(a) $1 \leq magnitude$



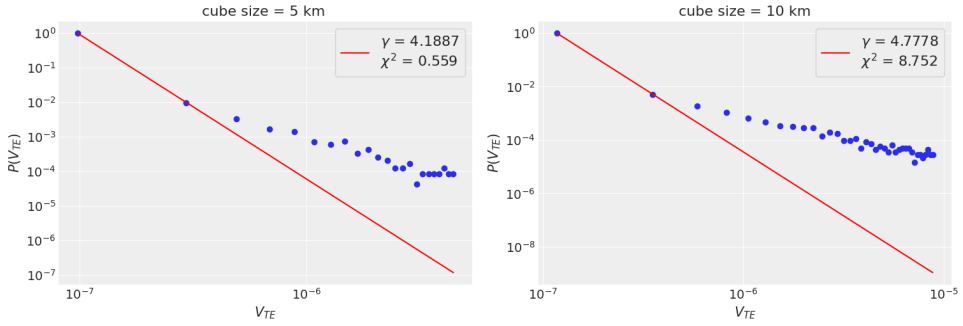
(b) $2 \leq magnitude$



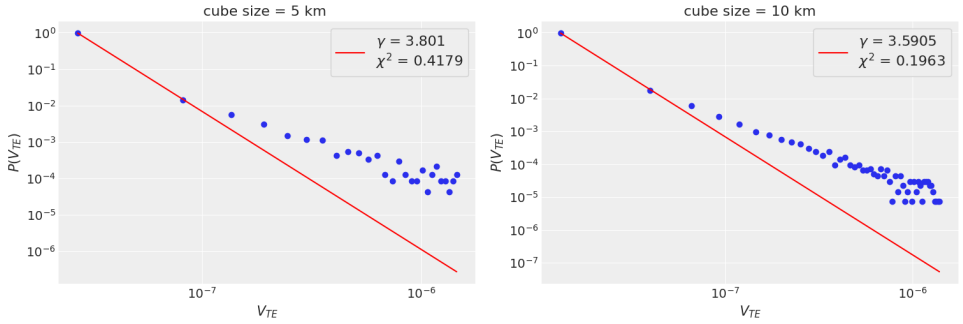
(c) $3 \leq magnitude$

Figure 4.17: $V_{ME} = Volume/Mean\ Energy$ distribution in log-log plots for tetrahedron motifs in Vrancea for 3 magnitude restrictions. The resulting interpolation shows that the distribution appears scale-free with γ ranging from ~ 1.44 to 2.8

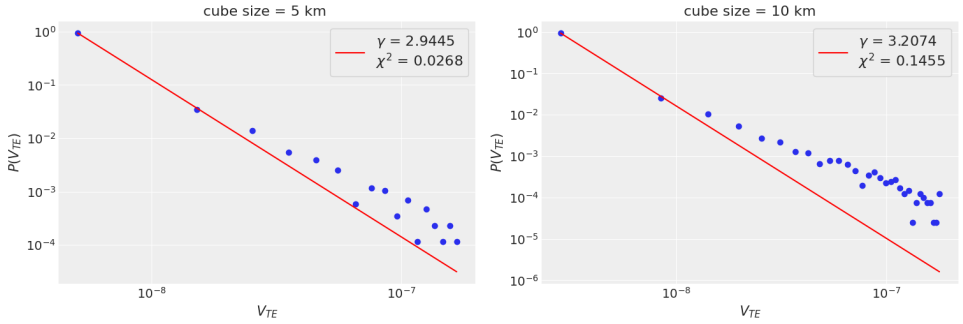
Vrancea Total Energy Weighted Volumes



(a) $1 \leq magnitude$



(b) $2 \leq magnitude$



(c) $3 \leq magnitude$

Figure 4.18: $V_{TE} = Volume/Total\ Energy$ distribution in log-log plots for tetrahedron motifs in Vrancea for 3 magnitude restrictions. The resulting interpolation shows that the distribution appears scale-free with γ ranging from ~ 2.94 to 4.77

California Mean Energy Weighted Volumes

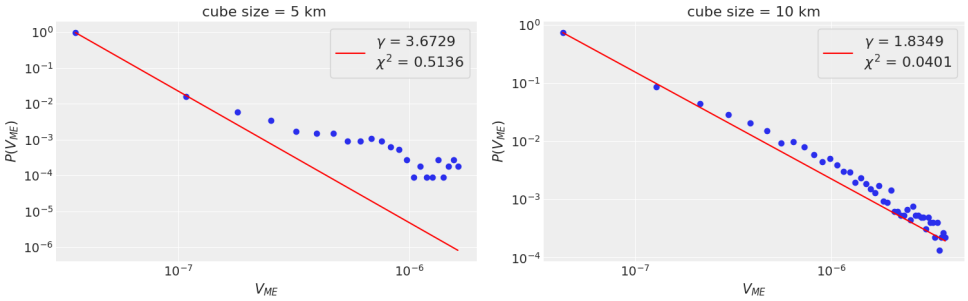


Figure 4.19: $V_{ME} = Volume/Mean\ Energy$ distribution in log-log plots for tetrahedron motifs in California for $3 \leq magnitude$. The resulting interpolation shows that the distribution appears scale-free better at 10 km cube side granularization, with $\gamma = 1.835$

California Total Energy Weighted Volumes

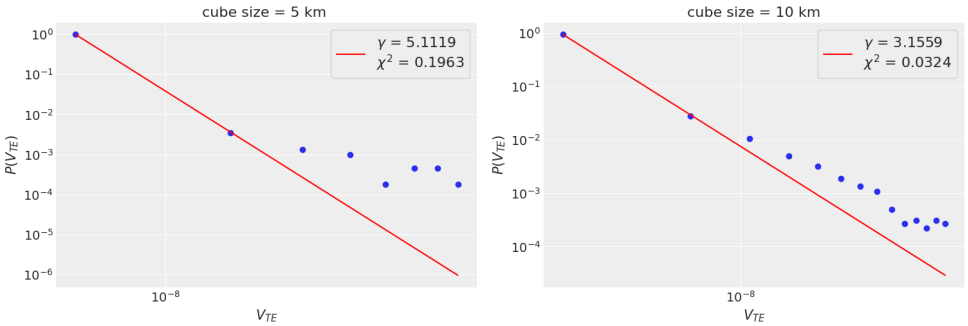


Figure 4.20: $V_{TE} = Volume/Total\ Energy$ distribution in log-log plots for tetrahedron motifs in California for $3 \leq magnitude$. The resulting interpolation shows that the distribution appears scale-free better at 10 km cube side granularization, with $\gamma = 3.156$

Italy Mean Energy Weighted Volumes

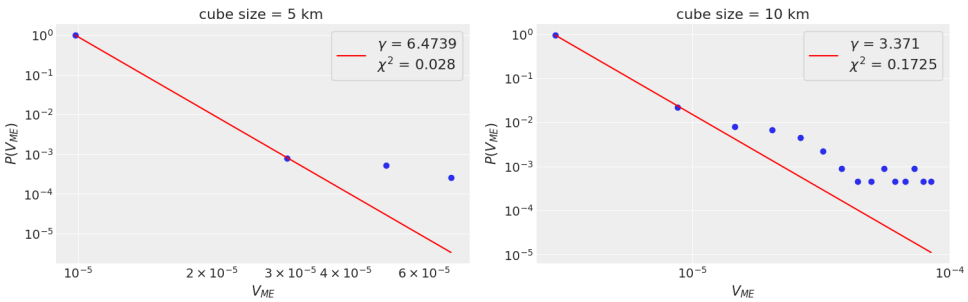


Figure 4.21: $V_{ME} = Volume/Mean\ Energy$ distribution in log-log plots for tetrahedron motifs in Italy for $3 \leq magnitude$. The resulting interpolation shows that the distribution appears scale-free better at 10 km cube side granularization, with $\gamma = 3.371$

4.4 Motifs Visualization

As a visualization tool for the earthquakes networks and their motifs we used the application ParaView, which works together with the python library "VTK".

The Visualization ToolKit (VTK) is an open source software system for 3D graphics, visualization and image processing. VTK supports a variety of visualization algorithms : scalar, vector, tensor, texture and volumetric methods, making it easy to translate graphs into geometric form.

ParaView [24] is an open-source application designed to visualize data of varying sizes, from small to very large. Under the hood, ParaView uses VTK as the data processing and rendering engine.

In python, we created our graphs using the library "networkx" for it's easy and intuitive graph creation method, and easy to access node attributes using python dictionaries. The translation from networkx to vtk is fairly straight-forward: the nodes are translated as 3D Glyphs, the edges as 3D Tubes and the motifs as 3D Surfaces, using coordinates between 0 and 1 for each dimension (latitude, longitude and depth) for the vertices:

- Firstly, we properly scale the coordinates of each node to fit in the [0,1] range of dimensions (needed in Paraview). Each node is represented by a geometric "Point" in vtk and each link or motif is represented by a "Cell" delimited by the proper points.
- We are also able to add scalar attributes to these geometric elements:
  - for nodes: the degree of connectivity of each node;
  - for edges : the weight of the edge and the color "quality" of the edge (0 if it's a normal edge, 1 if it's part of a motif).
- The size of each geometric element can be adjusted according to the scalar attributes (for example, nodes with higher degree appear larger, links with larger wight appear thicker).
- The color of each geometric element can be adjusted according to the scalar attributes (for example, a gradient is applied to nodes according to their degree).
- Also the opacity of each element can be adjusted (even logarithmically, for example lower degree nodes can be less visible than higher degree nodes).

For visualization purposes we have chosen a smaller size network in order to properly see the geometric elements in Paraview. So, we have restricted the magnitude scale to only the events with magnitude larger than 4 (and larger than 5 for Japan).

Romania Real Network Visualization

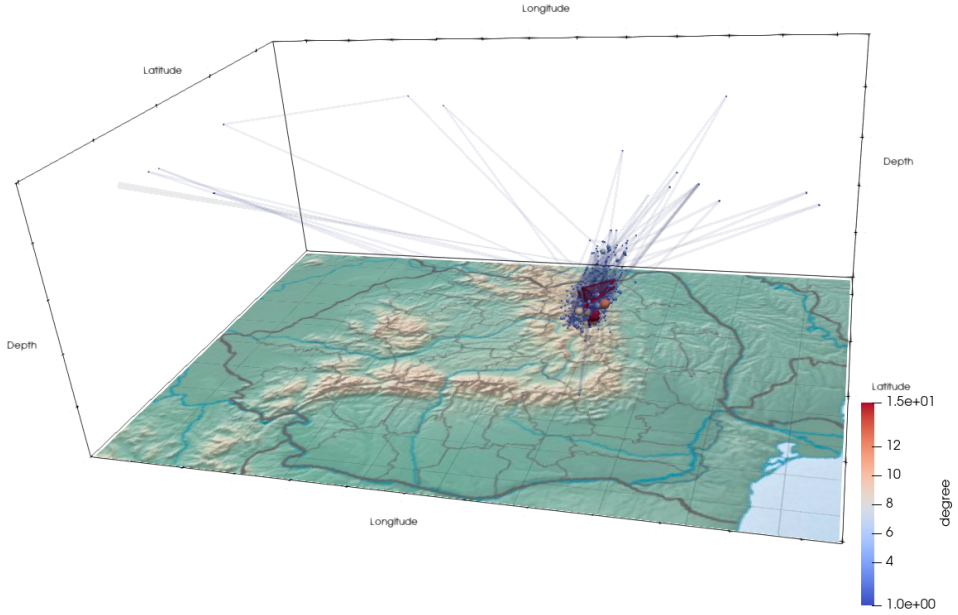
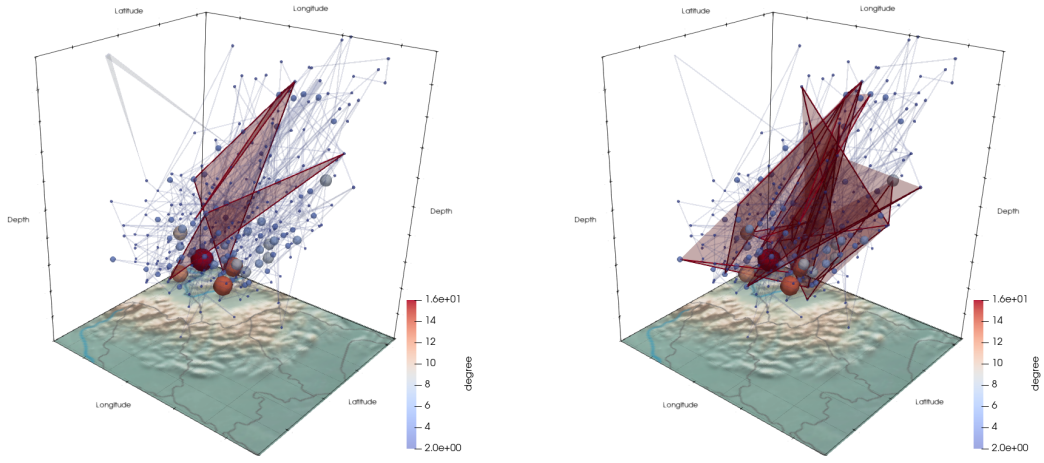


Figure 4.22: Motifs in Romania Seismic Network, earthquakes with magnitude > 4. It is obviously noticed that most of the network is concentrated in the Vrancea seismic zone.

We can isolate only the Vrancea seismic region because that is the main area of seismicity in Romania. The coordinate restrictions are mentioned in chapter 3. Thus we obtain a graph with 255 nodes, 375 edges, 3 triangles and 10 tetrahedrons.



(a) Triangles in Vrancea Seismic Network

(b) Tetrahedrons in Vrancea Seismic Network

Figure 4.23: Motifs in Vrancea Seismic Network - 380 events stored in 255 nodes and connected through 375 edges. The degree of connectivity of the nodes ranges from 1 to 16. The motifs are represented as surfaces, for the triangles, or by volumes for the tetrahedrons (drawn in red)

California Real Network Visualization

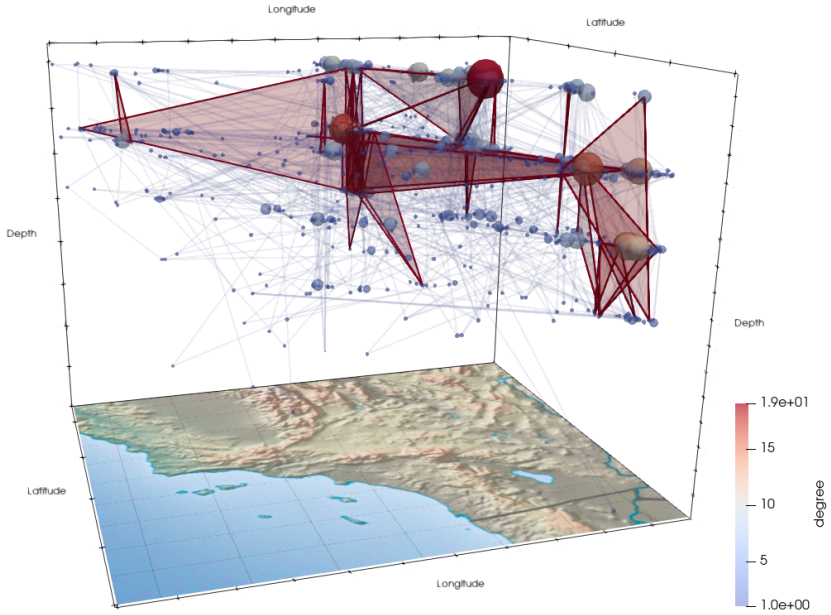


Figure 4.24: Motifs in California Seismic Network, earthquakes with magnitude > 4 : 1119 events are stored in 718 nodes, which are connected by 1036 edges. The degree of connectivity of the nodes ranges from 1 to 19. With red surfaces the triangles of this zone are presented.

Italy Real Network Visualization

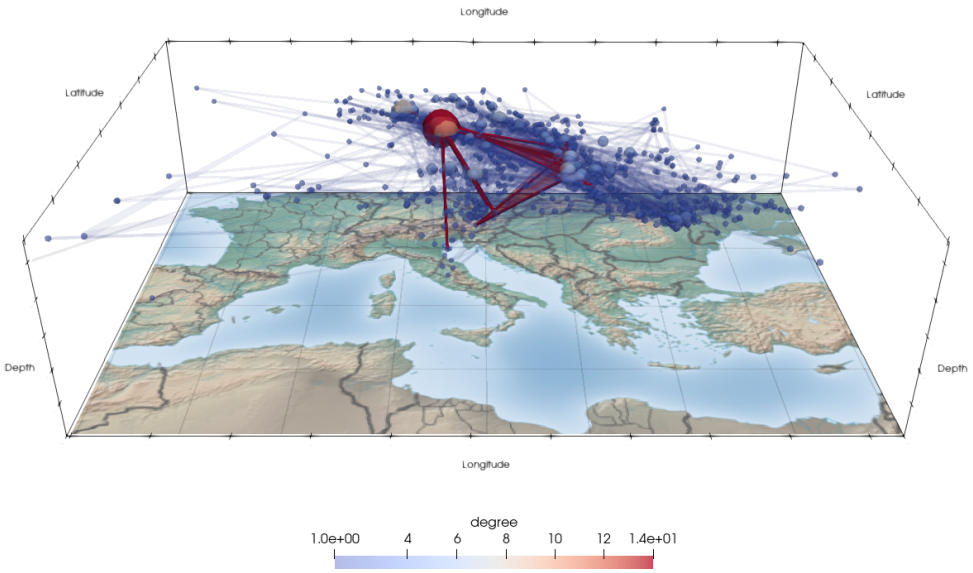


Figure 4.25: Motifs in Italy Seismic Network, earthquakes with magnitude > 4 : 1490 events are stored in 1325 nodes, which are connected by 1452 edges. The degree of connectivity of the nodes ranges from 1 to 16. With red surfaces the tetrahedrons of this zone are presented.

Japan Real Network Visualization

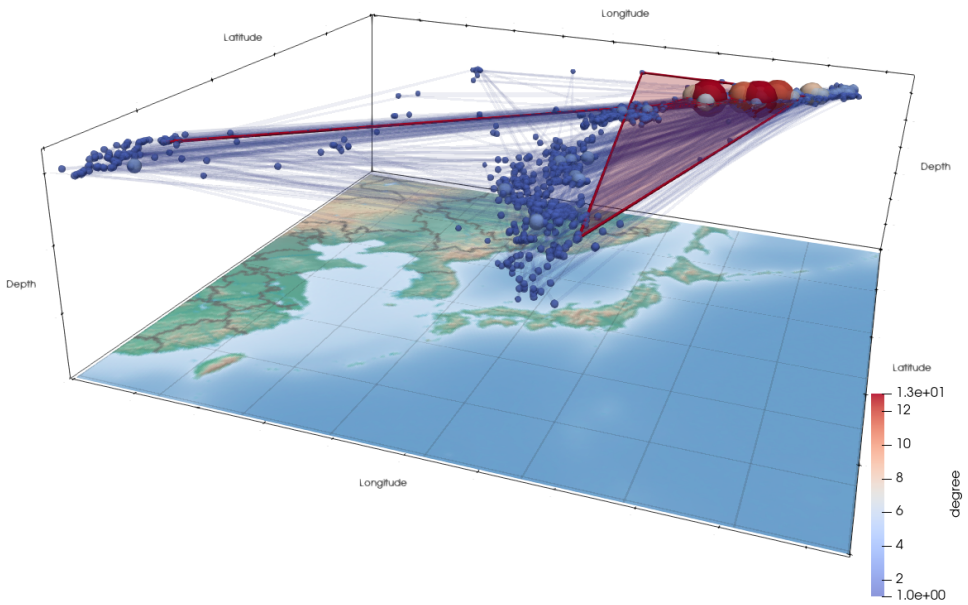


Figure 4.26: Motifs in Japan Seismic Network, earthquakes with magnitude > 5 : 16948 events are stored in 14396 nodes, which are connected by 20332 edges. The degree of connectivity of the nodes ranges from 1 to 13. With red surfaces the triangles of this zone are presented.

Chapter 5

Seismic Spatio-Temporal Autocorrelations

5.1 Correlations and Autocorrelations

Correlations represent a measure of how one value or system responds to another. There are many different types of correlation functions which can be used to determine the correlation of two random variables or systems. For example, time correlation functions are used in the theory of noise and stochastic processes in statistical physics and spectroscopy.

In signal processing, the term cross-correlation represents a type of correlation function, which is a generalized form of regular linear correlation. It is used to compare different time series, allowing to see how two signals match and where the best matching occurs, so hidden patterns in a signal may be revealed.

We introduced the term cross-correlation in order to understand autcorroleation, which is what we are interested in our work. The major difference between the two is that while cross-correlation is used when two **different** sequences are correlated, autocorrelation means that the correlation occurs between two of the **same** sequences, i.e. you correlate the signal with itself.

5.2 Spatial Autocorrelations

"Everything is related to everything else, but near things are more related than distant things." - Waldo R. Tobler, the first law of geography.

Spatial autocorrelation helps understand the degree to which one object is similar to other nearby objects. One of the most popular tests of spatial autocorrelation is the Moran's I test [25]. In the following we present how we applied this function and the results. The objects of our function represent the total energy released by the earthquakes in each of the cubes that we split our seismic region in.

After creating the earthquake network we can establish the spatial correlations between nodes. First we compute the total energy release of the earthquakes in each node. So each cube now has the total energy E_{ijk} associated with it.

Next we compute the mean energy in the network:

$$< E >= \frac{1}{N * M * L} \sum_{i=1}^{N} \sum_{j=1}^{M} \sum_{k=1}^{L} E_{ijk}, \tag{1}$$

N = Number of cubes in the latitude dimension,

M = Number of cubes in the longitude dimension,

L = Number of cubes in the depth dimension.

Then we can compute the variance that represents the distance from the central value:

$$var(E) = \frac{1}{N * M * L} \sum_{i=1}^{N} \sum_{j=1}^{M} \sum_{k=1}^{L} (E_{ijk} - < E >)^2. \quad (2)$$

The standard deviation can thus be defined as:

$$\sigma_E = \sqrt{var(E)}. \quad (3)$$

Then we can introduce the covariance:

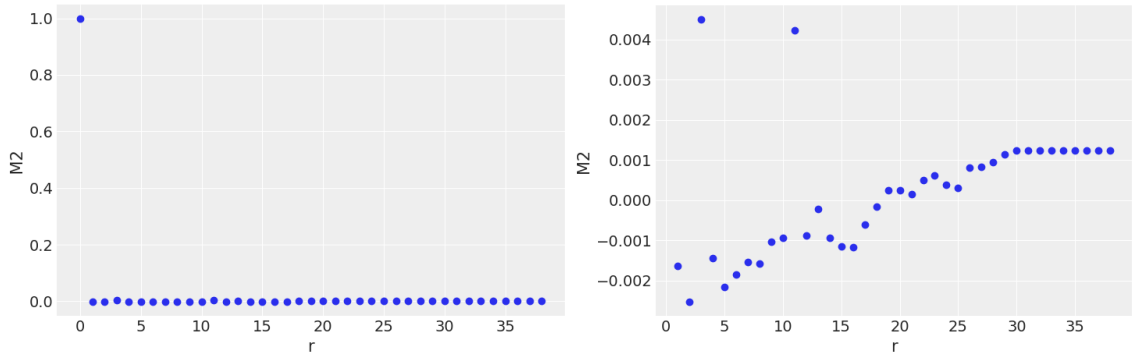
$$cov(E, r) = \frac{1}{N * M * L} \sum_{i=1}^{N} \sum_{j=1}^{M} \sum_{k=1}^{L} \sum_{l=1}^{N} \sum_{m=1}^{M} \sum_{n=1}^{L} (E_{ijk} - < E >)(E_{lmn} - < E >). \quad (4)$$

where the cube with total energy E_{lmn} is situated at distance r from the cube with total energy E_{ijk}.

Lastly, the spatial correlation index, $M2$ is defined as the ratio of the covariance to the standard deviation squared σ_E^2:

$$M2(E, r) = \frac{cov(E, r)}{\sigma_E^2}. \quad (5)$$

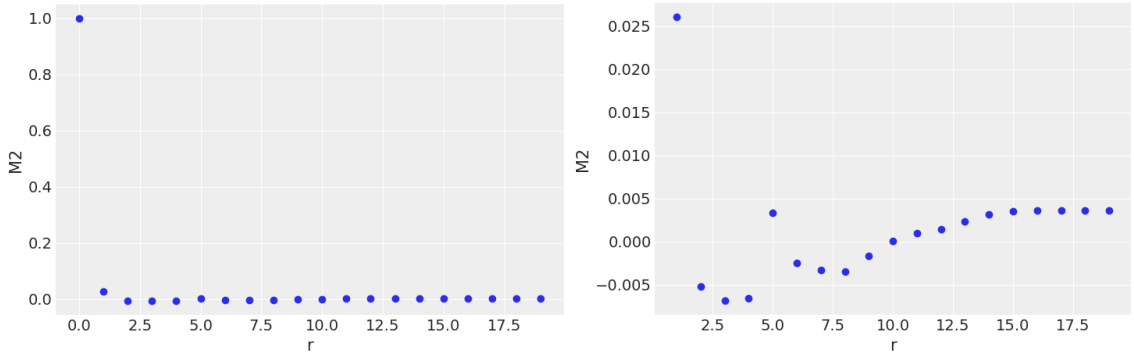
Vrancea Spatial autocorrleation function $M2(r)$ for Vrancea networks - 3 different cube sizes $5 \times 5 \times 5$ km, $10 \times 10 \times 10$ km and $20 \times 20 \times 20$ km. Each function is presented twice: the left plot is represented with the point in $M2(0)$ and on the right, without that point in order to better see the feature of the plot.



(a) With M2(0)

(b) Without M2(0)

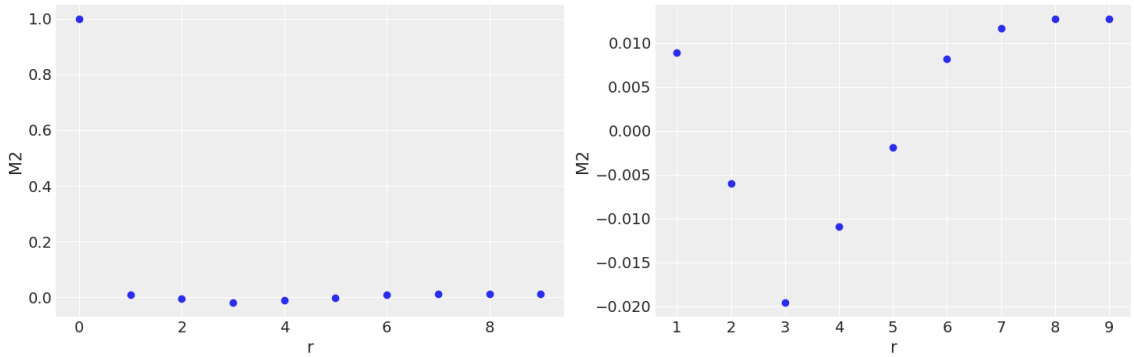
Figure 5.1: Spatial autocorrelation function $M2(r)$ for Vrancea seismic zone split into $5 \times 5 \times 5$ km cubes. r represents the distance in multiples of 5 km. The right hand plot, without the point $M2(0)$ better displays the feature of the plot



(a) With M2(0)

(b) Without M2(0)

Figure 5.2: Spatial autocorrelation function $M2(r)$ for Vrancea seismic zone split into $10 \times 10 \times 10$ km cubes. r represents the distance in multiples of 5 km. The right hand plot, without the point $M2(0)$ better displays the feature of the plot



(a) With M2(0)

(b) Without M2(0)

Figure 5.3: Spatial autocorrelation function $M2(r)$ for Vrancea seismic zone split into $20 \times 20 \times 20$ km cubes. r represents the distance in multiples of 5 km. The right hand plot, without the point $M2(0)$ better displays the feature of the plot

5.3 Temporal Autocorrelations

In time series analysis, autocorrelation is used to correlate observations at a time step with observations at previous time steps, called *lags*.

In general, considering a time series $y_1, ..., y_n$, it's mean is:

$$\bar{y} = \frac{1}{n} \sum_{i=1}^{n} y_i. \tag{6}$$

The autocovariance function at lag k, for $k \geq 0$, of the time series is defined by:

$$s_k = \frac{1}{n} \sum_{i=1}^{n-k} (y_i - \bar{y})(y_{i+k} - \bar{y}). \tag{7}$$

Finally, the autocorrelation function (ACF) at lag k of the time series is:

$$r_k = \frac{s_k}{s_0}. \tag{8}$$

For our networks, we need to establish a way of defining the lags, and the quantity that is to be autocorrelated. These calculations are made as such:

- Establish the time period of the network, from the first earthquake of the database to the last. For example, as mentioned in Chapter 3 the timeframe we have for our whole network in Vrancea(Romania) is 19:03:01 August 19 1976 to 00:11:55 February 28 2021, so we would have a $timeframe$ of 16263 days and 05:08:54 hours.

- Next we iterate through the table and find the biggest interval dt between two consecutive earthquakes.

- Then, by dividing the total $timeframe$ by that interval dt we create a certain amount of time windows W, so we have split the total timeframe in equal parts and surely each part contains at least one earthquake.

- Place each earthquake in it's respective time window and calculate the total energy release for each time window. Now we have E_i total energies (the values of our time series y_i) with $i = 1, 2, ...W$ (the lags).

Having defined our earthquake energies time series, we can now proceed to calculate the temporal autocorrelation function by adapting the formula described earlier:

Firstly, compute the mean energy, by summing over all time windows W:

$$\bar{E} = \frac{1}{W} \sum_{i=1}^{W} E_i. \tag{9}$$

Then you can compute the autocovariance function at lag k, for $k >= 0$, of the time series as:

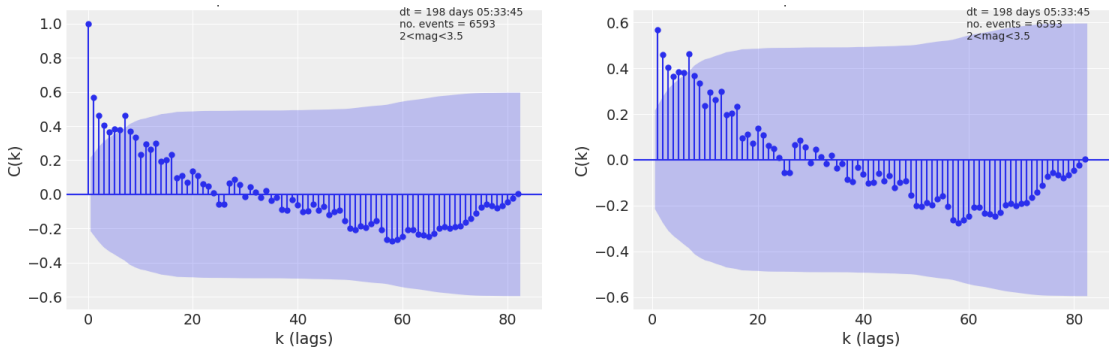
$$s_k = \frac{1}{W-k} \sum_{i=1}^{W-k} (E_i - \bar{E})(E_{i+k} - \bar{E}). \tag{10}$$

Finally, the autocorrelation function is defined as:

$$C(k) = \frac{s_k}{s_0}. \tag{11}$$

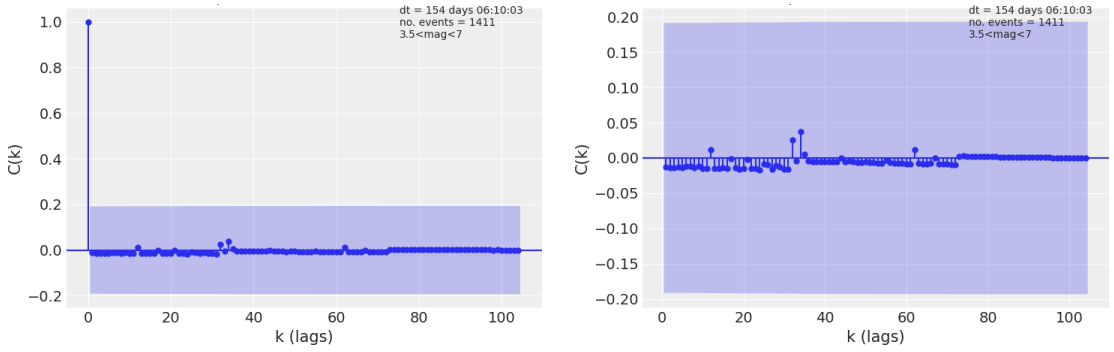
We realised Temporal AutoCorrelation Functions (TACFs) of the earthquake energies time series on several seismic networks, for two magnitude windows: below and above 3.5. Below are the results, with and without the point $C(0)$ in order to better recognize the feature of the plot (the point $C(0)$ is always equal to 1).

Vrancea TACF Vrancea(Romania) Seismic Network



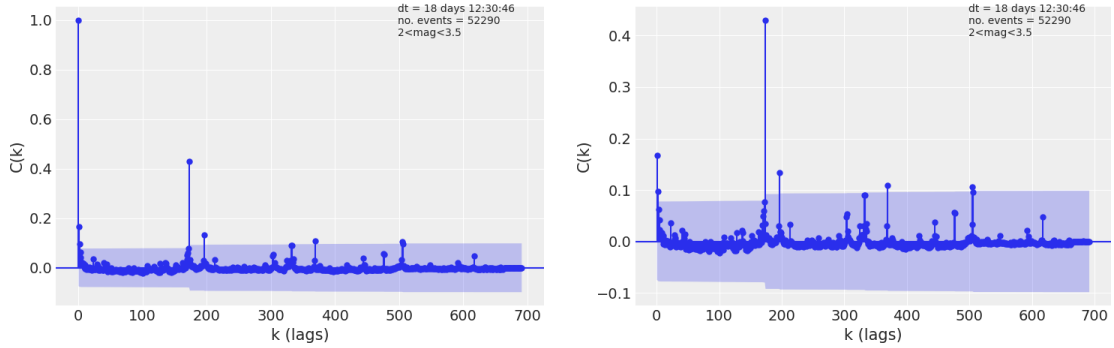
(a) With C(0) (b) Without C(0)

Figure 5.4: Temporal Autocorrelation Function for Vrancea(Romania) for magnitude restrictions: $2 < mag < 3.5$. A $timeframe$ of 16263 days 05 : 08 : 54 hours, containing a total of 6593 number of events, is split into $W = 83$ equal windows of $dt = 198$ days 05 : 33 : 45 hours.



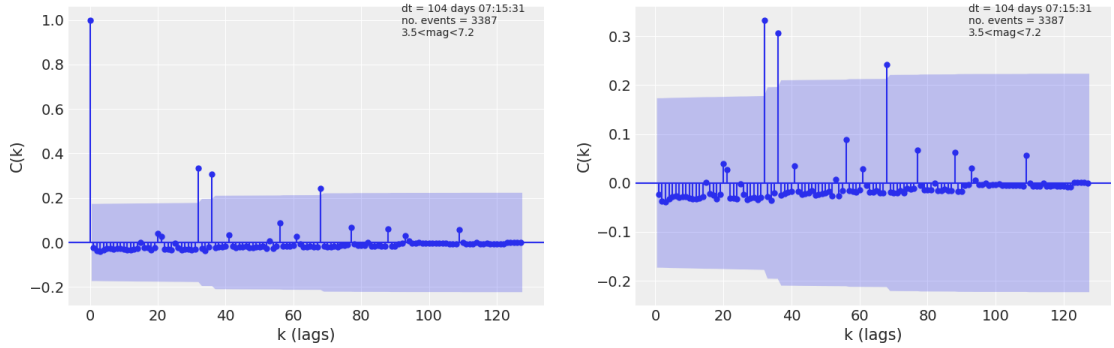
(a) With C(0) (b) Without C(0)

Figure 5.5: Temporal Autocorrelation Function for Vrancea(Romania) for magnitude restrictions: $3.5 < mag < 7$. A $timeframe$ of 16263 days 05 : 08 : 54 hours, containing a total of 1411 number of events, is split into $W = 105$ equal windows of $dt = 154$ days 06 : 10 : 03 hours.



(a) With C(0) (b) Without C(0)

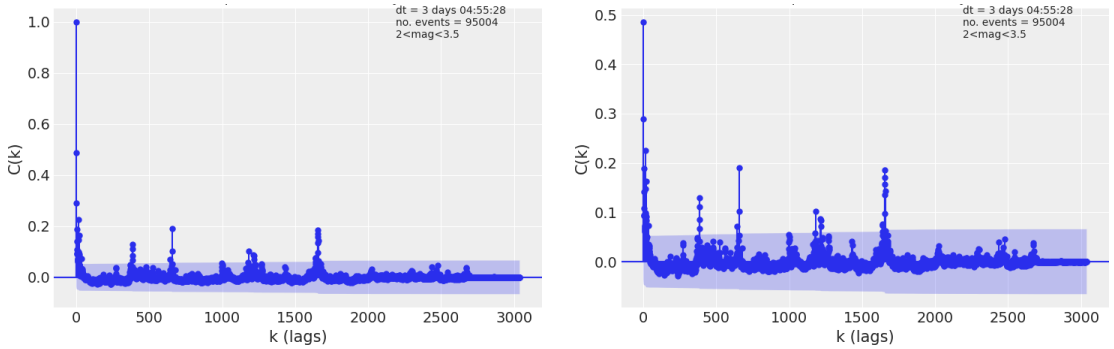
Figure 5.6: Temporal Autocorrelation Function for California(USA) for magnitude restrictions: $2 < mag < 3.5$. A $timeframe$ of 13514 days $03:53:33$ hours, containing a total of 52290 events, is split into $W = 692$ equal windows of $dt = 18$ days $12:30:46$ hours.



(a) With C(0) (b) Without C(0)

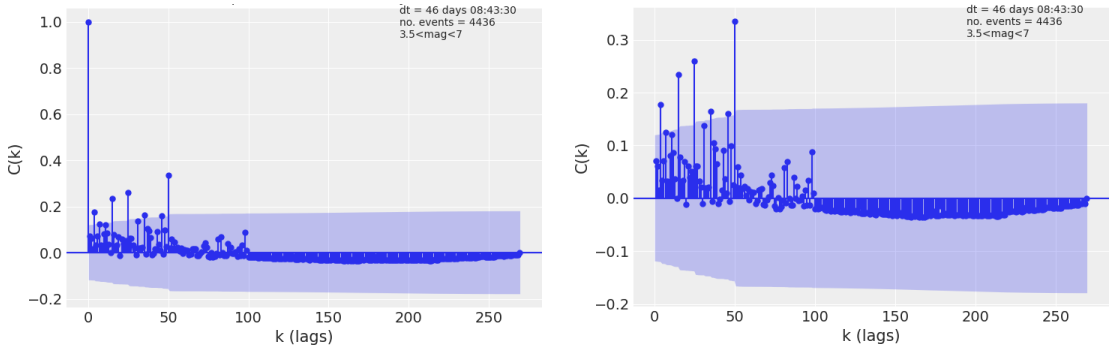
Figure 5.7: Temporal Autocorrelation Function for California(USA) for magnitude restrictions: $3.5 < mag < 7.2$. A $timeframe$ of 13514 days $03:53:33$ hours, containing a total of 3387 events, is split into $W = 128$ equal windows of $dt = 104$ days $07:15:31$ hours.

Italy TACF Italy Seismic Network



(a) With C(0) (b) Without C(0)

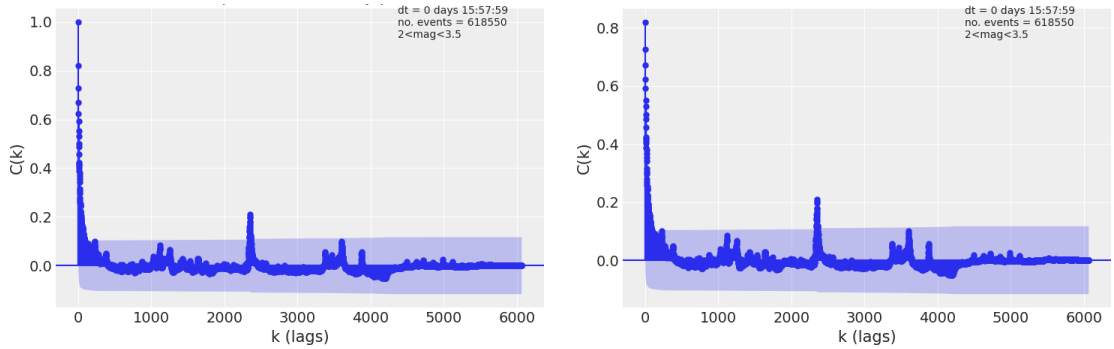
Figure 5.8: Temporal Autocorrelation Function for Italy for magnitude restrictions: $2 < mag < 3.5$. A $timeframe$ of 12783 days 06 : 18 : 25 hours, containing a total of 95004 events, is split into $W = 3040$ equal windows of $dt = 3$ days 04 : 55 : 28 hours.



(a) With C(0) (b) Without C(0)

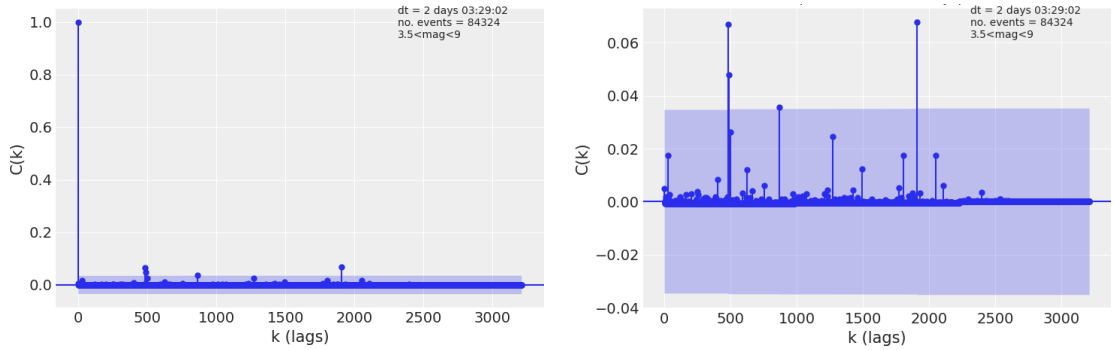
Figure 5.9: Temporal Autocorrelation Function for Italy for magnitude restrictions: $3.5 < mag < 7$. A $timeframe$ of 12783 days 06 : 18 : 25 hours, containing a total of 4436 events, is split into $W = 270$ equal windows of $dt = 46$ days 08 : 43 : 30 hours.

Japan ACF Japan Seismic Network



(a) With C(0) (b) Without C(0)

Figure 5.10: Temporal Autocorrelation Function for Japan for magnitude restrictions: $2 < mag < 3.5$. A $timeframe$ of 10104 days 22 : 53 : 49 hours, containing a total of 618550 events, is split into $W = 6068$ equal windows of $dt = 04 : 55 : 28$ hours.



(a) With C(0) (b) Without C(0)

Figure 5.11: Temporal Autocorrelation Function for Italy for magnitude restrictions: $3.5 < mag < 9$. A $timeframe$ of 10104 days 22 : 53 : 49 hours, containing a total of 84324 events, is split into $W = 3213$ equal windows of $dt = 2$ days $03 : 29 : 02$ hours.

By analyzing the TACFs for earthquakes with $2 < magnitude < 3.5$, the events can be interpreted as fluctuations that exhibit a chaotic behaviour and at larger magnitudes, above 3.5, this behaviour is lost. Here, peaks are recognized at certain lags suggesting a correlation between events at certain time intervals. This can be an indication that large events are more likely than not to be separated by these time frames. For example, in Vrancea we recognize a couple of peaks at around 30 lags, which would translate as ~ 12.65 years. This means that it is possible for some large events to have a 12.65 years span between them.

Conclusions

In conclusion, we have shown how to access and extract information about earthquakes from the databases available. We presented how to construct various seismic networks for regions around the globe and by measures of connectivity and motif discovery we proved that the scale-free nature of these networks is **very robust**.

With the help of visualization tools such as VTK and ParaView, we have shown how these network can be graphically represented in real, geographic coordinate space, with the connectivity, edge weights and motifs pictured with the help of graphs.

We also computed spatial and temporal autocorrelation functions for the regions analyzed and concluded two distinct behaviours when taking into account lower magnitude, in contrast with higher magnitude earthquakes. At lower energies, events can be viewed as some kind of chaotic fluctuations, while at higher energies, some pattern may emerge.

This work sustains the popular notion that earthquakes are complex systems that show self-organization to criticality behaviour.

Further study can be done in this space, with the possibility of analyzing other seismic regions by employing the same computational tools used in this report. Also, from the perspective of network theory, other centrality or community measures may give insight about clusters of earthquakes, opening the possibility of splitting a region into more relevant sections.

Bibliography

[1] Scott E. Page. *Diversity and Complexity*. 1st. USA: Princeton University Press, 2010.

[2] National Research Council. *Condensed-Matter Physics*. Washington, DC: The National Academies Press, 1986.

[3] David Willshaw. "CHAPTER 1 - Self-organization in the Nervous System". In: *Cognitive Systems - Information Processing Meets Brain Science*. Ed. by Richard Morris, Lionel Tarassenko, and Michael Kenward. London: Academic Press, 2006, pp. 5–33. DOI: https://doi.org/10.1016/B978-012088566-4/50004-0.

[4] Henrik Flyvbjerg. "Simplest Possible Self-Organized Critical System". In: *Phys. Rev. Lett.* 76 (6 1996), pp. 940–943. DOI: 10.1103/PhysRevLett.76.940.

[5] Per Bak, Chao Tang, and Kurt Wiesenfeld. "Self-organized criticality: An explanation of the 1/f noise". In: *Phys. Rev. Lett.* 59 (4 1987), pp. 381–384. DOI: 10.1103/PhysRevLett.59.381.

[6] Nicholas W. Watkins et al. "25 Years of Self-organized Criticality: Concepts and Controversies". In: *Space Science Reviews* 198.1-4 (2015), 3–44. DOI: 10.1007/s11214-015-0155-x.

[7] S. Boccaletti et al. "Complex networks: Structure and dynamics". In: *Physics Reports* 424.4 (2006), pp. 175–308. DOI: https://doi.org/10.1016/j.physrep.2005.10.009.

[8] Albert-Laszlo Barabasi. *Linked: How Everything Is Connected to Everything Else and What It Means*. Plume, 2003.

[9] Leonhard Euler. "Solutio problematis ad geometriam situs pertinentis". In: *Commentarii Academiae Scientiarum Imperialis Petropolitanae* 8 (1736), pp. 128–140.

[10] Per Bak. *How Nature Works*. 1st. Copernicus, 1996.

[11] Olami, Feder, and Christensen. "Self-organized criticality in a continuous, non-conservative cellular automaton modeling earthquakes." In: *Physical review letters* 68 8 (1992), pp. 1244–1247.

[12] R. Burridge and L. Knopoff. "Model and theoretical seismicity". In: *Bulletin of the Seismological Society of America* 57.3 (June 1967), pp. 341–371.

[13] *The database of earthquakes in Romania is freely available at.* URL: http://www.infp.ro/.

[14] *The database of earthquakes in California(USA) is freely available at.* URL: https://www.scec.org/.

[15] *The database of earthquakes in Italy is freely available at.* URL: https://www.ingv.it/.

[16] *The database of earthquakes in Japan is freely available at.* URL: https://www.jma.go.jp/jma/index.html.

[17] S Abe and N Suzuki. "Scale-free network of earthquakes". In: *Europhysics Letters (EPL)* 65.4 (2004), pp. 581–586. DOI: 10.1209/epl/i2003-10108-1.

[18] Wooyoung Kim et al. "NemoSuite: Web-based Network Motif Analytic Suite". In: *Advances in Science, Technology and Engineering Systems Journal* 5 (Dec. 2020), pp. 1545–1553. DOI: 10.25046/aj0506185.

[19] Sebastian Wernicke. "Efficient Detection of Network Motifs". In: *IEEE/ACM transactions on computational biology and bioinformatics / IEEE, ACM* 3 (Oct. 2006), pp. 347–59. DOI: 10.1109/TCBB.2006.51.

[20] Tien Huynh, Soma Mbadiwe, and Wooyoung Kim. "NemoMap: Improved Motif-centric Network Motif Discovery Algorithm". In: *Advances in Science, Technology and Engineering Systems Journal* 3 (Sept. 2018), pp. 186–199. DOI: 10.25046/aj030523.

[21] Joshua A. Grochow and Manolis Kellis. "Network Motif Discovery Using Subgraph Enumeration and Symmetry-Breaking". In: *RECOMB*. 2007.

[22] Saeed Omidi, Falk Schreiber, and Ali Masoudi-Nejad. "MODA: An efficient algorithm for network motif discovery in biological networks". English. In: *Genes and Genetic Systems* 84.5 (2009), pp. 385 –395. DOI: 10.1266/ggs.84.385.

[23] Kahan W. "What has the Volume of a Tetrahedron to do with Computer Programming Languages?" In: 1 (), 16–17.

[24] Kenneth Moreland. *The ParaView Tutorial, Version 5.6*. 2018.

[25] P. A. P. MORAN. "NOTES ON CONTINUOUS STOCHASTIC PHENOMENA". In: *Biometrika* 37.1-2 (June 1950), pp. 17–23. DOI: 10.1093/biomet/37.1-2.17.

Appendices

Appendix A

Collecting Databases, Energy Release and Cube Splitting

```
import pandas as pd
import math

def sqlCollect(condition,side,region,withCubes=True,
  enRel=False):
 mydb = mysql.connector.connect(
 host="",
 user="",
 password="",
 database="",
 auth_plugin='mysql_native_password'
 )

 mycursor = mydb.cursor()

 # Call the database with the set conditions
 mycursor.execute(condition)

 # Append the database to the result object array
 result = mycursor.fetchall()

 # Initialize our lists
 date= []
 lat = []
 long = []
 dep = []
 mag = []

 # Append the result of sql into our lists
 for i in range(len(result)):
  date.append(result[i][0])
  lat.append(result[i][1])
  long.append(result[i][2])
  dep.append(result[i][3])
  mag.append(result[i][4])
```

```
# Force the results into float type
lat = [float(item) for item in lat]
long = [float(item) for item in long]
dep = [float(item) for item in dep]
mag = [float(item) for item in mag]

# If withCubes TRUE => collect database and
#create cube indexes
if withCubes:
 # Create cubes indexes for all the dimensions
 #-----------------------------------------

 # Set min max for the dimensions
 minLat = min(lat)
 maxLat = max(lat)
 minLong = min(long)
 maxLong = max(long)
 mindep = min(dep)
 maxdep = max(dep)


 # Find the no. cubes, given the size of the cube in km
 # Depending on region, one degree lat/long differs:

 # IN VRANCEA 1 deg Lat = 111km / 1 deg Long = 79km
 if region=='Vrancea' or region=='Romania':
  longInKm = 79

 # IN CALI 1 deg Lat = 111km / 1 deg Long = 94km
 if region=='California':
  longInKm = 94

 # IN ITALY 1 deg Lat = 111km / 1 deg Long = 84km
 if region=='Italy':
  longInKm = 84

 # IN JAPAN 1 deg Lat = 111km / 1 deg Long = 91km
 if region=='Japan':
  longInKm = 91

 x = round((maxLat-minLat)*111 / side)
 y = round((maxLong-minLong)*longInKm / side)
 z = round((maxdep-mindep) / side)

 # Create cubes indexes for all the dimensions
```

```
 xlat = [math.floor((i-minLat)*x/(maxLat-minLat))+1
  for i in lat]
 ylong = [math.floor((i-minLong)*y/(maxLong-minLong))+1
  for i in long]
 zdep = [math.floor((i-mindep)*z/(maxdep-mindep))+1
  for i in dep ]
 cubeIndex=[]
 # and a general cubeindex for graph formation
 for i in range(len(xlat)):
  cubeIndex.append(int(str(xlat[i])+str(ylong[i])
   +str(zdep[i])))


 # Input coordinates of every cubeIndex
 cubelat=[round(minLat + (side/111)*(i-1+1/2),4)
  for i in xlat]
 cubelong=[round(minLong + (side/longInKm)*(i-1+1/2),4)
  for i in ylong]
 cubedep=[round(mindep + (side)*(i-1+1/2),4)
  for i in zdep]

 # if enRel TRUE, add a column with the energy release
 if enRel:
  enRel=[math.pow(10, 5.24 + 1.44*i) for i in mag]

  d = {'date':date,'lat':lat,'long':long,
    'dep':dep,'mag':mag, 'enRel':enRel,'x':xlat,
    'y':ylong,'z':zdep,'cubeIndex':cubeIndex,
    'cubelat':cubelat,'cubelong':cubelong,
    'cubedep':cubedep}
  return pd.DataFrame(data=d)


 else:
  # Create DataFrame
  d = {'date':date,'lat':lat,'long':long,
    'dep':dep,'mag':mag,'x':xlat,
    'y':ylong,'z':zdep,'cubeIndex':cubeIndex,
    'cubelat':cubelat,'cubelong':cubelong,
     'cubedep':cubedep}
  return pd.DataFrame(data=d)

# if withCubes==False , do not create cubes
else:
```

```
    if enRel:
     enRel=[math.pow(10, 5.24 + 1.44*i) for i in mag]

     d = {'date':date,'lat':lat,'long':long,'dep':dep,
          'mag':mag,'enRel':enRel}
     return pd.DataFrame(data=d)
    else:
     d = {'date':date,'lat':lat,'long':long,
       'dep':dep,'mag':mag}
     return pd.DataFrame(data=d)
```

Appendix B

Network Creation

```
import datetime
import pandas as pd
import networkx as nx


# Takes the qk DataFrame and returns the qk network
# in networkx graph (G) with attributes:
#(cube dimensions and index of qk with same cIdx)
def graphCreation(qk,withEdgeWeight=False):

 # Create graph -  qk ONE BY ONE
 G = nx.Graph()
 # Iterate through the earthqk checking
 # for magnitude condition
 for i in range(len(qk['cIdx'])):
  # Create a node for this earthquake
  G.add_node(qk['cIdx'][i])
  # check the following earthqk chronologically
  for j in range(i+1, len(qk['cIdx'])):
   # I want only the first occurence (break at the end)
   # create a node for the target quake index
   G.add_node(qk['cIdx'][j])

   # Check if you need the edge weights
   if withEdgeWeight==True:
    # WITH EDGE WEIGHT
    # check to see if there is an edge between them
    if G.has_edge(qk['cIdx'][i], qk['cIdx'][j]):
     # we added this one, increase the weight by 1
     G[qk['cIdx'][i]][qk['cIdx'][j]]['weight'] += 1
     break

    else:
     # new edge. add with weight=1
     G.add_edge(qk['cIdx'][i], qk['cIdx'][j], weight=1)
     break
   else:
    # WITHOUT EDGE WEIGHT
```

```
    G.add_edge(qk['cIdx'][i], qk['cIdx'][j])
    break

# Setup network attributes as dictionaries
# Only dictionaries are supported by networkx
quake_index = {}
quake_zDepth = {}
quake_yLongitude = {}
quake_xLatitude = {}

# Turn the dataframe into a dictionary, the indexing qty
for index, row in qk.iterrows():

 # Create a dictionary with cIdx :[list of quake idxs]
 # if the cube has already been indexed
 if row['cIdx'] in quake_index.keys():
  # just add that event in the dictionary
  quake_index[row['cIdx']].append(str(index))
 # if not yet indexed
 else:
  # create that dictionary entry
  quake_index[row['cIdx']]=[]
  # and add that event in the dictionary
  quake_index[row['cIdx']].append(str(index))

 # Create dict for each dimension (cIdx : dimension)
 quake_zDepth[row['cIdx']] = str(row['z'])
 quake_yLongitude[row['cIdx']] = str(row['y'])
 quake_xLatitude[row['cIdx']] = str(row['x'])


nx.set_node_attributes(G, quake_zDepth, 'quake_zDep')
nx.set_node_attributes(G, quake_yLongitude, 'quake_yLong')
nx.set_node_attributes(G, quake_xLatitude, 'quake_xLat')
nx.set_node_attributes(G, quake_index, 'quake_index')


# Relabel the nodes
nodeList=[]
for n in G.nodes():
 nodeList.append(n)

# Create the mapping : dict with {G node value} :
#new value ( from 1 to n = len of G nodes )
mapping = {}
```

```
 for i in range(len(nodeList)):
  # i+1 to create from 1 to n ( not from 0 )
  mapping[nodeList[i]] = i+1

 # Create new graph with relabeled nodes
 G = nx.relabel_nodes(G, mapping)

 return(G)
```

Appendix C

Connectivity

```
from quakesNetwork_graphCreation import*
from sqlCollectDatabaseWithCubes import sqlCollect

mag=1
side=5

region = Vrancea
edgeWeight = input('␣Edgeweight␣-␣True␣/␣False␣:␣')

# VRANCEA
if region=='Vrancea':
 condition=(f"SELECT␣*␣FROM␣romplus␣WHERE
␣␣␣␣`dateandtime`>='1976-01-01␣00:00:00'"
      f"␣AND␣`latitude`>=45.2␣AND
␣␣␣␣␣␣␣␣`latitude`<=46␣AND␣`longitude`>=26"
      f"␣AND␣`longitude`<=27␣AND␣`depth`>=50
␣␣␣␣␣␣␣␣AND␣`depth`<=200"
      f"␣AND␣`magnitude`>={mag}")

# Collect the database and create the graph
quakes = sqlCollect(condition,side,region)

# Process graph and its connectivity with edge weight
if edgeWeight=='True':
 G = graphCreation(quakes,withEdgeWeight=True)

 # Degree
 degree_dict = dict(G.degree(G.nodes(),weight='weight'))
 connectivity=[]
 for item in degree_dict.values():
  connectivity.append(item)
# Withoud weight
else:
 G = graphCreation(quakes)

 # Degree
 degree_dict = dict(G.degree(G.nodes()))
 connectivity=[]
```

```
 for item in degree_dict.values():
  connectivity.append(item)

# Histogram of connectivity
hist, bins = np.histogram(connectivity,
    bins=round(math.sqrt(len(connectivity))));

# Create the data k,y from the hist and bins
# k = bins centers. Create empty array of the length of hist
k = np.zeros_like(hist)
# Append to it the centers of the bins
for i in range(1,len(bins)):
 k[i-1]=((bins[i]+bins[i-1])/2)

# Check for zeros in the hist list and cut both
#k and y where the first zero occurs
for i in range(len(hist)):
 if hist[i]==0:
  y=np.array(hist[:i])
  k=np.array(k[:i])
  break
 # If there is no zeros, make y=hist (full data )
 else:
  y=hist

# Renormalize the ydata to 1 => P_k
P_k = np.array([float(i)/sum(y) for i in y])
# Compute the mean degree of G : k_mean
k_mean = np.dot(k,P_k)
# Compute the remaining degree distribution q_k
q = np.array([(k[i+1]*P_k[i+1])/k_mean for
  i in range(len(k)-1)])
# Compute the entropy
H_q = -np.dot(q,np.log(q))


# Compute the power_law fit to our data
pars, cov = curve_fit(f=power_law,xdata=k,ydata=P_k,
 maxfev=5000)
# Compute the chi_sq  of fit = sum((obs-exp)^2/exp)
chi_squared = np.sum((P_k-power_law(k,*pars))
 **2/power_law(k,*pars))


fig, ax = plt.subplots(1,2,figsize=(15,5))
```

```
# The data, scattered
plt.scatter(k,P_k)
plt.set_xscale('log')
plt.set_yscale('log')
# The fit
plt.plot(k,power_law(k,*pars),
      label=f'$\gamma$={np.round(pars[1],4)}\n
     $\chi^2$={np.round(chi_squared,4)}',
      color='red')

# Legend : gamma coefficient of fit and chi_sq
plt.legend(loc='upper right',fontsize=16,frameon=True)

# Title of connectivity distribution ( data + fit )
plt.set_title('cube size = 5 km ')
plt.set_xlabel('connectivity k')
plt.set_ylabel('P(k)')
```

Appendix D

Motifs

D.1 Mean and Total Energy per Motif

```
# Calculates the total and mean energy per motif
def totalMeanEnergyMotif(region,side,mag,
  motif,G,qkDataFrame):
    # Open the motifs file
    fileMotif = open(f'motif.txt', 'r')
    linesMotif = fileMotif.readlines()

    # Properly evaluate the Lines to get the Lists
    motifNodes=[]
    for item in linesMotif:
        motifNodes.append(ast.literal_eval(item))


    # Dictionary for setting motif :
    energyMotif = {}

    # Iterate through the Motifs in our list
    for motifs in motifNodes:

        # lists ( 3 elements ) with total and
        #mean energy in Motif ( per nodes )
        energyInMotif = []
        meanEnergyInMotif = []

        # Iterate through the nodes of the motif
        for i in range(len(motifs)):

            # List the quakes in the node
            qkInNode = G.nodes[int(motifs[i])]['qk_idx']
            # Set the energy of that node to 0
            energyNode = 0

            # Iterathe through the quakes of that node
            for quake in quakesNode:
                # Raise the total energy accumulated
```

```
                energyNode += qkDataFrame['enRel'][int(qk)]

            energyInMotif.append(energyNode)
            meanEnMotif.append(energyNode/len(quakesInNode))


        if motif=='triangles':
            energyMotif[str(motifs)]=
            [sum(energyInMotif),sum(meanEnMotif)/3]
        if motif=='squares':
            energyMotif[str(motifs)]=
            [sum(energyInMotif),sum(meanEnMotif)/4]

    return(motifNodes,energyMotif)
```

D.2 Area of Triangle Motif calculation

```
# Uses the magnitudesMotif to calculate
# area/totalMag and area/meanMag per motif
def areasInTriangles(motifNodes,energyMotif,G,qk):

    # Initialize lists for areas
    areas=[]
    areasWeightTotalMag=[]
    areasWeightMeanMag=[]

    # Points of trianlge: x=lat, y=long, z=depth
    x=[0,0,0]
    y=[0,0,0]
    z=[0,0,0]

    # Iterate through the motifs
    for tr in motifNodes:
        for i in range(3):
            x[i] = qk['cubeLat']
              [int(G.nodes[int(tr[i])]['qk_idx'][0])]
            y[i] = qk['cubeLong']
              [int(G.nodes[int(tr[i])]['qk_idx'][0])]
            z[i] = qk['cubeDep']
              [int(G.nodes[int(tr[i])]['qk_idx'][0])]

        # The 3 points of the triangle in GEOPY Point
        X=Point(x[0],y[0],z[0])
```

```
        Y=Point(x[1],y[1],z[1])
        Z=Point(x[2],y[2],z[2])

        flat_distance_a = distance(X[:2], Y[:2]).km
        flat_distance_b = distance(Y[:2], Z[:2]).km
        flat_distance_c = distance(X[:2], Z[:2]).km

        # Introduce the altitude euclidian distance
        a = math.sqrt(flat_distance_a**2 + (X[2] - Y[2])**2)
        b = math.sqrt(flat_distance_b**2 + (Y[2] - Z[2])**2)
        c = math.sqrt(flat_distance_c**2 + (X[2] - Z[2])**2)

        # calculate semiperimeter
        sp = (a+b+c)/2

        # Use Heron's formula
        A = math.sqrt(abs(sp*round(sp-a,4)
         *round(sp-b,4)*round(sp-c,4)))
        if A < 5:
            continue
        else:

            AWeightTotalMag = A / energyMotif[str(tr)][0]
            areasWeightTotalMag.append(AWeightTotalMag)

            AWeightMeanMag = A / energyMotif[str(tr)][1]
            areasWeightMeanMag.append(AWeightMeanMag)

    return(areasWeightTotalMag,areasWeightMeanMag)
```

Appendix E

Paraview Visualization

E.1 VTK

```
import vtk

def writeObjects(nodeCoords,
                 motifCoords={},
                 edges = [],
                 scalar = [], name = '', power = 1,
                 escalar = [], ename = '', epower = 1,
                 method = 'vtkPolyData',
                 fileout = 'test'):

    # INITIALIZE THE NODES
    points = vtk.vtkPoints()
    for node in nodeCoords:
        points.InsertNextPoint(node)

    if motifCoords:
        # INITIALIZE TRIANGLES
        if len(motifCoords[0]) == 3:
            triangles = vtk.vtkCellArray()
            # Initialize the points for the triangle
            for motif in motifCoords:
                triangle = vtk.vtkTriangle()
                keys=[]
                for nodeIdx in motif.keys():
                    keys.append(nodeIdx)

                triangle.GetPointIds().SetId(0,keys[0]-1)
                triangle.GetPointIds().SetId(1,keys[1]-1)
                triangle.GetPointIds().SetId(2,keys[2]-1)

                # Append the created triangle to the triangles
                triangles.InsertNextCell(triangle)


    # INITIALIZE THE EDGES
```

```
    if edges:
        line = vtk.vtkCellArray()
        line.Allocate(len(edges))
        for edge in edges:
            line.InsertNextCell(2)
            line.InsertCellPoint(edge[0])
            line.InsertCellPoint(edge[1])

    if scalar:
        attribute = vtk.vtkFloatArray()
        attribute.SetNumberOfComponents(1)
        attribute.SetName(name)
        attribute.SetNumberOfTuples(len(scalar))
        for i, j in enumerate(scalar):
            attribute.SetValue(i,j**power)


    if escalar:
        eattribute = vtk.vtkFloatArray()
        eattribute.SetNumberOfComponents(1)
        eattribute.SetName(ename)
        eattribute.SetNumberOfTuples(len(edges))
        for i, j in enumerate(escalar):
            eattribute.SetValue(i,j**epower)


    if method == 'vtkPolyData':
        polydata = vtk.vtkPolyData()
        polydata.SetPoints(points)

        if motifCoords:
            polydata.SetPolys(triangles)
        if edges:
            polydata.SetLines(line)
        if scalar:
            polydata.GetPointData().AddArray(attribute)
        if escalar:
            polydata.GetCellData().AddArray(eattribute)
        if nodeLabel:
            polydata.GetPointData().AddArray(label)
        writer = vtk.vtkXMLPolyDataWriter()
        writer.SetFileName(fileout+'.vtp')
        writer.SetInputData(polydata)
        writer.Write()
```

E.2 Paraview

```
import vtk
import networkx as nx
import ast
import numpy as np

# Select the cube side length in km: ( 10 / 5 km)
side = 5
# Select desired magnitude threshold
mag = 4
# Select region
region='Vrancea'

G = nx.read_gexf(f'quakes{region}_{side}km_{mag}mag.gexf')

# Use readlines() to open the
fileTriangle = open(f'quakes{region}_triangles.txt', 'r')
linesTriangle = fileTriangle.readlines()

# Properly evaluate the Lines to get the Lists
triangleNodes=[]
for item in linesTriangle:
    triangleNodes.append(ast.literal_eval(item))

# Graph containing triangles only
H = nx.Graph()
for item in triangleNodes:
    H.add_edge(int(item[0]),int(item[1]))
    H.add_edge(int(item[1]),int(item[2]))
    H.add_edge(int(item[0]),int(item[2]))

# Set the triangle attribute = 0 to each edge
nx.set_edge_attributes(G, 0, name='triangle')

# Iterate through our triangle only network edges
for (u,v) in H.edges():
    # Assign to og network edges attribute triangl =1
    G[u][v]['triangle'] = 1

# Get attributes that go into VTK function

# X Y Z coords
lat =[]
```

```
long = []
depth =[]
for n in G.nodes():
    lat.append(int(G.nodes[n]['quake_xLatitude']))
    long.append(int(G.nodes[n]['quake_yLongitude']))
    depth.append(int(G.nodes[n]['quake_zDepth']))

minLat = min(lat)
maxLat = max(lat)
minLong = min(long)
maxLong = max(long)
minDepth = min(depth)
maxDepth = max(depth)
maxDim = max(maxLat,maxLong,maxDepth)


coords=[]
for n in G.nodes():
    coords.append([ np.float32(round((int(G.nodes[n]
        ['quake_xLatitude'])- minLat)*(maxLat/max)
          /(maxLat-minLat),3)),
         np.float32(round((int(G.nodes[n]
           ['quake_yLongitude'])- minLong)*(maxLong/maxDim)
             /(maxLong-minLong),3)),
         np.float32(round((int(G.nodes[n]
           ['quake_zDepth'])- minDepth)*(maxDepth/maxDim)
             /(maxDepth-minDepth),3))])

# Degree of nodes edges
degree = [d for n, d in G.degree()]

# Weight of edges
weights = []
for (i,j) in G.edges():
    weights.append(G[i][j]['weight'])

# Triangle quality of edges
triangles = []
for (i,j) in G.edges():
    triangles.append(G[i][j]['triangle'])

from writeNodesEdges import writeObjects

writeObjects(nodeCoords=coords,
             edges=G.edges(),
```

```
                scalar=degree, name='degree',
                scalar2=weights, name2='weight',
                escalar2=triangles, ename2='triangle',
#                nodeLabel=nodeLabel,
                fileout=f'network_triangles')
```

Appendix F

Temporal AutoCorrelations

```
import arviz as az
import matplotlib
import numpy as np
import datetime

from datetime import datetime
from datetime import timedelta



import matplotlib.pyplot as plt
import math
import collections

from statsmodels.graphics.tsaplots import plot_acf
from sqlCollectDatabaseWithCubes import sqlCollect
from quakesNetwork_graphCreation import*

region='California'
magMin=2
magMax=3.5
side=5

condition=(f"SELECT * FROM california WHERE
`dateandtime`>='1984-01-01 00:00:00'"
            f" AND `depth`>=0 AND `magnitude`>={magMin}
AND `magnitude`<={magMax}"
            f" AND (`magtype` LIKE 'l'
OR `magtype` LIKE 'w')")

quakes = sqlCollect(condition,side,region,enRel=True)

timeWindow = max(quakes['date']) - min(quakes['date'])
deltas = [quakes['date'][i+1]-quakes['date'][i]
 for i in range(len(quakes['date'])-1)]
dt = max(deltas)

delta = timedelta(days=1) + dt
windows = round(timeWindow/delta)
```

```
interval = {}
for i in range(windows):
    interval[i] = [quakes['date'][0] +
     i*delta,quakes['date'][0] + (i+1)*delta]


quakesList = {}
enInt = dict.fromkeys(interval.keys(),0)
for i in range(len(interval)):
    quakesList[i] =list(set(quakes.index[interval[i][0]<=
     quakes['date']].tolist()).intersection
     (quakes.index[quakes['date'] <
      interval[i][1]].tolist()))
    for k in quakesList[i]:
        enInt[i] += quakes['energyRelease'][k]

data = list(energyInterval.values())
values = np.array(data)

f = plt.figure()
ax = f.add_subplot(111)
plot_acf(values,lags=len(energyInterval)-1,ax=ax)

plt.xlabel(r'k (lags)')
plt.ylabel(r'C(k)')
```