Towards Data Self-Awareness: automated aggregation, geometrization, predictability and cross-validation of real-world data.

by

Veniamin L. Smirnov

A Dissertation

In

Mathematics and Statistics

Submitted to the Graduate Faculty
of Texas Tech University in
Partial Fulfillment of
the Requirements for the Degree of

Doctor of Philosophy

Approved

Dr. Dimitri Volchenkov
Chair of Committee

Dr. Dmitri Pavlov
Committee Member

Dr. Alexander Solynin
Committee Member

Dr. Akbar Namin
Graduate Dean's Representative

September, 2021

ACKNOWLEDGEMENTS

I wish to acknowledge the help and support of my advisor Dr. Dimitri Volchenkov. Without his continuous advice, guidance and help, this research would not have been a success. Also I would like to thank AVX Aircraft, PeopleTec for funding our research in Aircraft Maintenance Data. I have been blessed to be taught by Dr. Dmitri Pavlov and Dr. Alex Solynin and am very grateful for their valuable insight in Topology and Complex Analysis. Finally, I am very thankful for my family who supported me throughout my journey at TTU.

TABLE OF CONTENTS

Acknowledgements . . . iii
Abstract . . . vi
List of Tables . . . vii
List of Figures . . . viii
1. Introduction . . . 1
2. Multi-scale Analysis of Urban Spatial Structures acquired from *OpenStreetMap* . . . 9
2.1 Introduction . . . 9
2.2 Spatial graphs of urban environments . . . 11
2.3 Discrete Time Anisotropic Scale-dependent Random Walks . . . 12
2.4 Stationary Distributions and Balance Equations of Anisotropic Random Walks . . . 14
2.5 Exploring Graphs with Random Walks . . . 15
2.6 The City of Lubbock is Running Away . . . 17
2.6.1 Isolation index . . . 19
2.6.2 Integration index . . . 20
2.7 Discussion and Conclusion . . . 22
3. Five Years of Phase Space Dynamics of the Standard & Poor's 500 . . . 24
3.1 Introduction . . . 24
3.2 Data source, data validation and preparation . . . 26
3.3 The discrete model of Standard & Poor's 500 phase space . . . 27
3.4 Methods . . . 27
3.4.1 Predictability of stock behavior in market phase space . . . 28
3.4.1.1 Information content of cells in market phase space . 28
3.4.1.2 T-days mutual information between cells in market phase space . . . 29
3.4.1.3 T-days precursors for market events . . . 29
3.4.2 Predictable and unpredictable information estimated from an empirical transition matrix . . . 30
3.5 Results . . . 33

3.5.1 Visualizing the stock market crashes, rallies, and market tumbling on shocking events . . . 33
3.5.2 Phase portrait , t–days predictability of states, and t–days mutual information in S&P 500 phase space. . . . 34
3.5.3 Predictable and unpredictable information in company stock prices. Stocks of different mobility . . . 36
3.6 Discussion and Conclusion . . . 38
4. Extreme events and emergency scales . . . 45
4.1 Introduction . . . 45
4.2 Data source and description . . . 50
4.3 Log returns: between noise and random walks . . . 51
4.4 Tails, thresholds, and extreme events . . . 54
4.4.1 Generalized extreme value distributions . . . 55
4.4.2 How to choose a threshold . . . 58
4.4.2.1 Graphical approaches to estimate threshold . . . 59
4.4.2.2 Automatic methods to estimate thresholds . . . 61
4.4.2.3 Rules of thumb to choose a threshold . . . 66
4.4.3 Statistics of extreme events under threshold uncertainty . . 71
4.5 Defining Emergency Scales by thresholds uncertainty. . . . 74
4.6 Conclusion . . . 77
5. Flight data refinement and reduction. . . . 79
5.1 Introduction . . . 79
5.2 Automated Non-flight Data Files Filtering Algorithm . . . 81
5.3 Synchronization of Flight and Maintenance data systems . . . 84
5.4 Missing data restoratiion via VADR/HUMS/ULLS Cross-validation 90
5.5 Time series healing and data manifolds. . . . 95
5.6 Automated Assessment of Flight Operational Stages . . . 99
6. Conclusion . . . 104
Bibliography . . . 106
Appendix: Computer code . . . 118

ABSTRACT

We propose a strategy for automated assessment of self-awareness of real-world data sets based on automated aggregation, geometrization, predictability and cross-validation. The strategy was tested using three different scenarios: urban spatial structures acquired from the OpenStreetMap in which We propose computationally feasible statistical algorithms for the automated assessment of isolation and integration of urban locations and neighborhoods by using maps acquired from the Open-StreeMap service; Standard & Poor's 500 financial time series in which we showed that inhomogeneous density of states in a discrete model of Standard & Poor's 500 phase space leads to inequitable predictability of market events, most frequent events might be efficiently predicted in the long run as expected from Mean reversion theory; and Maintenance and Flight data provided by the Department of Defense.

Keywords: Anisotropic random walks, Measures of isolation and disintegration, Urban studies, Measures of information, Entropy, Mathematical finance, Extreme events, Emergency scales, Uncertainty of threshold.

LIST OF TABLES

4.1 Parameter Estimates for the GEV fitted model with Maximum Likelihood Estimator. The 95% confidence intervals for each estimates are included. 58
4.2 Location of extrema points of the uncertainty curves from Fig. 4.17. . 77

LIST OF FIGURES

1.1 The strategy of automated evaluation of data self-awareness. 2
2.1 The city of Lubbock is running away from the lasso of *Loop 289* marking the old city boundaries. 18
2.2 Isolation index of nodes in the city graph of Lubbock, TX acquired from the *OpenStreetMap* service. The colored bar indicates the value of isolation index in the decibel scale – the isolated neighborhoods are red colored. The axis labels are the geographic coordinates. 20
2.3 Integration index of nodes in the city graph of Lubbock, TX acquired from the *OpenStreetMap* service. The colored bar indicates the value of integration index in the decibel scale – the integrated neighborhoods are red colored. The axis labels are the geographic coordinates. . . . 21
2.4 The distribution of social and economic variables in the city of Lubbock, TX. a.) The racial composition of neighborhoods in Lubbock accordingly the US Census Bureau. The area of white population concentration are dark colored; b.) Lubbock crime rates and statistics density heat map (downloaded on 6/29/2018) from the Turlia website [56]. Trulia uses crime reports to provide valuable information on the relative safety of homes in the US; c.) Unemployment Rate in Lubbock, TX in Oct. 2018 accordingly the US Bureau of Labor Statistics. 22
3.1 The smoothed phase space trajectories (individual stock prices at market close, one observation per day). a.) The Amazon company stocks (from 12/18/2013 to 01/10/2014); b.) The Apple company stocks (from 02/21/2018 to 03/14/2018). The regions of negative return are colored in red; the regions of positive returns are green colored. . . . 28

3.2 a.) The state diagram for tossing a unfair coin, in which each state ('*heads*' or '*tails*') repeats itself with the probability $0 \leq p \leq 1$. b.) The information components $\mathcal{D}(p)$, the past-future mutual information (excess entropy [70]); $\mathcal{U}(p)$, the conditional mutual information available at the present state of the chain and relevant to the future states; $\mathcal{E}(p)$, the ephemeral information existing only in the present state of the chain, being neither a consequence of the past, nor of consequence for the future. 31

3.3 The daily snapshots of S&P 500 stocks in phase space during a.) a stock market crush; b.) a stock market rally; c.) & e.) and d.) & f.) a market tumbling phenomenon. The regions of phase space characterized by the positive/ negative values of return are colored by 'green' and 'red', respectively). 40

3.4 a.) The color - coded histogram indicating the number of visits of the phase space cells throughout the entire period of observations; b.) The color - coded histogram representing the information content of phase space cells; c.) The predictability of states based on 1-day precursors; d.) The predictability of states based on 24-day precursors. 41

3.5 T–days mutual information of S&P 500. 42

3.6 The shares of transition information predictable from the historic time series ($\mathcal{D}$), from the present observation ($\mathcal{U}$), and ephemeral information ($\mathcal{E}$) vs. Shannon's entropy H measuring uncertainty of daily return for every studied company. The lines represent the polynomial trends for the empirically observed relations. 42

3.7 Uncertainty of daily stock price measured by Shannon's entropy H along with the information components, $\mathcal{D}$, $\mathcal{U}$ and $\mathcal{E}$, for 468 major companies from the S&P 500. 43

3.8 A "unfair coin" of the S& P 500 stock market. a.) Unpredictable ($\mathcal{E}$) vs. predictable information ($\mathcal{D}+\mathcal{U}$) in unfair coin tossing Fig. (3.2. a); b.) Predictable information available from the present state alone ($\mathcal{U}$) vs. predictable information available from the past series ($\mathcal{D}$) in unfair coin tossing Fig. (3.2. a); c.) Unpredictable ($\mathcal{E}$) vs. predictable information ($\mathcal{D}+\mathcal{U}$) in the S& P 500; d.) Predictable information available from the present state alone ($\mathcal{U}$) vs. predictable information available from the past series ($\mathcal{D}$) in the S& P 500. 44
4.1 Triality of an extreme event. 47
4.2 Degree of uncertainty of different values of the thresholds based on the amount of trading days taken into account. Three shaded regions mark three scales of emergency: I - subcritical, II - critical, III - extreme and the solid curve represents uncertainty of the Red Queen State. 48
4.3 The S&P 500 Index of 500 large-cap U.S. stocks assessing market performance. 50
4.4 The S&P 500 index log-return at market close. 51
4.5 Distribution of log-return values in the $\log 2$-linear scale. Solid lines correspond to a Zipf's Law ($\propto x^{-1}$), dotted lines represent Power Law ($\propto x^{-3.5}$), and dashed lines correlate to Gaussian distribution ($\propto \exp(-x^2)$). Curves are given for reference only. 52
4.6 The q-order Hurst exponents H_q for the time series of positive (the dashed line) and negative (the bold line) log-returns. 54
4.7 a.) A general extreme value QQ-plot with maximum likelihood estimation ; b.) Density plot of empirical data where a dashed curve A is based on the empirical data, and a dashed curve B is modeled. $N = 2036$ and bandwidth is 135.9. 57
4.8 Mean residual life plot for the S&P 500 positive returns. Solid jagged line is empirical MRL with approximate pointwise Wald 95% confidence intervals as dashed lines. The threshold u is estimated at 0.016. A vertical dashed line marks this threshold. 61

4.9 Mean residual life plot for the S&P 500 negative returns. Solid jagged line is empirical MRL with approximate point-wise Wald 95% confidence intervals as dashed lines. The threshold u is estimated at 0.017. A vertical dashed line marks this threshold. 62
4.10 Distribution of exceedances normalized by thresholds $u = 0.016$ for positive and $u = -0.017$ for negative returns, respectively. Dotted lines represent Zipf's Law ($\propto x^{-1}$), dashdot lines represent Gaussian distribution ($\propto \exp(-x^2)$) and dashed lines represent power law ($\propto x^{-3.3}$). The curves are given for reference only. 63
4.11 a.) Quantile-Quantile plot with maximum likelihood estimation for the negative threshold; b.) QQ-plot with maximum likelihood estimation for the positive threshold. 64
4.12 Value of the thresholds for positive and negative log-return based on L-moments. The solid black and blue lines correspond to negative and positive log return thresholds, respectively, and based on a window of 100 trading days. The dotted black and blue lines correspond to negative and positive log return thresholds, respectively, and based on a window of 400 trading days. 66
4.13 Possible threshold changing ranges from 03/11/1981 to 12/31/2018 based on 300 preceding trading days. A green strip represents positive log-return and an orange strip shows the threshold domain for negative log-return values. 68
4.14 Possible threshold changing ranges from 05/18/1982 to 12/31/2018 based on 600 preceding trading days. A green strip represents positive log-returns and an orange strip shows the threshold domain for negative log-return values. 69
4.15 The probability that a threshold of the log-return values will be changed on any given day calculated over the different data windows ranging from 25 to 6000 trading days. 70

4.16 The statistics of time intervals (in days) between the sequential extreme events for the fixed threshold values, $u = 0.016$ and $u = -0.017$, for positive and negative fluctuations of the log-return respectively. The solid line $\propto t^{-2}$ corresponding to the asymptotic quadratic decay (4.12) is given for reference. 73
4.17 Degree of uncertainty of different values of the thresholds based on the amount of trading days taken into account. Three shaded regions mark three scales of emergency: I - subcritical, II - critical, III - extreme and the solid curve represents the Red Queen State. 76
5.1 Data Refinement and Reduction flowchart. 80
5.2 An example of a file that does not contain any actual flight information. 81
5.3 Machine Learning Algorithms (K-Means and SVM) applied to multiple functional time series for flight stage detection. 83
5.4 Perfect matching of the flight stages with Army-defined flight operational regimes. 84
5.5 Distribution of flight and non-flight HUMS files. 85
5.6 Baseline Time of an Aircraft. 86
5.7 File attribution problem. Matching parts of HUMS, VADR data sets. 86
5.8 VADR-HUMS attribution and synchornization: Step 1 Downsampling. 87
5.9 VADR-HUMS attribution and synchornization: Step 2 Triple iterative synchronization refinement. 88
5.10 VADR-HUMS attribution and synchornization: Step 3 Two-way move of the algorithm. 90
5.11 VADR-HUMS attribution and synchornization. Intermediate results using MASS algorithm and SSIM. 90
5.12 VADR-HUMS attribution and synchornization. Final results. 91
5.13 Distribution of Flight and Maintenance data sets of the year of 2018. 92
5.14 Application of Needleman algorithm to fusion of Flight and Maintenance data systems. 93
5.15 Final results of fusion of the Flight and Maintenance data systems. . 94

5.16 a.) Entries in which all flights we combines by a pilot to a single entry found by the algorithm; b.) Entries in which one flight we split by a pilot to several entries found by the algorithm; c.) Entries in which flight data was stored only by either HUMS or VADR. 94
5.17 Distribution of HUMS (red bars), VADR (green bars), and ULLS-A (black bars) files on a time line. 95
5.18 a.) Complete time series of a flight restored from both VADR and HUMS; b.) An example of a table containing all information about restored flight, including time delta between HUMS and VADR and total duration of the flight. 96
5.19 Gallons of fuel per hour readings generated by a fuel pump (Original data readings). 96
5.20 Gallons of fuel per hour readings generated by a fuel pump (Original data readings) in the phase space. The points in the green rectangle are the regular point of the data manifold. The points inside the red rectangles are the outliers generated by the drops in the readings. . . 97
5.21 Gallons of fuel per hour readings after returning irregular points back to the regular data manifold. 98
5.22 Air speed original readings (blue time series) along with the associated data manifold in the phase space, and the air speed readings after cross validation (orange time series). 99
5.23 a.The data monifold for Indicated Speed- 1; b.) The data monifold for Pitch-1 c.) The data monifold for Acceleration in y-direction; d.) The data monifold for Velocity Yaw. 100
5.24 Ten time series used as the regime triggers. 100
5.25 Automatically identified Regimes using the regime triggers. Blue time series is for original regime readings and the orange time series shows the regimes identified by the Support Vector Machine. 101
5.26 The color-coded sub-manifold of the data manifold of the annual aircraft kinematic readings in Regime 3. 102

CHAPTER 1
INTRODUCTION

My name is Veniamin. I am a graduate student. I live in Lubbock. I like cooking. These few sentences describe me as a self-aware person. I know who I am and where I am located. I am also capable of discriminating myself from other human beings. This helps me make decisions appropriate to achieve my goals. I am capable of predicting my future states with some degree of certainty. I am able to evaluate my actions and observe mistakes I make and react to unexpected events in my life.

Since invention of the perceptron algorithm in 1958 [1], scientists tried to mimic, to some extent, a human being brain to make machine learning algorithms more intelligent. As of today, there exists a plethora of algorithms capable of solving a broad range of real-life problems, such as, supervised learning, reinforcement learning, multi-task learning, ensemble learning, neural networks, etc., to name a few. In general, the purpose of machine learning is to learn from the data [2]. Depending on a particular problem, some of the algorithms might produce great solutions, while others may miserably fail, which requires human interference and evaluation of the problem a practitioner tries to solve. Taking the problem further, even if a result of applying machine learning to a data set may seem satisfactory, we still have to deal with an uncertainty of our findings: was the data set reliable to begin with, will our model generalize to new coming data. If a data point looks unusual, does it mean we witness an error or a phenomenon that our model cannot explain, etc.

The concept of self-awareness in computing systems originates from Kephart's work on autonomic computing in 2003 [3], and the complete definition of self-awareness is offered by Kounev [4, 5], which includes self-reflection, self-prediction, and self-adaption.

Ideally, we would like to have a set of algorithms that can meet the above characteristics of the self-aware systems on their own, i.e. the set of algorithms making a given data set self-aware of its own geometry, a current state and be able to predict future states, be able to separate extreme events from erroneous data points, etc.

Recently self-awareness of data was studying in the compression systems, whose task-parallelism and data-parallelism can be adaptively adjusted according to differ-

ent input data [6], time series forecasting [7], engineering of a self-aware aerospace vehicle that can adapt the way they perform missions by gathering information about themselves and their surroundings and responding intelligently. [8], self-aware neural network systems [9], etc. By reviewing the employed techniques, we observe that the idea of data self-awareness is still in a preliminary stage.

In this dissertation, we propose a strategy of automated evaluation of data self-awareness that is schematically shown in Fig. 1.1.

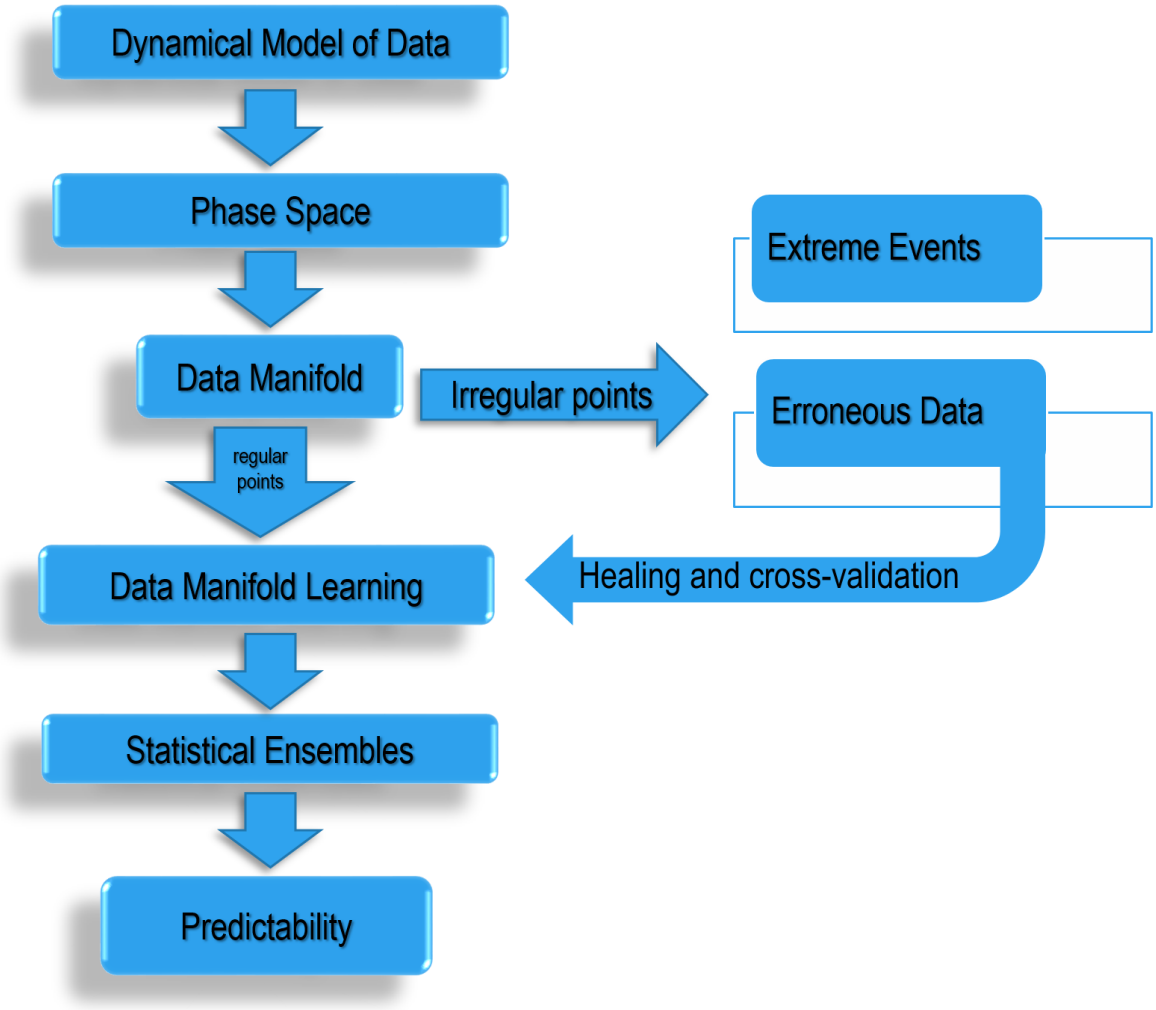


Figure 1.1. The strategy of automated evaluation of data self-awareness.

The root of the strategy lies in a *dynamical model*. In general, the dynamic model describes the behavior of a distributed parameter system in terms of how one qualitative state can turn into another. A qualitative state is described by a static model, i.e. the distributions and intersections of the qualitative fields at a particular time instant or interval. Further, we need to equip the dynamical model with the notion of a Euclidean metric. In terms of a graph G, we will require distances between nodes of the graph G, and the angles between vectors pointing at them by the means of the inner product between pairs of vectors [31]. Equipped with the Euclidean structure, we make one step further to introduce a length structure to study the random walks induced on the graph G. Every path in the graph G is associated with some probability of being followed by a random walker. Thus, the path length statistics is a suitable choice for the length structure on G [31].

The dynamical model lies a foundation to a *phase space*, the space in which the model lives. There could be several models for one data set and each of them will define a different phase space. Phase spaces are commonly used in data analysis: in medicine for tomography [10], or cardiac arrhythmia detection [11]; in chemistry to study chemical dynamics [12], syntheses in solid state chemistry [13], chemistry of semiconductor clusters [14], etc.; in physics [15, 16]. While studying S&P 500, the financial time series were submerged into the "Return vs Roughness" phase space; in case of the time series generated by the engineering systems of the HM-60 Black Hawk helicopters, "Position vs Velocity" phase space was more suitable to perform data analysis.

In both examples above, passing data to the phase space resulted in the dimensionality increment. Many practitioners try to reduce dimensionality instead to avoid the curse of dimentionality [17], however, in case of the phase spaces, we are able to add extra freedom to the given data without any negative consequences.

As we study the behavior of the model in the phase space, we can see that some states (points) are visited more often (or easily accessible) than the other states (points); all of which exist on a certain *data manifold*. Manifold-valued data have gained much importance in recent times due to their expressiveness and ready availability of machines [18]. However, data manifold learning spins around embedding high dimensional data to the lower dimensional space, preferably, without loosing any

useful information, our interest towards data manifold learning lies in the realm of predictability. As the dynamical system evolves in the phase space, we may separate the manifold into regions:

- the most visited states (regular points) have higher predictability,
- the parts of the manifold that are visited rarely (irregular points).

The irregular points are of two different kinds: erroneous ones caused by human errors or technical glitches and extreme events, the states that cannot be explained by the chosen dynamical model. The erroneous points can be healed via cross-validation [19] and brought back to the main (regular) regions of the data manifold. Extreme events are of the great interest on their own [20, 21, 22, 23, 24]. Such sporadic in time events, can cause severe damaging outcomes in economy and society. At this point, we complete all preprocessing steps.

Next, we can *learn the data manifold* by analyzing *statistical ensembles* of the states in the phase space. From the statistical point of view, we can try to define a probability for the dynamical system to be in one of its many (including infinitely many) states, i.e. we consider an idealization consisting of a number of virtual copies of the system. A statistical ensemble can be depicted as a "cloud" of points in a phase space. Further, one can introduce a probability distribution function $\rho(q, p, t)$ which tells us the probability of finding the dynamical system in one of its numerous states.

Following [31] we are able to analyze complex systems arising in different areas by measuring information by the means of the Shannon's entropy. The necessary preliminary definitions are provided below.

Given the joint probability $\Pr(X, Y)$, we define the *joint entropy* as

$$H(X, Y) = -\sum_{\{X,Y\}} \Pr(X, Y) \log \Pr(X, Y). \tag{1.1}$$

The eq. 1.1 measures the uncertainty (entropy) of the joint event $X \cup Y$.

The *conditional entropy* of Y given X is:

$$H(Y|X) = -\sum_{\{Y,X\}} \Pr(X, Y) \log \Pr(Y|X) = -\sum_{\{Y\}} \Pr(Y|X) \log \Pr(Y|X). \tag{1.2}$$

The eq. 1.1 shows how uncertain is Y given that X is known.

The fundamental measure of correlation between X and Y is the *mutual information*:

$$I(X;Y) = \sum_{\{X,Y\}} \Pr(X,Y) \log \frac{\Pr(X,Y)}{\Pr(X)\Pr(Y)}. \tag{1.3}$$

It is evident from the eq. 1.3 that X and Y are independent whenever $I(X;Y) = 0$.

The information generated at each step of a Markov one-step chain $\{X_t\}, t \geq 0$ defined over a finite set of states $\mathcal{X} = \{1, 2, \ldots, N\}$ is defined by the *entropy rate*, h:

$$h = H(X_{t+1}|X_t) = -\sum_{i=1}^{N} \pi_i \sum_{j=1}^{N} T_{ij} \log_2 T_{ij}, \tag{1.4}$$

where $\Pr(X_{t+1} = j | X_t = i) = T_{ij} \geq 0$, and $\boldsymbol{\pi}$ is the left eigenvector of the transition matrix $\boldsymbol{T}$ corresponding to the largest eigenvalue 1.

The mutual information between the past and the future states of the Markov chain is the *excess entropy*, $\mathcal{D}$ (the downward causation process):

$$\mathcal{D} = I(X_{t+1}|X_t) = H(X_t) - H(X_{t+1}|X_t) = H - h. \tag{1.5}$$

In eq. 1.5 $\mathcal{D}$ measures the strength of long-term structural correlations between blocks of symbols in the Markov chain.

The *mutual information* between the present state of the chain and the future state, $\mathcal{U}$ (the upward causation) is:

$$\begin{aligned} \mathcal{U} &= I(X_t; X_{t+1}|X_{t-1}) = H(X_{t+1}|X_{t-1}) - H(X_t|X_{t-1}) = \\ &= \sum_{i=1}^{N} \pi_i \sum_{j=1}^{N} \left[T_{ij} \log_2 T_{ij} - (T^2)_{ij} \log_2 (T^2)_{ij} \right]. \end{aligned} \tag{1.6}$$

The quantity $H(X_t|X_{t+1}, X_{t-1})$ denoted by $\mathcal{E}$ measures the portion of uncertainty that cannot be predicted, this information component is *ephemeral* that exists only in the present moment:

$$H(X_t|X_{t+1}, X_{t-1}) = \mathcal{E} = h - \mathcal{U}. \tag{1.7}$$

Using the equations above, we are able to decompose the uncertainty of the state in the Markov chains into three independent information components [31], such that

$$H(p) = \mathcal{D}(p) + \mathcal{U}(p) + \mathcal{E}(p)$$

such that

1) $\mathcal{D}(p)$ measures our capability to predict the future state from the past states.

2) $\mathcal{U}(p)$ measures our capability to predict the forthcoming state from the last occurred.

3) $\mathcal{E}(p)$ measures the portion of entropy we are unable to predict.

Each of the above steps can be automated, excluding human being judgments and reducing an expert interference when it comes to data analysis. Data can "understand" itself to some degree.

In Chapter 2 we applied the above strategy to propose computationally feasible statistical algorithms for the automated assessment of isolation and integration of urban locations and neighborhoods by using geo spatial data acquired from the OpenStreeMap service. The dynamical model is founded on Anisotropic Random Walks (ARW). We reported on the patterns of isolation and integration in the city of Lubbock, TX (USA). Our approach may be implemented for the detailed automated expertise illuminating the hidden community structures of any urban pattern and the associated transport networks that may include many transportation modes.

The random walks are used in many applications: modeling stock prices in economics [25, 26], automated assessment of popularity of texts [27], for salient object detection [28], as well as in collaborative filtering, recommender systems, link prediction, computer vision, semi-supervised learning, network embedding, element distinctness [29], etc.

In Chapter 3 we show that inhomogeneous density of states in a discrete model of Standard & Poor's 500 phase space ("Return vs Roughness") leads to inequitable predictability of market events. Most frequent events might be efficiently predicted in the long run as expected from Mean reversion theory. Stocks have different mobility in phase space. Highly mobile stocks are associated with less unsystematic risk.

Less mobile stocks might be cast into disfavor almost indefinitely. Relations between information components in Standard & Poor's 500 phase space resemble of those in unfair coin tossing. We proposed the novel methodology for a quantitative assessment of the amounts of predictable and unpredictable information in any stochastic process whenever an empirical transition matrix between the states observed over a certain time horizon becomes available. The proposed technique is an extension of the famous Ulam's method [62] for approximating invariant densities of dynamical systems.

In Chapter 4 we study extreme events, the events whose magnitude exceeds some threshold. A choice of a threshold is subject to uncertainty caused by a method, the size of available data, a hypothesis on statistics, etc. We assess the degree of uncertainty by the Shannon's entropy calculated on the probability that the threshold changes at any given time. If the amount of data is not sufficient, an observer is in the state of Lewis Carroll's Red Queen who said "When you say hill, I could show you hills, in comparison with which you'd call that a valley". If we have enough data, the uncertainty curve peaks at two values clearly separating the magnitudes of events into three emergency scales: subcritical, critical, and extreme. Our approach to defining the emergency scale is validated by 39 years of Standard and Poors 500 (S&P500) historical data.

Finally, in Chapter 5 we apply the strategy we apply the strategy to the real-life data set provided to us by the Department of Defense while working on the contract W911W6-13-2-0004 "Data Refinement and Reduction for Aviation Sustainment (DRRAS)" that goal was to combine maintenance and fleet data generated by many various sources that are not in agreement with each other.

Introduction of the concepts of dynamical models, phase spaces and data manifolds proved to be extremely effecting in the real-life settings:

1) We developed algorithms for automated separation of flight and non-flight files. Such a functional separation allows for dramatic reduction in the data volume and numbers of files.

2) We created the Baseline time of an aircraft.

3) We integrated the Flight data systems by attributing and synchronizing the VADR and HUMS files. The missing HUMS and VADR files have been detected

at this stage.

4) We then performed the integration of the Flight and Maintenance data systems by synchronizing of aircraft and maintenance events. The missing and functionally merged ULLS files have been detected at this stage.

5) We have developed algorithms for missing and corrupted data restoration via VADR – HUMS – ULLS data cross-validation.

6) After cleaning jerky readings in the Flight and Maintenance data, we developed algorithms for the automated flight stage assessments opening us a way to selectable filtering and reconstructing the flight stage-wise data manifolds for the reliable detection of outliers – the readings that do not belong to the regular data manifolds.

7) Identification of outliers allows, first, to heal them by sending them back to the regular manifolds, and second, developing the Automated Health State Predictive Analytics algorithms for the future Self-Aware Aircrafts.

We conclude in the last section 6.

CHAPTER 2
MULTI-SCALE ANALYSIS OF URBAN SPATIAL STRUCTURES ACQUIRED FROM *OPENSTREETMAP*

2.1 Introduction

Urbanization has been the dominant demographic trend in the entire world during the last half century. Rural to urban migration, international migration, and the re-classification or expansion of existing city boundaries have been among the major reasons for increasing urban population. The essentially fast growth of cities in the last decades urgently calls for a profound insight into the common principles stirring the structure of urban developments all over the world [30, 31].

A belief in the influence of the built environment on humans was common in architectural and urban thinking for centuries [32, 31]. Cities generate more interactions with more people than rural areas because they are central places of trade that benefit those who live there. People moved to cities because they intuitively perceived the advantages of urban life [30]. City residence converted a compact space pattern into a pattern of relationships by constraining mutual proximity between people. Spatial organization of a place has an extremely important effect on the way people moving through spaces and meeting other people by chance [33]. Compact neighborhoods can foster casual social interactions among neighbors, while creating barriers to interaction with people outside a neighborhood. Spatial configuration promotes peoples encounters, as well as making it possible for them to avoid each other, shaping social patterns [34].

In our work, we propose feasible statistical algorithms for the automated assessment of isolation and integration of urban locations and neighborhoods by using maps acquired from the *OpenStreeMap* service, a collaborative project to create a free editable map of the world. Map data in the project is collected from scratch by volunteers performing systematic ground surveys using tools such as a handheld GPS unit, a notebook, digital camera, or a voice recorder. The data is then entered into the OpenStreetMap database being freely available for downloading.

In our approach, isolation of a place in a complex fabric of a city is quantified statistically by the relative first-passage time of isotropic random walks (with no self-

intersections) on the city street map measured in the decibel scale. The proposed *isolation index* helps to detect sprawling, low-density, auto-dependent urban developments in American cities (see Fig. 2.2 representing the isolation pattern in the city of Lubbock wandering across planes of West Texas).

Sociologists think that isolation in sprawling metropolitan areas worsens an area's economic prospects and fuel poverty and crime by reducing opportunities for commerce and engendering a sense of isolation in inhabitants [35, 31]. Suburban sprawl also hurts social mobility by lowering a poor child's chances of moving up the economic ladder as an adult [36]. Although structural isolation defined above can be combated by constructing the new city belt roads that would encompass the isolated regions and cut-off neighborhoods, building new roads just makes the problem worse, as increasing road capacity would lead to more traffic and more sprawl in the future.

The *integration index* of a place is introduced in our work as a relative share of *infinite paths* (assuming all of them are equally probable) hosted by the place in the city street map measured also in the decibel scale. Places hosting the lion shares of infinite paths (existing in the connected graphs even if the graphs are small) play the role of hubs featuring the global mobility patterns in the structure (see Fig. 2.3 representing the integration pattern in the city of Lubbock). By comparing the integration and isolation patterns shown in Fig. 2.2 and Fig. 2.3, we may see that integration and isolation defined above are not always opposites. Some secluded places in the city can be characterized as being in *integrated isolation*, and this wording is more than an oxymoron, as the *wealthiest* people live there.

In the present work, we report on the patterns of isolation and integration in the city of Lubbock, TX (USA). Our approach may be implemented for the detailed automated expertise illuminating the hidden community structures of any urban pattern and the associated transport networks that may include many transportation modes. In Sec. 2.2, we discuss spatial graphs of urban environments, as well as the software libraries used by us for getting the urban spatial graph of the city of Lubbock, TX. In Sec. 2.3, we introduce the major mathematical tool of our analysis, the infinite family of anisotropic scale-dependent random walks (ARW) defined on finite indirected connected graphs. In Sec. 2.4, we discuss the stationary distributions and balance equation for the ARWs. in Sec. 2.5, we describe the family of probabilistic

metrics attributed to ARWs that are used to quantify accessibility of nodes in the spatial graphs and their roles in hosting the path of different lengths. We apply the proposed measures of isolation and integration to the urban street map of the city of Lubbock in Sec. 2.6 We conclude in the last section.

2.2 Spatial graphs of urban environments

In traditional urban researches, the dynamics of an urban pattern come from the landmasses, the physical aggregates of buildings delivering place for people and their activities [37, 30]. The relationships between certain components of the urban texture are often measured along streets and routes considered as edges of a planar graph, while the traffic end points and street junctions are treated as nodes. Such a primary graph representation of urban networks is grounded on relations between junctions through the segments of streets. The usual city map based on Euclidean geometry can be considered as an example of primary city graphs.

In *space syntax theory* [33, 37], built environments are treated as systems of spaces of vision subjected to a configuration analysis. Being irrelevant to the physical distances, spatial graphs representing the urban environments are removed from the physical space. It has been demonstrated in multiple experiments that spatial perception shapes people understanding of how a place is organized and eventually determines the pattern of local movement, [37]. The aim of the space syntax study is to estimate the relative proximity between different locations and to associate these distances to the densities of human activity along the links connecting them [38, 39, 40].

In our work, we use the graph representation of the transportation system of Lubbock acquired from *OpenStreetMap* file located at `https://dataverse.harvard.edu/dataverse/osmnx-street-networks`. We used *Python*'s *lxml* library to parse raw data and to construct an adjacency matrix. The data set was cleaned by removing non connected parts, such as the Preston Smith International airport that is not a structural part of the city. There are 10,421 nodes in the spatial graph representing the city street map of Lubbock, including but not limited to residential, secondary, tertiary roads, trunk links, highways etc. Both *Python's Numpy* library and *Matlab* were used to perform calculations, to avoid possible errors, also *Matlab* was used to draw colored maps representing the isolation and integration patterns in the city.

2.3 Discrete Time Anisotropic Scale-dependent Random Walks

For a finite set V, $|V| = N$, we denote the space of real functions on V as $\mathfrak{F}(V)$. The inner product $(f, g) = \sum_{i \in V} f(i)g(i)$ associates each pair of functions f, g in the space $\mathfrak{F}(V)$ with a scalar quantity. A graph is a collection of ordered pairs $G \subseteq V \times V$ (called *edges*) with respect to a binary relation of adjacency $\smile$ defined on V (called *vertices*). The linear *adjacency operator* $\mathcal{A}$ is defined on all $f \in \mathfrak{F}(V)$ by $(\mathcal{A}f)(i) = \sum_{j \smile i} f(j)$, where $j \smile i$ is whenever i and j are adjacent. The adjacency operator $\mathcal{A}$ can be uniquely represented (with the fixed canonical basis) by a $N \times N-$ matrix $\mathbf{A}$, the *adjacency matrix* of the graph, such that

$$A_{ij} \;=\; \begin{cases} 1, & \text{if } i \text{ and } j \text{ are adjacent,} \\ 0, & \text{otherwise.} \end{cases} \tag{2.1}$$

The adjacency matrix allows us to analyze graphs and their structure by means of the well known operations with matrices. The adjacency matrix is non-negative and irreducible if there is at least a single path from any vertex to any other vertex in the graph. The entries of its positive integer power, A^n_{ij} equal the numbers of walks of length $n \geq 1$ connecting the vertices i and j in the graph G. Let $\mathbf{deg}_n \in \mathbb{Z}^N$ be the vector whose elements $\deg_n(i)$, $i = 1, \ldots, N$, are the numbers of paths of length $n \geq 1$ available from $i \in V$. The adjacency operator defines the following dynamical system [41],

$$\mathbf{deg}_{n+1} \;=\; \mathbf{A}\ \mathbf{deg}_n, \tag{2.2}$$

mapping the vector $\mathbf{deg}_n$ into the vector $\mathbf{deg}_{n+1}$.

The discrete time *locally anisotropic* (i.e., direction dependent) nearest neighbor *random walks* is defined by the following stochastic matrices:

$$T_{ij}(n+1) \;=\; \frac{A_{ij}\ \deg_n(j)}{\deg_{n+1}(i)} \;=\; A_{ij}\,\frac{\sum_{s=1}^{N} A^n_{js}}{\sum_{r=1}^{N} A^{n+1}_{sr}}, \quad n \geq 0. \tag{2.3}$$

Theorem 1. *In the random walk defined by (2.3), all possible paths of length $n \geq 1$ starting at the node $i \in V$ of the graph G are chosen with equal probability.*

Proof. The first order random walk $T(1)_{ij} = A_{ij}/\deg_1(i)$ is *locally isotropic*, since all 1-paths available for the random walker at every vertex of the graph are chosen with

equal probability.

For the random walks of order $n = 2$, the transition probability (2.3) reads as following:

$$T_{ij}(2) \;=\; \frac{A_{ij}\sum_{s=1}^{N} A_{js}}{\sum_{s=1}^{N} A_{is}\sum_{r=1}^{N} A_{sr}}, \tag{2.4}$$

so that each walk of length 2 starting at the node i is chosen with equal probability. Although all walks of length 2 starting at the node i are *equiprobable* under the transition operator (2.4), the probabilities of transition to the nearest neighbors from the node i might be different. The nearest neighbors providing more opportunities for lengthy paths than others are more preferable. The resulting random walk (2.4) is a direction dependent random walk (locally anisotropic). The further conclusion is inductive and self evident. □ □

Theorem 2. *The series of locally anisotropic random walks (2.3) converges as $n \to \infty$ to a random walk, in which all infinitely long paths starting at every node of the graph are equally probable, viz.,*

$$\lim_{n\to\infty} T_{ij}(n) \;=\; T_{ij}(\infty) \;=\; \frac{A_{ij}\psi_{1j}}{\alpha_1\psi_{1i}}, \tag{2.5}$$

where ψ_1 is the completely positive eigenvector of the graph adjacency matrix $\mathbf{A}$ belonging to its maximal eigenvalue α_1.

Proof. The N eigenvalues of the graph adjacency matrix $\mathbf{A}$ are assumed to be ordered, such as $\alpha_1 > \alpha_2 \geq \ldots$. The $n-$th order degree of the node $i \in V$ is

$$\begin{aligned} \deg_n(i) \;&\equiv\; \textstyle\sum_j (A^n)_{ij} \;=\; \sum_k \alpha_k^n \psi_{ik} \underbrace{\sum_j \psi_{kj}}_{\gamma_k} \;\equiv\; \sum_k \alpha_k^n \gamma_k \psi_{ik} \\ &=\; \alpha_1^n \gamma_1 \psi_{i1} \left(1 + \textstyle\sum_{k>1} \left(\frac{\alpha_k}{\alpha_1}\right)^n \frac{\gamma_k}{\gamma_1}\frac{\psi_{kj}}{\psi_{1j}}\right) \simeq_{n\to\infty} \alpha_1^n \gamma_1 \psi_{i1}, \end{aligned} \tag{2.6}$$

since the last sum in (2.6) is dominated by the largest eigenvalue α_1 of the adjacency matrix in the limit $n \gg 1$. Therefore

$$\lim_{n\to\infty} T_{ij}(n) \;=\; \frac{A_{ij}\psi_{j1}\gamma_1\alpha_1^{n-1}}{\psi_{i1}\gamma_1\alpha_1^n} = \frac{A_{ij}\psi_{j1}}{\alpha_1\psi_{i1}}. \tag{2.7}$$

Finally, it is easy to check that the matrix (2.5) is a stochastic matrix, since $\sum_j A_{ij}\psi_{j1} = \alpha_1\psi_{i1}$. □ □

2.4 Stationary Distributions and Balance Equations of Anisotropic Random Walks

For a random walk defined on a connected undirected graph, the Perron - Frobenius theorem [42] asserts the unique strictly positive probability vector $\vec{\pi} = (\pi_1, \ldots, \pi_N)$, which is the left eigenvector of the transition matrix $\mathbf{T}$ belonging to the maximal eigenvalue $\mu = 1$, $\vec{\pi}\,\mathbf{T} = 1 \cdot \vec{\pi}$, is the *stationary distribution* of the random walk on the graph. The condition of *detailed balance* is defined by

$$\pi_i\, T_{ij} \;=\; \pi_j\, T_{ji}, \tag{2.8}$$

from which it follows that a random walk defined on an undirected graph is *time reversible*: it is also a random walk if considered backward.

Theorem 3. *For the anisotropic random walk $T_{ij}(n)$, $n \geq 1$, the stationary distribution is*

$$\pi_i(n) \;=\; \frac{\deg_n(i)\deg_{n-1}(i)}{2E_n}, \quad 2E_n \equiv \sum_i \deg_n(i)\deg_{n-1}(i). \tag{2.9}$$

Proof.

$$\begin{array}{rcl} \sum_i \pi_i(n)\frac{A_{ij}\deg_j(n-1)}{\deg_i(n)} & = & \sum_i \frac{\deg_n(i)\deg_{n-1}(i)}{2E_n}\frac{A_{ij}\deg_{n-1}(j)}{\deg_n(i)} \\ & = & \frac{\deg_n(j)\deg_{n-1}(j)}{2E_n} \;=\; \pi_j(n). \end{array} \tag{2.10}$$

The detailed balance condition is satisfied by (2.9):

$$\begin{array}{rcl} \pi_i(n)T_{ij}(n) & = & \pi_j(n)T_{ji}(n) \\ & = & \frac{A_{ij}\deg_{n-1}(i)\deg_{n-1}(j)}{2E_n}. \end{array} \tag{2.11}$$

□ □

For $n = 1$, the normalization factor $2E_1 = 2E$, where E is the total number of edges in the graph, since $\deg_0(i) = 1$. The stationary distribution equals $\pi_i(1) = \deg_1(i)/2E$, $2E = \sum_i \deg_1(i)$.

Theorem 4 ([43, 44])**.** *For the limiting anisotropic random walk $T_{ij}(\infty)$, the stationary distribution is*

$$\pi_i(\infty) \;=\; \psi_{1i}^2 \tag{2.12}$$

where ψ_1 is the major eigenvector of the graph adjacency matrix $\mathbf{A}$ belonging to the maximal eigenvalue α_1.

Proof.

$$\sum_i \pi_i(\infty)\frac{A_{ij}\psi_{1j}}{\alpha_1\psi_{1i}} \;=\; \psi_{1j}^2 \;=\; \pi_j(\infty). \tag{2.13}$$

The detailed balance condition is satisfied by (2.12):

$$\pi_i(\infty)T_{ij}(\infty) \;=\; \pi_j(\infty)T_{ji}(\infty) \;=\; \frac{A_{ij}\psi_{1i}\psi_{1j}}{\alpha_1}. \tag{2.14}$$

□ □

2.5 Exploring Graphs with Random Walks

The stationary distribution of random walks (2.3) of length $n \geq 1$ defined on a connected undirected graph $G(V, E)$ determines a unique measure on V, $\mu_n = \sum_{j\in V}\pi_i(n)\delta_j$, with respect to which the transition operator (2.3) becomes self-adjoint,

$$\widehat{T_{ij}}(n) \;=\; \left(\hat{\pi}^{1/2}(n)\,\mathbf{T}(n)\,\hat{\pi}^{-1/2}(n)\right)_{ij} \tag{2.15}$$

where $\hat{\pi}(n)$ is the diagonal matrix of the densities of nodes $\pi_i(n)$ (2.9) in the random walk $T_{ij}(n)$.

In the present section, we remove the index n (the length of equally probable walks in (2.3)) from our notations. All formulas and conclusions discussed and derived in this section are equally suitable for any anisotropic random walks.

Diagonalizing the symmetrized transition matrix (2.15), we obtain $\widehat{\mathbf{T}} \;=\; \Psi\,\mathbf{M}\,\Psi^{\top}$, where Ψ is an orthonormal matrix of eigenvectors, $\Psi^{\top} \;=\; \Psi^{-1}$, and $\mathbf{M}$ is a diagonal matrix with entries $1 = \mu_1 > \mu_2 \geq \ldots \geq \mu_N > -1$ (here, we do not consider bipartite graphs, for which $\mu_N = -1$).

1. **The probability to find a random walker in and out of a node:** The Perron-Frobenius eigenvector $\vec{\psi}_1$ belonging to the major eigenvalue $\mu_1 = 1$, $\vec{\psi}_1\widehat{\mathbf{T}} = \vec{\psi}_1$, determines the density of nodes in the graph G visited by the random walks [45, 31],

$$\pi_i \;=\; \psi_{1,i}^2, \quad i = 1, \ldots, N. \tag{2.16}$$

The Euclidean norm of other eigenvectors in the orthogonal complement of $\vec{\psi}_1$, $\sum_{s=2}^{N} \psi_{s,i}^2 = 1 - \pi_i > 0$, is the probability to find a random walker out of $i \in V$. The inverse density of the node i,

$$\mathcal{R}_i \equiv \frac{1}{\psi_{1,i}^2}, \tag{2.17}$$

defines the expected *recurrence time* to the node $i \in V$ [45], characterizing the (local) connection of the node to the graph.

2. **The first-hitting time:** The expected number of steps a *self-avoiding* random walker (never revisiting any visited node) started from the node i needs to reach j for the first time is called the *first-hitting time*, [45, 31]. The first-hitting time satisfies the following equation, $\mathcal{H}_{ij} = 1 + \sum_{i\sim v} \mathcal{H}_{vj} T_{vi}$, withe the boundary condition $\mathcal{H}_{ii} = 0$, reflecting the fact that the first step takes a random walker to a neighbor $v \in V$ of the starting node $i \in V$, and then it has to reach the node j from there. The spectral representation of $\mathcal{H}_{ij}$ [45, 31] is given by

$$\mathcal{H}_{ij} = \sum_{s=2}^{N} \frac{1}{1-\mu_s} \left(\frac{\psi_{s,j}^2}{\psi_{1,j}^2} - \frac{\psi_{s,i}\psi_{s,j}}{\psi_{1,i}\,\psi_{1,j}} \right). \tag{2.18}$$

3. **Random target access time:** The *random target access time* is the expected number of steps required for a self-avoiding random walker to reach a target node chosen randomly in the graph with the stationary density π [45, 31], viz.,

$$\mathcal{T}_G \equiv \sum_{j\in V} \pi_j \mathcal{H}_{ij} = \sum_{s=2}^{N} \frac{1}{1-\mu_s} = \sum_{k=2}^{N} \tau_s, \tag{2.19}$$

where $\tau_s = (1-\mu_s)^{-1}$ is the rate parameter describing how quickly the the diffusion mode s undergoes [31]. The random target access time (2.19) is a global spectral characteristic of the graph independent of the starting node $i \in V$.

4. **First-passage time:** The *first-passage time* is the expected number of steps required for a self-avoiding random walker to reach the node $i \in V$ for the first

time started from a node randomly chosen in the graph with the stationary density π [45, 31], viz.,

$$\mathcal{F}_i \;\equiv\; \sum_{j\,\in\,V} \pi_j\,\mathcal{H}_{ji} \;=\; \sum_{s=2}^{N} \frac{1}{1-\mu_s}\frac{\psi_{s,i}^2}{\psi_{1,i}^2} \;=\; \mathcal{R}_i \sum_{s=2}^{N} \frac{\psi_{s,i}^2}{1-\mu_s}. \tag{2.20}$$

The last equality in (2.20) establishes a relation between the recurrence time to a node $\mathcal{R}_i$ (the local connection of the node) and the first-passage time $\mathcal{F}_i$ characterizing the (global) *connectedness* of the node in the graph. Several models were developed to study the mean first-passage time taken by a walker to move from an arbitrary source to a target in complex media. For instance, such situations were usually encountered in predatory animals and biological cells, [46].

5. **Commute time:** The *commute time* is the expected number of steps required for a self-avoiding random walker started at $i \in V$ to reach $j \in V$ and then to return back to i [45], viz.,

$$\mathcal{K}_{ij} \;\equiv\; \mathcal{H}_{ij} + \mathcal{H}_{ji} \;=\; \sum_{s=2}^{N} \frac{1}{1-\mu_s}\left(\frac{\psi_{s,i}}{\psi_{1,i}} - \frac{\psi_{s,j}}{\psi_{1,j}}\right)^2 \tag{2.21}$$

that introduces the Euclidean metric in the projective space $P\mathbb{R}^{(N-1)}$ of vectors associated to the graph nodes. The average of the commute time with respect to the initial point of random walk, $\sum_i \pi_i \mathcal{K}_{ij} = \mathcal{T}_G + \mathcal{F}_j$, equals the sum of the first passage time to j from a random node in the graph and the random target access time required to reach the random node. The double average of the commute time equals the doubled random target access time, viz., $\sum_{i,j} \pi_i \mathcal{K}_{ij} \pi_j = 2\mathcal{T}_G$.

2.6 The City of Lubbock is Running Away

Railroad paved way for Lubbock to become a hub [47]. After the first train steamed into town on Sept. 25, 1909, Lubbock had managed to quadruple its population to 4,000 in a 10-year period. Once town's dusty roads had been covered with bricks, Lubbock had been selected as the site of the new *Texas Technological College*, and the county's population had ballooned to almost 40,000 in 1930. Lubbock had accom-

plished the journey from town to city when *Loop 289* was completed by the mid-1960s (Fig. 2.1).

Although railway construction enhanced the city status in early days, its maintenance consumed large budgets, was complicated to organize, and had a considerable impact on the urban development of Lubbock. Railways barricade streets, cutting down the number of possible paths people can drive or walk, thus creating neighborhoods isolated from nearby areas. The large urban voids remained in north and east Lubbock cut off from other parts of the city by rails. Not surprising, the city fabric pushed away from the angle formed by the veering railway tracks and grew to the southwest anyhow, running away from the lasso of *Loop 289* marking the old city boundaries (Fig. 2.1).

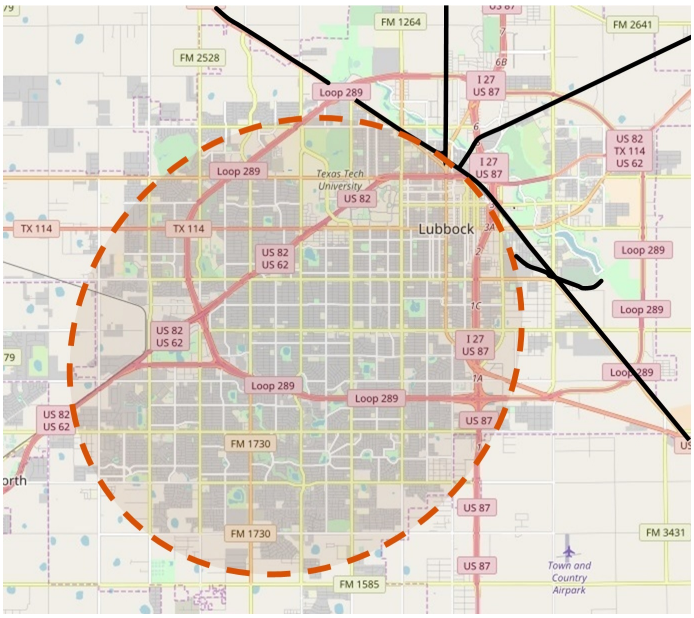


Figure 2.1. The city of Lubbock is running away from the lasso of *Loop 289* marking the old city boundaries.

The Texas Tech University is still anchored in the center and cannot follow the city running away. Situated on the boundary of the neighborhoods going downhill, the University mediates between the isolation and integration patterns in the city of Lubbock (Fig. 2.1).

2.6.1 Isolation index

Structural isolation is viewed as one of the top risk factors for area's economic, social and environmental prospects [48, 49, 50, 51, 52, 53]. The notion of isolation acquires a statistical interpretation by means of random walks [32, 54, 30, 55, 31].

While the recurrence time of random walks to a node (2.17) depends upon the degree of *connectivity* of the node, a *local* structural characteristic of the node in the graph, the first-passage time to the node (2.20) is a *global* structural characteristic of the node quantifying its *connectedness*, since all possible paths to the node existing in the graph are taken into account by the first-passage time although some paths are rendered more probable than others [31]. The relatively low recurrence times are typical for well connected nodes, disregarding their role for the entire graph structure. For example, a bridge connecting the city districts situated on the opposite banks of a river is vital for the entire urban transportation system despite its limited connectivity to the immediate city neighborhoods [30]. The relatively low first-passage times indicate the high degree of importance of the nodes aggregating many a path for structural integrity of the entire graph even if their connectivity is low [31]. Being a global characteristic of a node in the graph, the first-passage time assigns *absolute accessibility scores* (2.20) to all nodes of the graph [32] and can be considered as a natural statistical centrality measure of the node within the graph [30]. The vertices of the urban street graph characterized by the shortest first-passage times are easy to reach by whatever origin-destination (Fig. 2.2). We introduce the *isolation index of a node* in the city street map acquired from the *OpenStreetMap* service by

$$\mathrm{ISL}_i \;=\; 10 \cdot \log_{10} \frac{\mathcal{F}_i(1)}{\min_{j \in V} \; \mathcal{F}_j(1)} \quad (dB), \tag{2.22}$$

in which $\mathcal{F}_i(1)$ is the first- passage time to the node $i \in V$ by isotropic random walks $\mathbf{T}(1)$. Isotropic random walks are insensitive to the structural peculiarities and boundaries of the graph, covering all nodes with respect to their immediate connectivity [31]. The isolation index (2.22) is calculated as relative measure of accessibility and measured in the decibel scale. The high values of the isolation index defined in (2.22) are attributed to the places, which are difficult to access in comparison to those in a structural focus of a city. The spatial pattern of isolation in

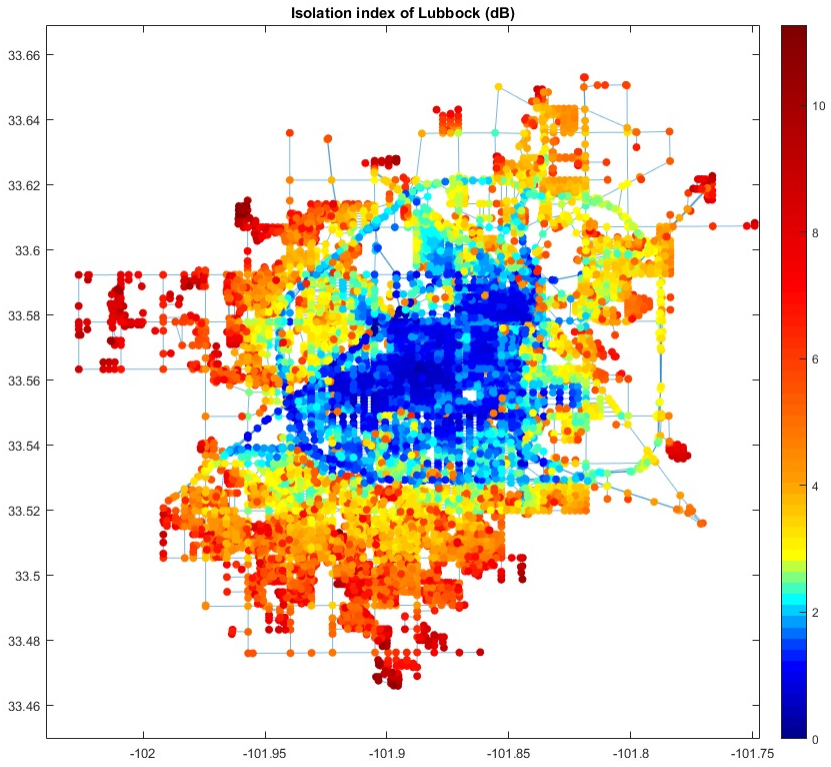


Figure 2.2. Isolation index of nodes in the city graph of Lubbock, TX acquired from the *OpenStreetMap* service. The colored bar indicates the value of isolation index in the decibel scale – the isolated neighborhoods are red colored. The axis labels are the geographic coordinates.

the city of Lubbock (red colored on Fig. 2.2) illuminates the sprawling urban areas homing the low-density communities located out of the old city boundaries. The colored bar in Fig. 2.2 indicates the value of isolation index in the decibel scale. The axis labels are the geographic coordinates.

2.6.2 Integration index

We define the integration index of a node in the graph based on its accessibility by the ARW $\mathbf{T}(\infty)$ introduced by (2.5). This random walks are extremely sensitive to structural defects and graph boundaries [43, 44, 31]. Our definition is based on the idea of that well integrated places should host the majority of infinitely long paths

possible in the spatial graph, eventually determining feasible movement patterns in the city (*see* Fig. 2.3).

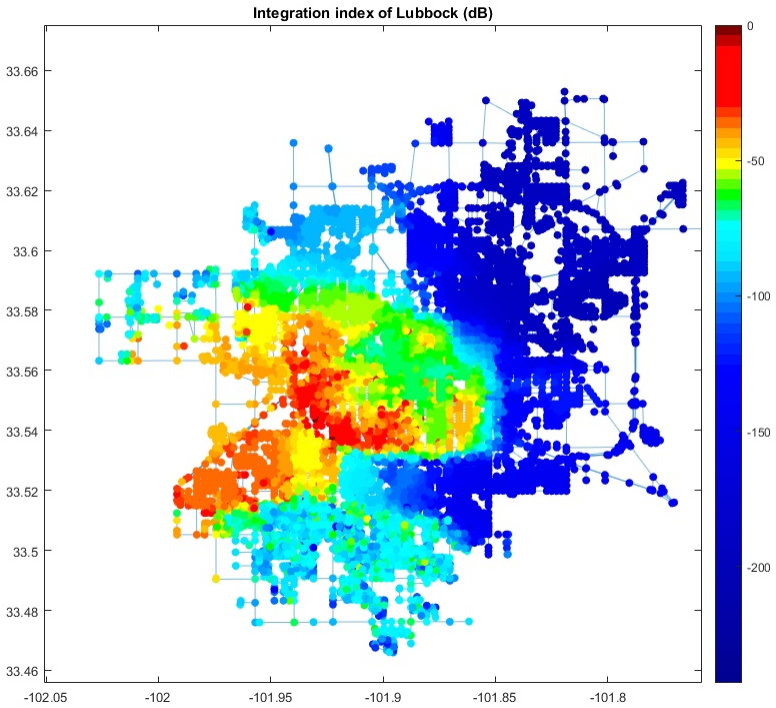


Figure 2.3. Integration index of nodes in the city graph of Lubbock, TX acquired from the *OpenStreetMap* service. The colored bar indicates the value of integration index in the decibel scale – the integrated neighborhoods are red colored. The axis labels are the geographic coordinates.

We introduce the *isolation index of a node* in the city street map acquired from the *OpenStreetMap* service by

$$\mathrm{Int}_i \;=\; 10\cdot\log_{10}\frac{\min_{j\in V}\;\mathcal{F}_j(\infty)}{\mathcal{F}_i(\infty)},\quad (dB) \tag{2.23}$$

in which $\mathcal{F}_i(\infty)$ is the first- passage time to the node $i\in V$ by ARW $\mathbf{T}(\infty)$. The walks $\mathbf{T}(\infty)$ are statistically bounded to the nodes hosting the lion share of infinite paths and repelling from defects and boundaries in the graph. By definition, the index (2.23) is somewhat opposite to the isolation index (2.22), but calculated with respect to anisotropic, direction sensitive walks $\mathbf{T}(\infty)$, not $\mathbf{T}(1)$. The index (2.23) is

measured in the decibel scale, so that places weakly integrated into the city fabric are characterized by large negative numbers. It is evident from Fig. 2.3 that sprawling areas (apart from the northeastern part of the city cut off the downtown by railways dramatically reducing the number of possible infinitely long paths) are able to statistically accumulate essentially long paths and shift the mobility patterns in the city of Lubbock from the downtown to the southwest periphery.

2.7 Discussion and Conclusion

We have proposed the novel concepts and related computationally feasible statistical algorithms for the automated assessment of isolation and integration of the places in the urban street maps acquired directly from the *OpenStreetMap* service. We have introduced the isolation and integration indices of a place in the complex urban fabric based on the first-passage times of isotropic and anisotropic random walks defined on the spatial graph of the city. While the isolation index (2.22) allows for detecting sprawling areas (Fig. 2.3), the integration index (2.23) reveals the areas hosting the essentially long paths in the city graph attributed to the major mobility patterns possible in the urban structure (Fig. 2.2).

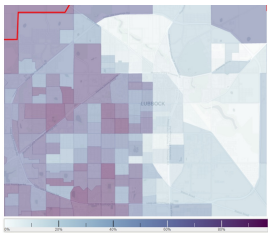
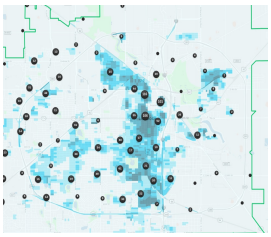
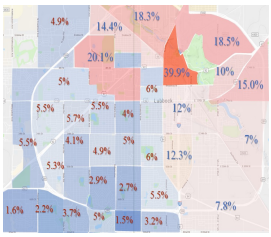
a.)  b.)  c.) 

Figure 2.4. The distribution of social and economic variables in the city of Lubbock, TX. a.) The racial composition of neighborhoods in Lubbock accordingly the US Census Bureau. The area of white population concentration are dark colored; b.) Lubbock crime rates and statistics density heat map (downloaded on 6/29/2018) from the Turlia website [56]. Trulia uses crime reports to provide valuable information on the relative safety of homes in the US; c.) Unemployment Rate in Lubbock, TX in Oct. 2018 accordingly the US Bureau of Labor Statistics.

Encompassing a big share of possible movements, urban sprawl pulls the highly integrated places in the city of Lubbock out of the downtown, fostering the process of urban decay in the north-eastern parts of the city and at the center. In Fig. 2.4,

we have summarized the distributions of social and economic variables over the city of Lubbock, TX. Lack of structural integration in the north-eastern areas of the city (blue colored) reduce opportunities for commerce, fuel poverty and crime, foster disparity of land prices and social inequality, in the long run. The highest crime rates in the city of Lubbock are observed in the northeastern part of the city barricaded by railways, cutting down the number of paths people can drive or walk dramatically.

Interestingly, the isolation pattern in the city of Lubbock assessed by means of the isolation index (2.22) and shown in Fig. 2.2 does not visually match the distributions of the social and economic variables represented on Fig. 2.4. Furthermore, by comparing the integration and isolation patterns shown in Fig. 2.2 and Fig. 2.3, we see that statistically defined integration and isolation are not always opposite, since some secluded places in sprawling areas in the wealthy south-western part of Lubbock can be isolated although integrated.

The further development of the city road system (e.g., the new *Loop 88* that will lasso the running away city of Lubbock in *Wolfforth* by 2030) would alleviate isolation (2.22) by improving accessibility to the south-western sprawls, yet encouraging further sprawl and more road travel and worsening the social and economic conditions in the downtown and on the north-western periphery. The city of Lubbock will keep running southwest.

CHAPTER 3
FIVE YEARS OF PHASE SPACE DYNAMICS OF THE STANDARD & POOR'S 500

3.1 Introduction

In the famous book "*The intelligent investor*", Benjamin Graham had advised not wasting time at forecasting how the market will perform in the future, as markets are fickle and market prices are largely meaningless. Instead, he taught that the intelligent investor should focus at the diligent financial evaluation of company's market value, buying stocks that are clearly underpriced in the market. These opportunities are hard to find, but worth waiting for [57]. Nowadays, we have enough publicly available historical data on stock market prices and the adequate mathematical methods in our hands to confirm or deny the conclusions of the father of value investing and security analysis.

We have investigated the recent five year stock price data for the components of the Standard & Poor's 500 (S&P 500) selected by a committee that assesses the company's market capitalization (must be greater than or equal to $6.1 bln USD), liquidity, domicile, public float, sector classification, financial viability, and length of time publicly traded and stock exchange. The companies selected by the S&P 500 committee are representative of the industries in the United States economy.

The efficient-market hypothesis (EMH) states that asset prices fully reflect all available information [58] though it is impossible to ascertain what a stock should be worth under an efficient market, since investors value stocks differently. Random events are entirely acceptable under an efficient market, and it is unknown how much time prices need to revert to fair value, so that the price remains uncertain at every moment of time and over any time horizon. Therefore, it is important to ask whether EMH undermines itself in its allowance for random occurrences or environmental eventualities. In our work, we suggest a conceptually novel answer resolving this paradox by demonstrating that uncertainty of a stock state in phase space (specified by the stock daily return and its time derivative, roughness) always contain a quantifiable information component that can neither be predicted from any available data (whether historical, new, or hidden "insider" information), nor has any repercussion for the

future stock behavior, but which holds on to the ephemeral present only ("ephemeral information", see Sec. 4). Although the price remains profoundly uncertain, uncertainty does not affect the global ability of financial market to correct itself by instantly changing to reflect new public information.

We study predictability of market states and stock prices of individual companies by reconstructing a discrete model of S&P 500 phase space, in which every component of S&P 500 is characterized by the daily return and roughness of the stock, and the market state corresponds to a certain distribution of stock in phase space.

The novelty of our approach is twofold.

First, in Sec. 4.1, we have developed the novel concept of predictability of states in phase space. Our approach is based on the idea that with some probability a state might be a t-day precursor of another state in phase space. The introduced measure of t-day predictability of a state is a sum of all information components from its t-day precursors. From such a point of view, Markov chains are characterized by the maximum predictability, as any state of a Markov chain is a predictor for any other state of the chain for all t with probability 1. The total amount of predictable information for any state of the Markov chain equals the total information content of the chain.

Second, in Sec. 4.2, we have proposed the novel methodology for a quantitative assessment of the amounts of predictable and unpredictable information in any stochastic process whenever an empirical transition matrix between the states observed over a certain time horizon becomes available. The proposed technique is an extension of the famous Ulam's method [62] for approximating invariant densities of dynamical systems.

The following conclusions can be drawn from the present study:

1. *Long-range forecasts* based on observed stock price time series might be efficient only for the *frequently observed events* in market phase space.

2. *Short-range forecasts* (1-3 days ahead) of rear events (such as *market crushes*) might be efficient *if and only if* the predicted states are resulted from the *processes of exponential decay (growth)* in time.

3. Stocks have different '*mobility*' in phase space. Mobility of a stock is usually

irrelevant to the market capitalization weight of the company. The time series for *highly mobile stocks* might contain more *predictable information* for the efficient forecasting of the future states in phase space than the time series corresponding to the less mobile stocks.

Interestingly, the relations between predictable and unpredictable information in the stock price time series S&P 500, as well as between the amounts of predictable information on the future state of a stock available from the historic time series and from the present state alone are resembling those of *unfair coin tossing*, in which each state repeats itself with the probability $0 \leq p \leq 1$.

3.2 Data source, data validation and preparation

Data on the individual stock prices of S&P 500 constituents for five years from 08/12/2013 to 08/09/2018 (i.e., 1,259 trading days in total) were acquired using *The Investor's Exchange API*, the Python script is available at `https://github.com/CNuge/kaggle-code/blob/master/stock_data`. We have chosen the largest data set publicly available for free and verifiable from different data sources. The data set contains price of the stock at market open, highest price reached in the day, lowest price in the day, price of the stock at market close, number of shares traded labeled by their ticker names for the most of S&P 500 companies during the studied period. We have used individual *stock prices at market close* since this information was always present in the data set. Moreover, in order to preserve consistency of our data, only 468 companies were considered; those selected were present in S&P 500 index throughout the entire observation period, and comprise 98.78% of S&P 500 total market weight. The other 37 constituents were excluded from the study because their presence in the index was inconsistent, and their stock prices were corrupted. The data set has been verified using *Yahoo Finance*. Computations were made using *Python*'s numerical libraries, such as *NumPy* and *Pandas*, *Maple* and *Matlab* to make sure the consistency of computations and eliminate possible errors.

3.3 The discrete model of Standard & Poor's 500 phase space

We use the classical concept of phase space as a collection of all states specified by the observed values of the one-day investment *return* (as a '*position*'):

$$R(t) = \frac{\text{Price}(t+1) - \text{Price}(t)}{\text{Price}(t)}, \tag{3.1}$$

where Price(t) is the price of the stock at *market close*, and t runs over trading days; and its time derivative,

$$\dot{R}(t) = R(t+1) - R(t), \tag{3.2}$$

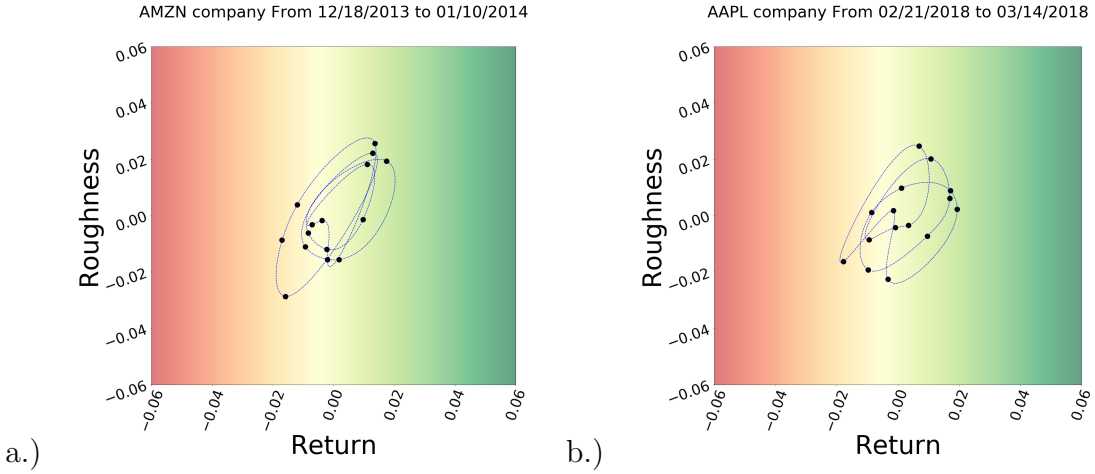
known as *roughness* (as a '*momentum*') [59]. A stock is represented by a point in phase space accordingly its daily return and roughness. The stock's price evolving over time traces a path in phase space (a *trajectory*) representing the set of states in a phase plot (see Fig. 3.1). The state of the market corresponds to a certain distribution and coherent movements of points in phase space.

For the sake of computational feasibility, the 5-year range of return values (minimum -0.53404886 and maximum of 0.3433584) as well as 5-year range of roughness values (minimum -0.59823906 and maximum 0.7059597) were divided into 50 intervals of equal lengths (of 0.01754815 and 0.0260839, respectively). The discrete model of market phase space was reconstructed as the ordered set of 2500 distinct cells, $(r, \dot{r}) \in \{R\} \times \{\dot{R}\}$ where $\{R\}$ and $\{\dot{R}\}$ are the sets of intervals of returns and roughness, respectively.

Our approach to reconstructing market phase space is substantially different from the previous works, in which a phase space was discussed as a metaphor of either all possible market events "where the transfer of rights to real estate takes place and decision-making processes occur" [60], or a collection of recurrence plots containing the information on the systemic trajectory repetition in exchange market times series [61].

3.4 Methods

Methods discussed in the present section can be used for a quantitative assessment of the amounts of predictable and unpredictable information in any stochastic process – whether Markovian or not – whenever an empirical transition matrix between the



a.) b.)

Figure 3.1. The smoothed phase space trajectories (individual stock prices at market close, one observation per day). a.) The Amazon company stocks (from 12/18/2013 to 01/10/2014); b.) The Apple company stocks (from 02/21/2018 to 03/14/2018). The regions of negative return are colored in red; the regions of positive returns are green colored.

states observed over a certain time horizon becomes available. The proposed methodology resembles the rigorous numerical scheme for approximating invariant densities of dynamical systems, Ulam's method [62], and can be adopted to work with diverse, multimodal data sets.

3.4.1 Predictability of stock behavior in market phase space

The degree to which a correct prediction of the future value of a company stock or other financial instrument traded on an exchange can be made is an important open question for an intelligent investor. The stock market is prone to the sudden dramatic declines of stock prices driven by panic as much as by underlying economic factors [63]. Although the behavior of stock prices might be highly uncertain, the amount of uncertainty can be quantified, and the relevant degree of predictability can be assessed from publicly available information.

3.4.1.1 Information content of cells in market phase space

The amount of uncertainty associated to an event is related to the probability distribution of the event. Once the event X has been observed, some amount of un-

certainty is removed, and the relevant amount of information is released. A highly uncertain event X, occurring with small probability $P(X) \ll 1$, contains more information than frequent events. The *information content* of the event X occurring with probability $P(X)$ is

$$J(X) = -\log_2 P(X) \tag{3.3}$$

measured in bits [64]. The information content of tossing a fair coin where the probabilities of heads and tails are $p = 0.5$ equals $-\log_2 0.5 = 1$ bit.

3.4.1.2 T-days mutual information between cells in market phase space

Mutual information measuring the dependence between two random variables had been introduced by Cover and Thomas [65] as a measure of predictability of stochastic states in time,

$$I(t) = \sum_{\{X,Y\}} P(X \xrightarrow{t} Y) \log_2 \frac{P(X \xrightarrow{t} Y)}{P(X)P(Y)}, \tag{3.4}$$

where $P(X \xrightarrow{t} Y)$ is the empirical probability found by dividing the number of times the transition between the X and Y cells occurred precisely in t days by the total number of observed transitions from X, $P(X)$ and $P(Y)$ are the marginal probabilities (of observing the stock in X and Y, independently of each other), and summation is performed over all possible pairs of cells X and Y. If the transition probability $P(X \xrightarrow{t} Y)$ is statistically independent of the cells X and Y, the amount of mutual information associated with such a pair of cells is zero. Mutual information decreases monotonically with time for a stationary Markov process [65].

3.4.1.3 T-days precursors for market events

Bayes' theorem describes the probability of an event, based on prior knowledge of conditions that might be related to the event [66]. Given that a stock is in the phase space elementary cell X, it may be a *t-days precursor* of that in t days the stock will be in the cell Y with the following probability:

$$P_t(X|Y) = \frac{P(X \xrightarrow{t} Y)\ P(X)}{P(Y)} \tag{3.5}$$

where $P(X \xrightarrow{t} Y)$ is the observed probability of transition of a stock from the phase space cell X to Y precisely in t days; $P(X)$, $P(Y)$ are the marginal probabilities of X and Y, respectively. $P_t(X|Y)$ can be interpreted as a *density* of the t-days precursors for the event Y in phase space. If $P_t(X|Y)$ is the same as $P(X)$, then Y is *unpredictable* (i.e., any X is a precursor for Y). The total amount of available information about the forthcoming event Y can be gained from observing all t-day precursors X is measured by the Kullback-Leibler (KL) divergence [65],

$$\mathcal{P}_t(Y) = \sum_{\{X\}} P_t(X|Y) \log_2 \frac{P_t(X|Y)}{P(X)} = \sum_{\{X\}} P(X) \frac{P(X \xrightarrow{t} Y)}{P(Y)} \log_2 \frac{P(X \xrightarrow{t} Y)}{P(Y)}. \quad (3.6)$$

The KL–divergence (3.6) has the form of *relative entropy* [65] that vanishes if and only if the density of the t- day precursors for the event Y in phase space $P_t(X|Y)$ is identical to $P(X)$. This implies that observation of stock in X is statistically independent of its observation in Y t days later, i.e. X is not a precursor for Y. Relative entropy was proposed as a measure of predictability in [68, 67], and we consider the relative entropy (3.6) as a *measure of predictability* for the forthcoming event Y t days ahead.

If the discussed process was Markovian, the marginal probability $P(X)$ is the major eigenvector of the transition matrix $P(X \xrightarrow{t} Y)$, and the probability (3.5) is $P_t(X|Y) = 1$, for all t and any Y. For a Markov chain, any state X is a predictor for any other state Y for all t with probability 1 (3.5). Then the total amount of predictable information (3.6) for any state Y of the Markov chain equals

$$\mathcal{P}_{\mathrm{M}} \;=\; -\sum_{\{X\}} \log_2 P(X) \;=\; \sum_{\{X\}} J(X), \quad (3.7)$$

the total information content of the chain. Markovian processes are characterized by the maximum degree of predictability.

3.4.2 Predictable and unpredictable information estimated from an empirical transition matrix

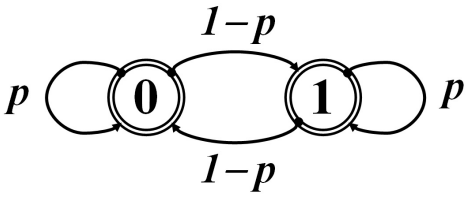
The empirical transition probability $P(X \xrightarrow{t} Y)$ constitutes a finite state discrete Markov chain in S&P 500 phase space. Every bit of information characterizing un-

certainty of a state in a Markov chain consists of three independent information components,

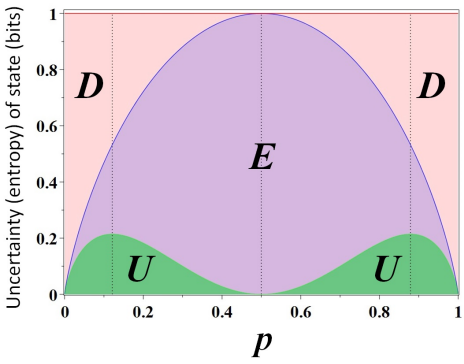
$$H \;=\; \mathcal{D} + \mathcal{U} + \mathcal{E}, \tag{3.8}$$

where $\mathcal{D}$ ("*downward causation*" [69]) characterizes our capability to predict the forthcoming state of the chain from the past states, $\mathcal{U}$ ("*upward causation*" [69]) characterizes our capability to guess the future state from the present one, and the amount of information $\mathcal{E}$ ("*ephemeral information*" [70]) can neither be predicted from the past, nor it's repercussions can be traced in the future.

For example, tossing a unfair coin, in which each state ('*heads*' or '*tails*') repeats itself with the probability $0 \leq p \leq 1$, is described by the Markov chain with the transition probabilities $\mathbf{T}(p) = \begin{pmatrix} p & 1-p \\ 1-p & p \end{pmatrix}$ (see Fig. (3.2. a)). For $p = 1$ and $p = 0$, the Markov chain generates the constant sequences of symbols, viz., $\ldots 0, 1, 0, 1, 0,$ (for $p = 0$), and $\ldots 1, 1, 1, 1, 1,$ or $\ldots 0, 0, 0, 0, 0$ (for $p = 1$). When $p = 1/2$, the Markov chain Fig. 3.2.a represents tossing a fair coin. The densities of



a.)



b.)

Figure 3.2. a.) The state diagram for tossing a unfair coin, in which each state ('*heads*' or '*tails*') repeats itself with the probability $0 \leq p \leq 1$. b.) The information components $\mathcal{D}(p)$, the past-future mutual information (excess entropy [70]); $\mathcal{U}(p)$, the conditional mutual information available at the present state of the chain and relevant to the future states; $\mathcal{E}(p)$, the ephemeral information existing only in the present state of the chain, being neither a consequence of the past, nor of consequence for the future.

both states are equal, $\pi = [1/2, 1/2]$, so that throwing a unfair coin to choose between '*heads*' and '*tail*' reveals a single bit of information as quantified by *Shannon's entropy*,

$H(X_t) = -\sum_{i=1}^{2} \pi_i \log_2 \pi_i = -\log_2 \frac{1}{2} = 1$, for any value of p [69].

The information associated to the *downward causation* process,

$$\begin{aligned} \mathcal{D}(p) \equiv H(X_t) - H\left(X_{t+1}|X_t\right) &= -\sum_{i=1}^{2} \pi_i \left(\log_2 \pi_i - \sum_{j=1}^{2} T_{ij} \log_2 T_{ij} \right) \\ &= -p \log_2 p - (1-p) \log_2 (1-p), \end{aligned} \tag{3.9}$$

quantifies the amount of uncertainty that is released from observation of the sequence of past states of the chain. When $p = 0$ or $p = 1$, the series is stationary, and the forthcoming state is determined by observing the past states, so that $\mathcal{D}(0) = \mathcal{D}(1) = 1$ bit. When tossing a fair coin ($p = 1/2$), the information component (3.9) vanishes ($\mathcal{D}(1/2) = 0$), and the forecast of a future state by observing the historical sequence of symbols loses any predictive power (see Fig. (3.2. b)).

Although our capability to predict the forthcoming state by observing the historical sequences weakens as $p > 0$ or $p < 1$, we can now guess the future state of the chain directly from the presently observed symbol (by alternating / repeating the current symbol with the probability $p > 0$, or $p < 1$, respectively). The related information component ('*upward causation*') quantifying the goodness of such a guess is the mutual information between the present and future states of the chain conditioned on the past [69], $\mathcal{U} = H\left(X_{t+1}|X_{t-1}\right) - H\left(X_{t+1}|X_{t-1}\right)$, viz.,

$$\begin{aligned} \mathcal{U}(p) &= \sum_{i=1}^{2} \pi_i \sum_{j=1}^{2} \left(T_{ij} \log_2 T_{ij} - (T^2)_{ij} \log_2 (T^2)_{ij} \right) \\ &= p \log_2 p + (1-p) \log_2 (1-p) - 2p(1-p) \log_2 2p(1-p) \\ &\quad - (p^2 + (1-p)^2) \log_2 \left(p^2 + (1-p)^2\right). \end{aligned} \tag{3.10}$$

Our capability to predict the future symbol from the present alone (as quantified by the mutual information (3.10)) increases as $p > 0$ ($p < 1$) till $p \approx 0.121$ ($p \approx 0.879$) when the effect of *destructive interference* between the obviously incompatible guesses on alternating the current symbol at the next step (for $p > 0$) and on repeating the current symbol (for $p < 1$) causes the attenuation and complete cancellation of this information component in the case of fair coin tossing ($\mathcal{U}(1/2) = 0$) (see Fig. (3.2. b)).

The remaining conditional entropy, $\mathcal{E} \equiv H\left(X_t|X_{t+1}, X_{t-1}\right)$, quantifies the portion

of uncertainty that can neither be predicted from the historical data, nor having repercussions for the future, viz.,

$$\begin{aligned}\mathcal{E}(p) = &-2p\log_2 p - 2(1-p)\log_2(1-p) + 2p(1-p)\log_2 2p(1-p) \\ &+ \left(p^2 + (1-p)^2\right)\log_2\left(p^2 + (1-p)^2\right).\end{aligned} \tag{3.11}$$

Since all information shared between the past and future states in Markov chains goes only through the present, the mutual information between the past states of the chain and its future states conditioned on the present moment is always trivial: $I(X_{t-1}; X_{t+1}|X_t) = 0$ [70, 69].

3.5 Results

We can gain insight into predictability and the related amounts of predictable and unpredictable information in the market events and daily stock prices of individual companies by analyzing the empirical t-days transition matrices $P(X \xrightarrow{t} Y)$ and the density of states $P(X)$ in market phase space.

3.5.1 Visualizing the stock market crashes, rallies, and market tumbling on shocking events

The state of the US economy can be visualized by daily snapshots of the S&P 500 stocks in phase space as displayed in Fig. 3.3. Standard & Poor's calculates the market capitalization weights (currently ranging from 0.00753 for the National Weather Service to 3.963351 for the Apple company) using the number of shares available for public trading. Movements in the prices of stocks with higher market capitalization (the share price times the number of shares outstanding) had a greater impact on the value of the index than do companies with smaller market caps. In Fig. 3.3, we indicate the capitalization weights of companies by the radii of circles and jet-colors for convenience of the reader.

Market crashes reveal themselves as a dramatic decline of stock prices across the market when the most of stocks appear in the 'red zone' of phase space (Fig. 3.3.a) characterized by the negative return and often negative roughness values, resulting in a significant loss of paper wealth. During the periods of sustained increases in the prices of stocks (i.e., *market rallies*), the stocks overwhelmingly move to the 'green

zone' of phase space, with positive returns and often positive roughness (Fig. 3.3.b).

Market tumbling on shocking events constitutes a 2-day synchronization phenomenon (Fig. 3.3.c & e; d & f). On the first day, many stocks across the market get synchronized on the zero–return value that becomes visible as a vertical line emerging on the phase space snapshots (Fig. 3.3.c & d). On the second day, the stock market would become essentially volatile, plunging and snapping back due to coherent sells-off / ramping of stocks that might be affected by the announced news. Indeed, the stock drops may result in the rise of stock prices for corporations competing against the affected corporations. Interestingly, a certain degree of coherence in the apparently volatile market movements might persist for a day as it seen from the continued alignment of stocks in phase space visible on the second day of the market recovery process (Fig. 3.3.e & f).

The first tumbling event shown on Fig. 3.3.c& e had occurred on Thursday, 07/18/2014, when a Malaysia Airlines plane (flight MH17) headed from Amsterdam to Kuala Lumpur carrying 298 passengers had been shot down by a Russian military unit invading eastern Ukraine. The second tumbling is observed on August $4, 2014$ (see Fig. 3.3. d & f) at the coincidence of alarming events, including Argentina's default on bond payments, more sanctions against Russia backed by EU and US in response to the downing of flight MH17, and the Islamic State seizure of fifth Iraqi oil field. Within that week, the S&P 500 lost 2.69%, the Dow fell 2.75%, and the Nasdaq slid 2.18%.

3.5.2 Phase portrait , t–days predictability of states, and t–days mutual information in S&P 500 phase space.

Not all cells in market phase space are visited equally often by stock prices. The central region, about the *zero-return zero-roughness* equilibrium point was the most visited of all (see Fig. 3.4.a) during five years of observations. This observation is in line with the *Mean reversion* theory suggesting that asset prices and returns eventually *return back* to the long-run mean or average of the entire data set [71]. Consequently, the information content is minimal for the frequently visited cells located in the central region, but is high for the periphery cells rarely visited by stocks (Fig. 3.4.b). The phase space cells corresponding to *market crushes* are characterized

by the *maximum* information content. Uneven attendance of cells in S&P 500 phase space results in the inequitable degree of predictability of the corresponding events.

The predictability of states calculated as the KL–divergence (3.6) between the density of t- days precursors for an event and the marginal density of states $P(X)$ across phase space changes with time (see Fig. 3.4.c & d). Summarizing on *short-range predictions* based on observation of all possible 1–day precursors for every cell in phase space, we obtain (see Fig. 3.4.c) that the *most predictable* states are aligned along the '*main diagonal*' ($R \propto \dot{R}$) of the phase portrait in S&P 500 phase space, roughly corresponding to the *exponential growth (decay)* processes with a single *unstable equilibrium* at the zero-return zero-roughness point $(0, 0)$. The states out of the 'main diagonal' shown in Fig. 3.4.c cannot be predicted efficiently.

Long-range forecasts predicting the behavior of stocks for more than 3 days in advance follow the different predictability pattern shown in Fig. 3.4.d. The efficiently predictable states for long-range forecasts are located overwhelmingly in the central region of the phase portrait, about the zero-return zero-roughness unstable equilibrium point. In Fig. 3.4.d, we have presented the predictability pattern based on 24-day precursors. This observation also supports the general wisdom of the Mean reversion theory [71]. Long-range predictions of infrequent events (such as market crushes) situated on the periphery of the phase diagram should be seen as a rough guide, as the accuracy of such predictions falls considerably around the few days mark.

The decay rate of time-dependent mutual information is an important characteristic of the *fluctuations of information flow* in a complex system [72]. In the case of small fluctuations of information flow, the time– dependent mutual information decays linearly in time; but the decay might be slower than linear, in the case of the large fluctuations of the information flow in the system [72]. Mutual information decays monotonically with time for Markov processes [65]. In formal languages, the mutual information between two symbols, as a function of the number of symbols between the two, decays exponentially in any probabilistic regular grammar, but can decay like a power law for a context-free grammar [73].

It is worth mentioning that the time decay of mutual information in time (3.4) calculated over S&P 500 phase space is very heterogeneous and even *non-monotonous*.

The value of mutual information plunges within the first three days to approximately 7 Kbits and then gets stuck there, at least up to 100-day's mark (see fig. 3.5). The observed non-monotonous decay of mutual information may indicate the presence of the long-lasting large fluctuations of information flow that might be attributed to *stock market bubbles* resulted from *groupthink* and *herd behavior* [74].

3.5.3 Predictable and unpredictable information in company stock prices. Stocks of different mobility

An intelligent investor is interested in diversification of the portfolio across stocks to reduce the amount of *unsystematic risk* of each security. Unsystematic risk (related to the occurrence of desirable events) might be associated to the high shares of unpredictable (ephemeral) information in daily stock prices. To quantify the risk levels in market phase space, we have investigated the shares of predictable and unpredictable information in the transitions of stocks between the cells in phase space for all 468 studied companies from the S&P 500 (Fig. 3.6). Below, we report the results obtained for 1-day transitions of company stock prices, planning the detailed report on the long-term transitions for the future publications.

We have analyzed the empirical 1-day transition matrices for the stocks of individual companies from the S&P 500 list observed in the last five years. Every transition matrix gives us three values (D, U, E) quantifying the predictable and unpredictable information components in uncertainty of a stock state with respect to one-day transitions. In Sec.4.2, we discussed that the transition matrices for unfair coin tossing can be parameterized by the state repetition probability, so that for every value of $0 \leq p \leq 1$, we get a triplet $(D, U, E)(p)$ shown in Fig. 3.2. While working with the S&P 500 data, we have parameterized the (D, U, E) – triplets by uncertainty of daily stock price assessed by its Shannon entropy H producing the curves shown in Fig. 3.7.

The functional analogy between the state repetition probability $0 \leq p \leq 1$ for tossing an unfair coin and uncertainty of daily stock price of a company can be intuitively understood in terms of stock "mobility" in phase space. The abundance of trajectory patterns pertinent to highly mobile stocks provide more valuable information for the efficient prediction of the future states. On the contrary, from the forecasting perspective, the behavior of low mobile stocks is prone to high unsystematic risk of

unpredictable events like tossing a fair coin.

The empirically observed relations shown in Fig. 3.6 demonstrate that the amount of predictable information in daily transition of stocks ($\mathcal{D}$+$\mathcal{U}$) grows while the amount of unpredictable information ($\mathcal{E}$) decays monotonously with Shannon's entropy H measuring uncertainty of the daily stock price for every studied company. Unpredictable information associated to unsystematic risk is *maximum* (exceeding the both predictable information components) for the companies characterized by the *lowest entropy values*. In contrast to it, the amount of unpredictable information practically vanishes for the companies characterized by the *highest uncertainty* of daily stock prices, which can be mostly resolved by observation the historic time series as indicated by the dominance of the information component $\mathcal{D}$ (see Fig. 3.7).

Although entropy is commonly associated with the amount of disorder, or chaos in a system, our conclusion does not appear as a paradox judged from the everyday perspective. Stocks have different '*mobility*' in phase space as quantified by inequitable uncertainty of daily return (Shannon's entropy) H in market phase space. In statistics, entropy is related to the *abundance* of '*microscopic configurations*' (i.e, trajectories of stocks updated daily) that are consistent with the observed '*macroscopic quantities*' (i.e., the density of cells obtained over the whole period of observation) [75]. Highly mobile stocks increase the degree to which the probability of visiting a cell is spread out over different possible trajectories in market phase space, providing more data valuable for the efficient forecasting the future states; the more such trajectories are available to the stock with appreciable probability, the greater it's entropy. On the contrary, stocks of low mobility characterized by the low levels of Shannon's entropy do not accumulate enough data for an efficient predictions of the future states.

The shares of information components in complex systems are naturally related to each other. The amount of unpredictable information decreases when the amount of predictable information increases; information predictable from the present state of a system alone is minimum when the future state can be determined from observation of the historic data.

Interestingly, there are remarkable similarities between the information relations observed across the S&P 500 stock market and in unfair coin tossing Fig. (3.2. a)

discussed by us in Sec. 3.4.2. In Fig. 3.8.a.) and b.), we have presented the relations between the amounts of unpredictable ($\mathcal{E}$) and predictable information ($\mathcal{D}+\mathcal{U}$) and between $\mathcal{D}$ and $\mathcal{U}$ in unfair coin tossing: these relations are linear and parabolic, respectively. The corresponding relations between the information components observed across the S& P 500 stock market are shown in Fig. 3.8.c.) and d.).

Following the analogy between the behavior of stock in S&P 500 phase space and tossing a unfair coin, we may say that stocks highly mobile in phase space characterized by the high entropy values correspond to the marginal values $p \gtrsim 0$, or $p \lesssim 1$ for the probability to repeat a symbol in unfair coin tossing. In both cases, the abundance of patterns in symbolic time series provide more valuable information for the efficient prediction of the future states. On the contrary, from the forecasting perspective, the behavior of 'low mobile' stocks prone to high unsystematic risk of undesirable events is somewhat similar to tossing a fair coin.

3.6 Discussion and Conclusion

We have reconstructed the discrete model of S&P 500 phase space and studied predictability of the cells and individual S&P 500 constituents. The certain distributions and coherent movements of points in phase space are associated to the particular states of the market.

Inhomogeneous density of states in S&P 500 phase space results in their inequitable predictability: more frequent market events are predicted more efficiently than relatively rare events, especially in the long-run. The heterogeneous and even non-monotonous decay of mutual information in time might be attributed to strong fluctuation of information flow across S&P 500 phase space arisen due to stock market bubbles.

Mobility of a stock in phase space can be quantified by entropy measuring uncertainty of the daily return of the company. Highly mobile stocks characterized by the high entropy values provide more data valuable for the efficient forecasting the future states and therefore are associated to lower unsystematic risk than the stocks of 'low mobility'. High stock mobility is usually irrelevant to the market capitalization weight of the company. *WEC Energy Group Inc.*, providing electricity and natural gas to 4.4 mln customers across four states through its brands, is characterized by

the highest value of entropy ($H_{\mathrm{WEC}} = 8.377$ bit) among all S&P 500 constituents. *Entergy Corporation*, an integrated energy company engaged in electric power production and retail electric distribution operations, and the *CMS Energy Corporation* share the second and third places with $H_{\mathrm{WEC}} = 8.352$ bit and $H_{\mathrm{CMS}} = 8.279$ bit, respectively. For comparison with *Apple Inc.*, the leader of the S&P 500 market capitalization weight (3.963351), it's on the 120th place with $H_{\mathrm{AAPL}} = 7.5526$ bit.

We confirm a great deal of wisdom shared with the common investors by Benjamin Graham in his famous book [57]. Reliable long-range predictions of rare events, such as market crushes, are hardly possible. Companies that offer a large margin of safety are not always those of the highest S&P 500 market capitalization weight. In the last five years, the companies of low unsystematic risk are found in the energy industry, forming the bedrock of our society.

We believe that our work would help to resolve a logical paradox related to EMH on that if no investor had any clear advantage over another, would there be investors who have consistently beat the market, such as Benjamin Graham and his disciple, Warren Buffett?

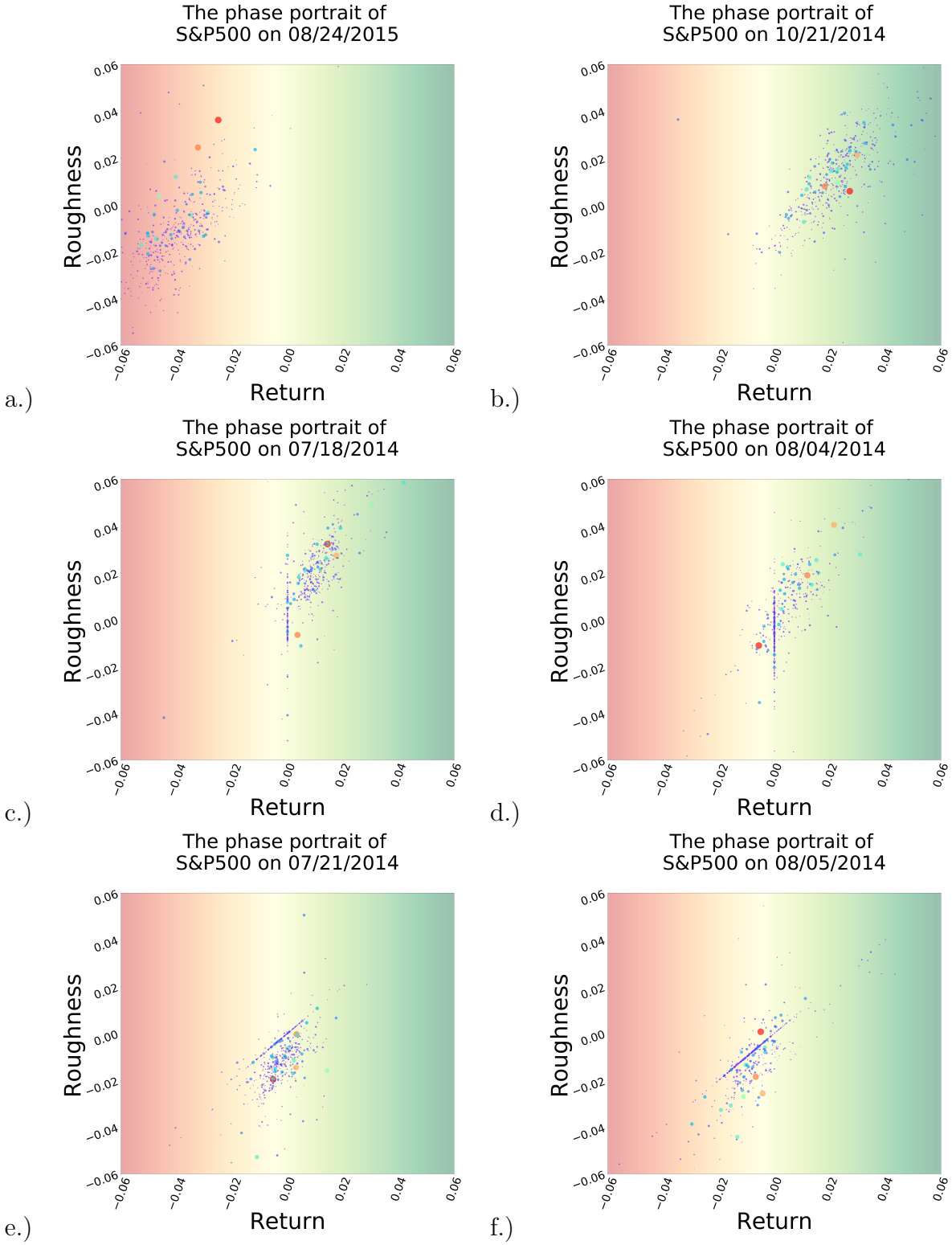


Figure 3.3. The daily snapshots of S&P 500 stocks in phase space during a.) a stock market crush; b.) a stock market rally; c.) & e.) and d.) & f.) a market tumbling phenomenon. The regions of phase space characterized by the positive/ negative values of return are colored by 'green' and 'red', respectively).

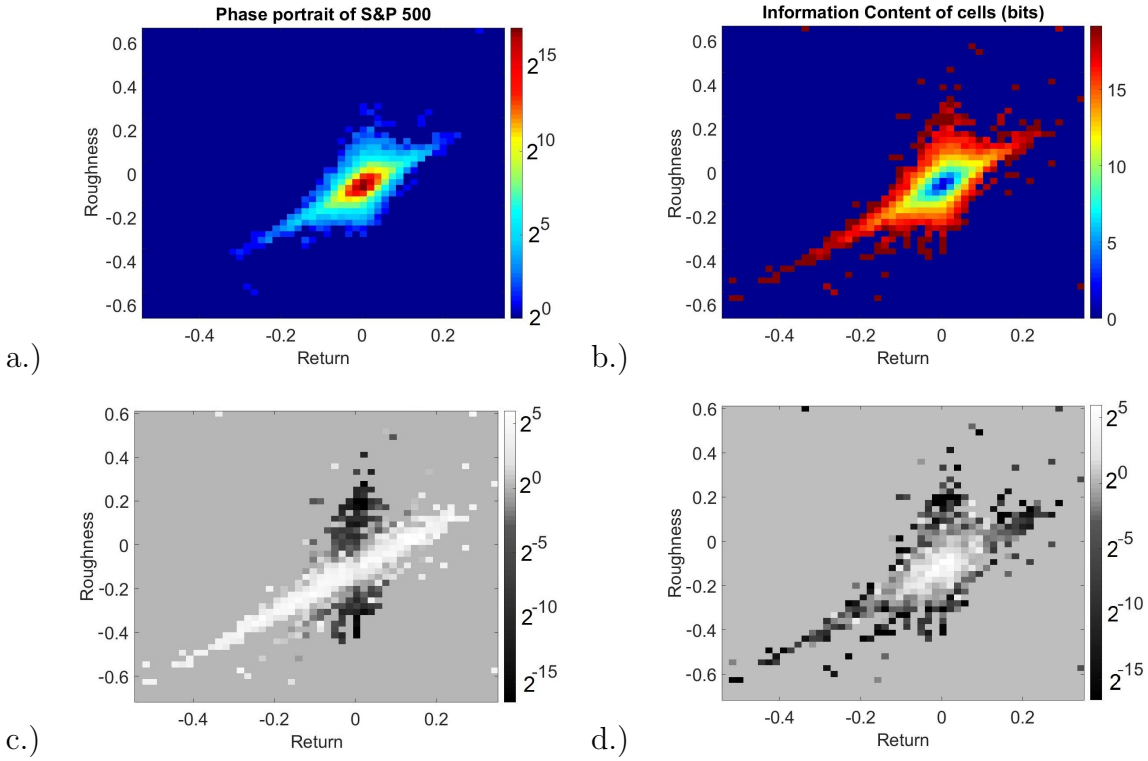


Figure 3.4. a.) The color - coded histogram indicating the number of visits of the phase space cells throughout the entire period of observations; b.) The color - coded histogram representing the information content of phase space cells; c.) The predictability of states based on 1-day precursors; d.) The predictability of states based on 24-day precursors.

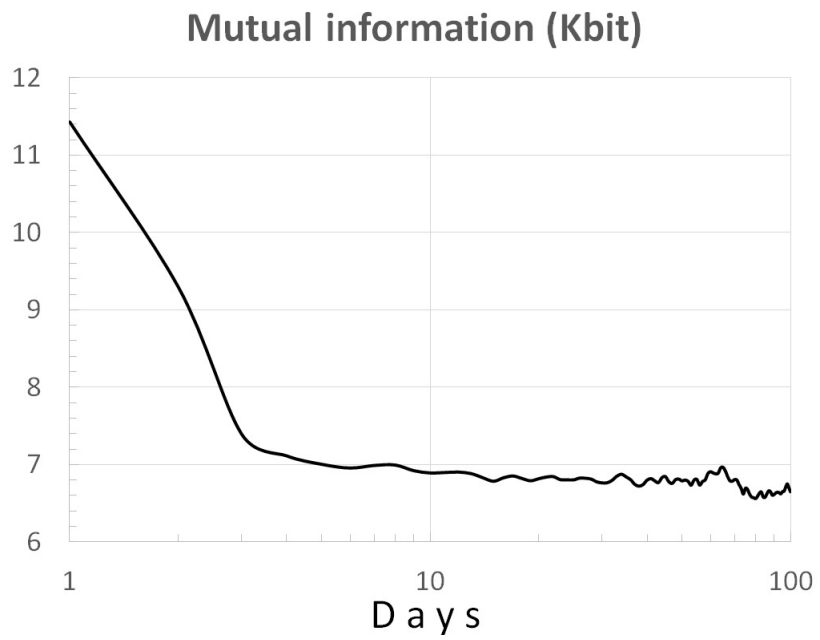


Figure 3.5. T–days mutual information of S&P 500.

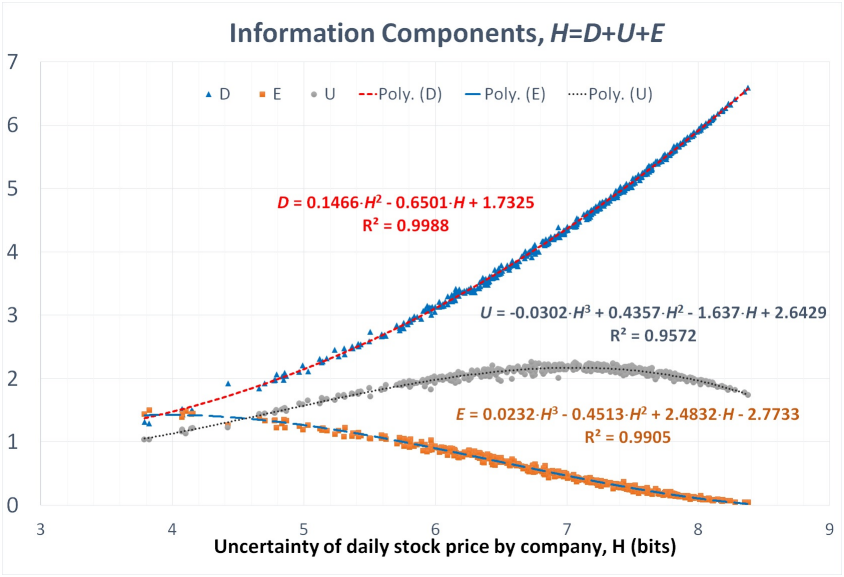


Figure 3.6. The shares of transition information predictable from the historic time series ($\mathcal{D}$), from the present observation ($\mathcal{U}$), and ephemeral information ($\mathcal{E}$) vs. Shannon's entropy H measuring uncertainty of daily return for every studied company. The lines represent the polynomial trends for the empirically observed relations.

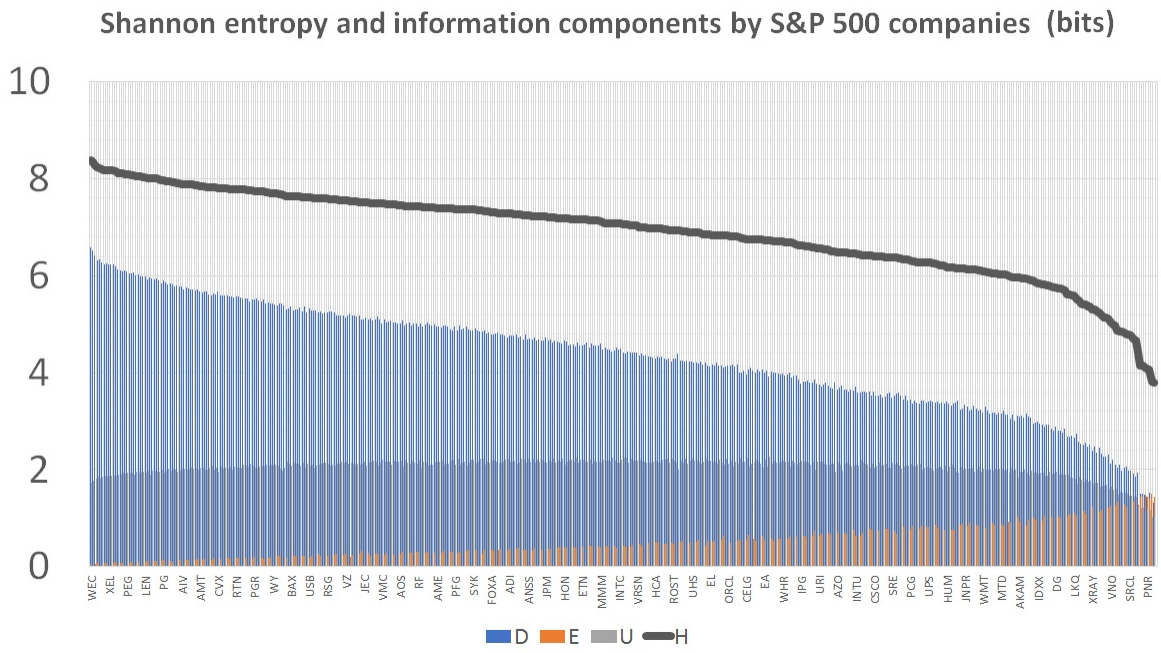


Figure 3.7. Uncertainty of daily stock price measured by Shannon's entropy H along with the information components, $\mathcal{D}$, $\mathcal{U}$ and $\mathcal{E}$, for 468 major companies from the S&P 500.

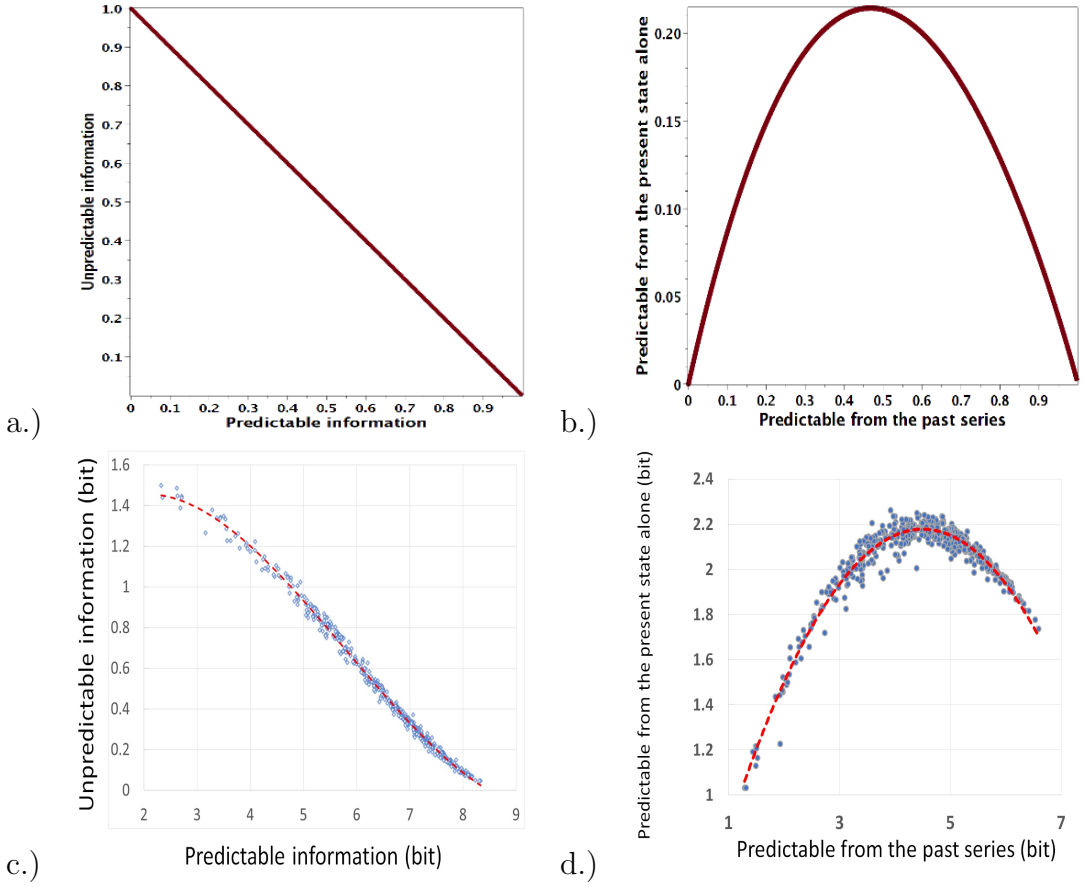


Figure 3.8. A "unfair coin" of the S& P 500 stock market. a.) Unpredictable ($\mathcal{E}$) vs. predictable information ($\mathcal{D}+\mathcal{U}$) in unfair coin tossing Fig. (3.2. a); b.) Predictable information available from the present state alone ($\mathcal{U}$) vs. predictable information available from the past series ($\mathcal{D}$) in unfair coin tossing Fig. (3.2. a); c.) Unpredictable ($\mathcal{E}$) vs. predictable information ($\mathcal{D}+\mathcal{U}$) in the S& P 500; d.) Predictable information available from the present state alone ($\mathcal{U}$) vs. predictable information available from the past series ($\mathcal{D}$) in the S& P 500.

CHAPTER 4
EXTREME EVENTS AND EMERGENCY SCALES

4.1 Introduction

Not a single day passes by without hearing about extreme events which surround us almost everywhere. On one hand, climate change results in droughts, heat waves, tornadoes, storms etc; movement of the tectonic plates is responsible for earthquakes and volcano eruptions. On the other hand, certain aspects of human activities may crash stock markets, influence tensions not only among groups of people in a country, but also between countries sometimes leading to military confrontations and mass migrations, etc. While there is no single definition of extreme events [76], they are considered events that cause infrastructure failures, economic and property losses, risk to health and life.

In order to quantify extreme events, practitioners developed several scales. For instance, Modified Mercalli Intensity scale [77] (describes the intensity of visible damage of earthquakes), Beaufort Wind scale [78] (measures speed and observed effects the wind has on the sea and land), Saffir-Simpsons Hurricane scale [79] (measures wind speed), Fujita scale [80] (rates the intensity of a tornado after it has passed), US Homeland Security Terror Alert scale [81] (measures five color-coded terror alert levels), U.S. Climate Extremes Index [82], etc. Rohn Emergency Scale [83] unites emergency scales using three independent dimensions: (i) scope; (ii) topographical change (or lack thereof); and (iii) speed of change. The intersection of the three dimensions provides a detailed scale for defining any emergency [83]. In some papers, the threshold for an extreme event is related to the number of standard deviations from the average amplitude [84, 85].

However, existing empirical scales tend to describe the characteristics of the event itself rather that the consequences; such scales are ill-suited to describe emergencies in a way that is meaningful for response [86]. For example, the severity of earthquake shaking (in terms of the amount of seismic energy released) is measured using the Richter magnitude scale that ranges from 1.0 (micro-earthquakes, not felt of felt rarely happening several million times per year) to 9.0 (at or near total destruction causing permanent changes in ground topography, and happening one per 10 to 50 years) [87].

At the same time, a projection of an earthquake on the Richter magnitude scale may not be an accurate scale of the disaster. The 2010 Haiti earthquake of a catastrophic magnitude 7.0 caused from $100,000$ to $160,000$ deaths and created between \$7.8 billion and \$8.5 billion dollars in damage [88]. Later on July 5, 2019 there was a magnitude 7.1 earthquake near Southern California's Searles Valley. Estimated losses are at least \$1 billion dollars. No deaths were reported related to this catastrophe [89]. Hence, the magnitude of the earthquakes is not equivalent to the value of their consequences.

We also witness extreme events of different magnitudes in financial markets from global recessions (defined by a global annual GPD growth rate of 3.0 percent of less) that happened in 1975, 1982, 1991, 2009 to "flash crashes" (for instance, on May 6, 2010, the S&P500 declined 7% in less than 15 minutes, and then quickly rebounded). Unlike the Richter magnitude scale, the severity of extreme events in financial markets is defined by the measures that need to be taken to ease panic such as halting trading. Under 2012 rules, market-wide circuit breakers (or 'curbs') kick in when the S&P 500 index drops 7% for Level 1; 13% for Level 2; and 20% for Level 3 from the prior day's close. A market decline that triggers a Level 1 or 2 circuit breaker before 3:25 p.m. Eastern Time will halt trading for 15 minutes, but will not halt trading at or after 3:25 p.m. Circuit breakers can also be imposed on single stocks as opposed to the whole market. Under current rules, a trading halt on an individual security is placed into effect if there is a 10% change in value of a security that is a member of the S&P 500 Index within a 5-minute time frame, 30% change in value of a security whose price is equal or greater than \$1 per share, and 50% change in value of a security whose price is less than \$1 per share [90]. Ironically, in August 2015, single stock circuit breakers produced unprecedented disruption as 327 exchange-traded funds experienced more than 1000 trading halts during a single day.

Statistics of extreme events show that the extreme events are found in the tails of probability distributions (i.e. the distributions extremities). But different distributions may admit asymptotic tails of the same appearance (a power law). In our paper we apply two approaches existing in practical extreme value analysis to study S&P 500 time series in the period from January 2, 1980 till December 31, 2018. The first one relies on deriving block maxima series as a preliminary step. We fit the actual

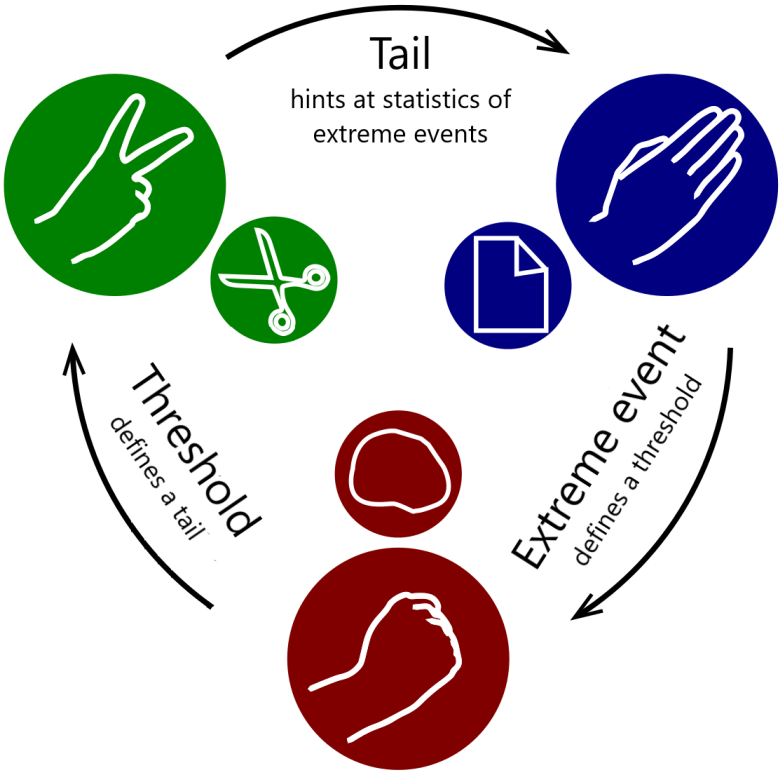


Figure 4.1. Triality of an extreme event.

data to the Generalized Extreme Value Distribution (GEVD).

Our results (Sec. 4.4.1) show that the distribution of block maxima is a composition of several distributions (Fig. 4.7b). The second approach relies on extracting the peak values reached for any period during which values exceed a certain threshold (falls below a certain threshold). We will cover major methods of choosing the threshold in Sec. 4.4.2.

Usually with the second approach the analysis may involve fitting two distributions: one for the number of events in a time period considered and a second for the size of the exceedances. But in practice a suitable threshold is unknown and must be determined for each analysis [91]. We show graphical methods, the rules of thumb and automatic methods are used by us to select the threshold.

All the above reveals the triality of nature of extreme events (Fig. 4.1). From a scientific point of view, the empirical emergency scales (based on the percent of the

populations affected and the economical impact of damages) determine the qualitative characteristics of extreme events. The scales essentially specify the threshold for each level of the severity of extreme events. Such empirical way of choosing the threshold, called the rule of thumb, is studied in Sec. 4.4.2.3. A choice of the threshold can be made using statistical analysis tools, such as graphical approaches (Sec. 4.4.2) and automatic methods (Sec. 4.4.2.2). Rather than characterize extreme events solely in terms of the distribution of weekly maxima, threshold methods take account of all events beyond a given high threshold, also known as exceedances, i.e. the chosen threshold establishes a tail of a distribution that shows the extreme events. This step completes the cycle (Fig. 4.1).

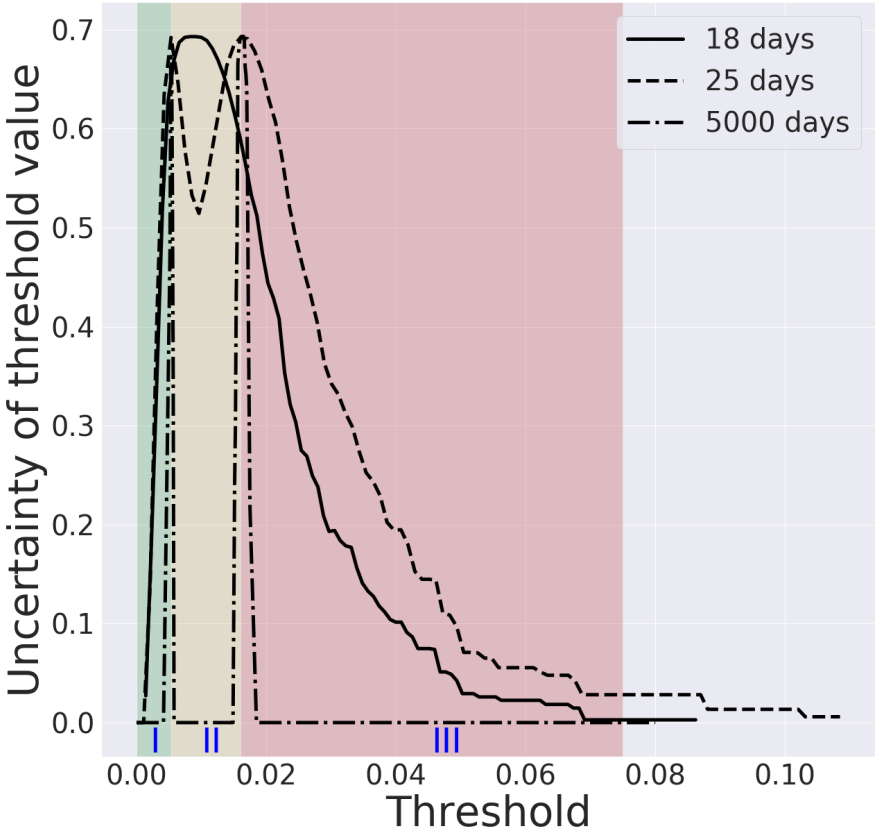


Figure 4.2. Degree of uncertainty of different values of the thresholds based on the amount of trading days taken into account. Three shaded regions mark three scales of emergency: I - subcritical, II - critical, III - extreme and the solid curve represents uncertainty of the Red Queen State.

At each part of the diagram (Fig. 4.1), we face uncertainty of defining the threshold to study a tail distribution to decide whether a particular event is extreme. We applied several methods to select a threshold in Sec. 4.4.2 such as a rule of thumb and an automatic method, and in each instance the choice depends on amount of data considered, a method used, etc. Moreover, once new data becomes available, we cannot guarantee sustainability of the previously chosen threshold value. In addition to that, assessment of the severity of the extreme event is also ruled by consequences, which may be revealed *ex past facto*. In Sec. 4.4.3 we presented statistics of extreme events under threshold uncertainty. We analyzed the degree of uncertainty of the threshold value (chosen with the rule of thumb) based on the probability of its change at any given day. Fig. 4.2 demonstrates the degree of uncertainty that depends on both the amount of data available and the values of thresholds. We observed several cases:

(i) If the amount of data is not sufficient (a solid line corresponding to an 18-day window of obervation in Fig. 4.2), then the uncertainty curve forms a skewed profile attaining a single maximum for some value of the threshold. In this situation, an observer's perception of events reminds *Red Queen* from "Through The Looking-Glass and What Alice Found There" by Lewis Carroll [92] who said "When you say hill, I could show you hills, in comparison with which you'd call that a valley". Our understanding of the events whether they are extreme or not is very limited and uncertainty is blurry. As events become more severe, our uncertainty that the events are extreme decreases. The observer realizes that the events are extreme, but a precise point at which the events turn to be severe cannot be determined. This case is called the *Red Queen State*.

(ii) As the window of observation becomes larger, the uncertainty curve exhibits two maxima indicating that the amount of data is sufficient. As we further extend the window the curve torrents into sharp peaks (Fig. 4.2). The latter ones clearly separate the threshold values into three regions: three levels of emergency.

   (I) In the region I (*subcritical*), the threshold values are small to raise concern about extreme events. Then we can observe a spike with the degree of

uncertainty attaining its first maximum. This extremum indicates a transition to the next kind of uncertainty.

(II) In the region II (*critical*), uncertainty is conceptualized with a question whether a magnitude of the event is already critical, extreme or not yet. Further, we see another jump of uncertainty. At this point we certain that events are not regular anymore.

(III) We consider all events in this region *extreme* with our uncertainty decreasing. If these events are not extreme then what are they?

We consider the Red Queen state and three scales, which are based on the degree of uncertainty, our contribution to the discussion on the extreme events and emergency scales. We conclude in the last Sec. 4.5.

4.2 Data source and description

The analysis of extreme events in the S&P 500 time series of log-return has been performed based on the data collected during 9,835 trading days (39 years), in the period from January 2, 1980, till December 31, 2018 (see Fig. 4.3) acquired using the publicly available source at Yahoo Finance (https://finance.yahoo.com/quote/%5EGSPC/).

Figure 4.3. The S&P 500 Index of 500 large-cap U.S. stocks assessing market performance.

The data set contains S&P 500 index at market open, highest point reached in the day, lowest point in the day, index of the stock at market close, number of shares traded. We have used index at market close since this information was always present

in the data set. Computations were made using Python's numerical libraries, such as NumPy and Pandas, as well as R-language to perform statistical analysis.

4.3 Log returns: between noise and random walks

Due to complexity of financial data, in order to analyze dynamics of the given financial time series, we computed *log return*, denoted $R_{\ln}$, according to the following formula [93]:

$$R_{\ln}(t) = \ln\left(\frac{\text{S\&P Index}(t)}{\text{S\&P Index}(t-1)}\right), \tag{4.1}$$

where S&P Index(t) is the value of S&P 500 at market close (Fig. 4.4).

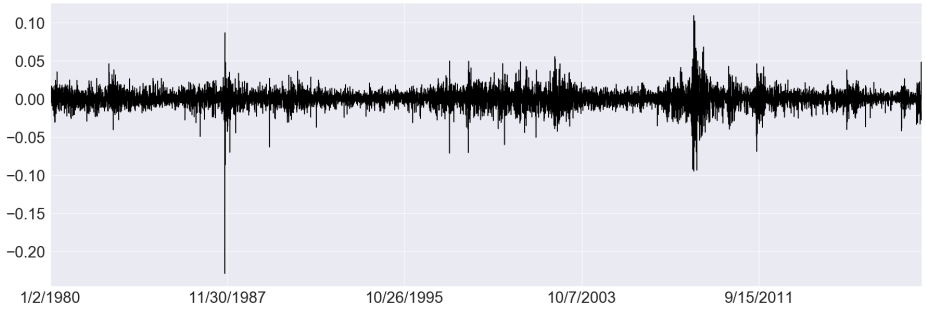


Figure 4.4. The S&P 500 index log-return at market close.

The lowest drop of log-return in Fig. 4.4 happened Monday, October 19,1987 known as Black Monday when the S&P500 shed value of nearly 23% [94]. Another significant drop occurred in 2008 when S&P 500 fell 38.49%, its worst yearly percentage loss [95]. In September 2008, Lehman Brothers collapsed as the financial crisis spread. However, on Oct 13, 2008: S&P 500 marks its best daily percentage gain, rising 11.58% and registers its largest single-day point increase of 104.13 points [95].

Distribution of $R_{\ln}$ values (Fig. 4.5) is asymmetrically skewed, with fat heterogeneous right and left tails.

The log-return time series has a scale invariant structure when the structure repeats itself on sub-intervals of the signal, $X(ct) = c^H X(t)$, where the Hurst exponent H characterizes the asymptotic behavior of the *auto-correlation function* of the time series [96, 97, 98]. The larger Hurst exponent is visually seen as more slow evolving

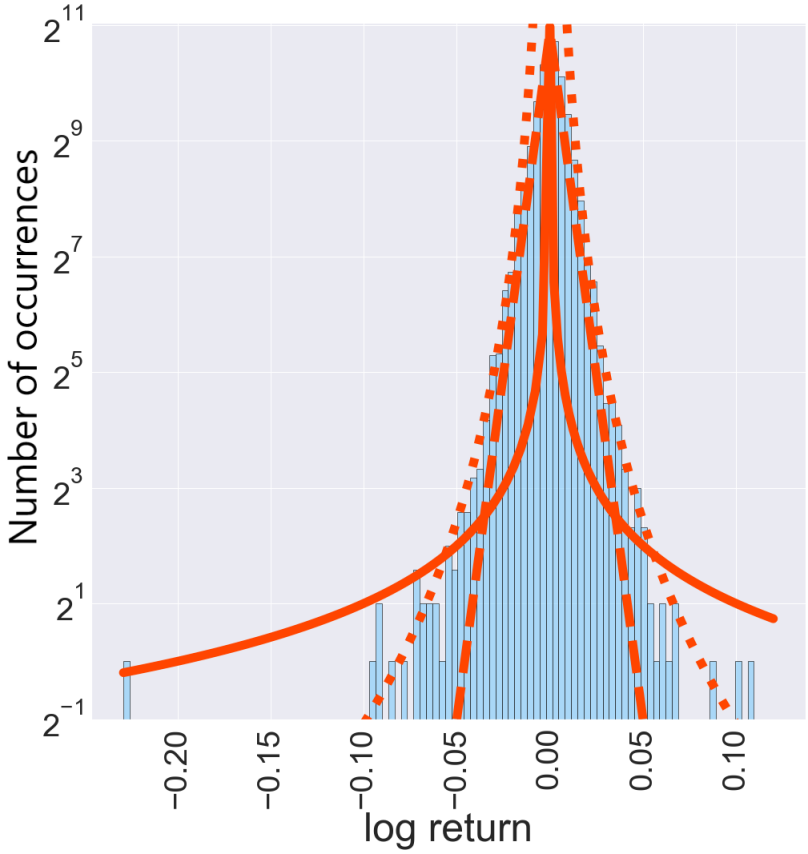


Figure 4.5. Distribution of log-return values in the log 2-linear scale. Solid lines correspond to a Zipf's Law ($\propto x^{-1}$), dotted lines represent Power Law ($\propto x^{-3.5}$), and dashed lines correlate to Gaussian distribution ($\propto \exp(-x^2)$). Curves are given for reference only.

variations (i.e., more persistent structure) of the time series [97, 99, 100]. Processes with $0 < H < 0.5$ exhibit *antipersistence*, with an increase in the process is likely to be followed by a decrease in the next time interval resulting in sample paths with a very rough structure [97, 99, 100]. On the contrary, values $0.5 < H < 1$ lead to long-range dependence ("long memory") in the time series, with more likely the same sign of successive increments (*persistence*) and smoother sample trajectories. Finally, the time series constitutes *random walks* when $H > 1$ that have more apparent slow evolving fluctuations [97, 99, 100]. The q-order Hurst exponent H_q is only one of several types of scaling exponents used to parameterize the multifractal structure of time series [97, 101].

The log-return time series for S&P 500 exhibits local fluctuations with both extreme small and large magnitudes, as well as short- and long-range dependences on different time scales [102, 103]; it is not normal distributed and all q-order statistical moments should to be considered to describe the spatial and temporal variation that reveals a departure of the log-return time series from simple random walk behavior [97, 99]. The q-order weights the influence of segments with large and small fluctuations. The negative q's are influenced by the time series segments with small fluctuations, and large fluctuations influence the time series segments for positive q's.

In our work, we use the standard *multifractal detrended fluctuation analysis* (MFDFA) algorithm [97, 101] for estimating the q-order Hurst exponents and the multifractal spectra directly from the time series:

1. The original time series x_k, $k = 1, \ldots, N$ is aggregated by computing the cumulative sums $Y(n) = \sum_{k=1}^{n} (x_k - \langle x \rangle)$, $n = 1, \ldots, N$, where $\langle x \rangle$ denotes the sample mean;

2. The aggregated data is divided into $\lfloor N/s \rfloor$ non-overlapping segments of length s;

3. The maximum likelihood estimator of the residual variance in segment ν, $\nu = 1, \ldots, N_s$,

$$F^2(\nu, s) = \begin{cases} \frac{1}{s} \sum_{i=1}^{s} \left\{ Y[(\nu - 1)s + i] - y_\nu(i) \right\}^2, & \text{for } \nu = 1, \ldots, N_s, \\ \frac{1}{s} \sum_{i=1}^{s} \left\{ Y[N - (\nu - N_s)s + i] - y_\nu(i) \right\}^2, & \text{for } \nu = N_s + 1, \ldots, 2N_s, \end{cases}$$

   where $y_\nu(i)$ is the $m-$degree polynomial fitted the aggregated observations in the segment;

4. For each segment of length s and for each positive or negative values of the moment order q, the q-order fluctuation function,
   $F_q(s) = \left\{ \frac{1}{N} \sum_{\nu=1}^{N_s} \left[F^2(\nu, s) \right]^{q/2} \right\}^{1/q}$, are calculated. The local fluctuations F_q with large and small magnitudes is graded by the magnitude of the negative or positive q-order, respectively;

5. A linear regression of $\ln F_q(s)$ on $\ln s$ for all s is performed, and the slope of the

linear function $\ln F_q(s) \propto H_q \ln s$ is used as an estimator of the q-order Hurst exponent H_q for each q-order fluctuation function F_q.

The fractal structures of the *positive* and *negative* log-return time series and its deviations within time periods with large and small fluctuations are assessed by the q-order Hurst exponents (see Fig. 4.6). The slopes H_q of the regression lines are q-

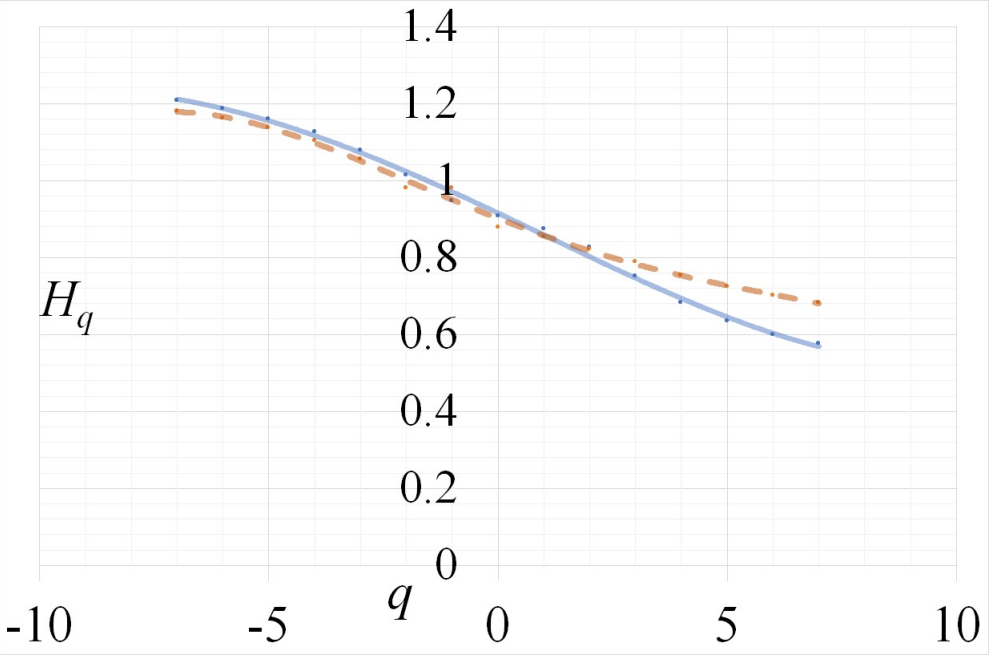


Figure 4.6. The q-order Hurst exponents H_q for the time series of positive (the dashed line) and negative (the bold line) log-returns.

dependent for the multifractal time series of positive (the dashed line) and negative (the bold line) log-returns. (see Fig. 4.6). Decreasing H_q with the q order indicates that the segments with *small fluctuations have a random walk like structure* whereas segments with *large fluctuations have a noise like structure*.

4.4 Tails, thresholds, and extreme events

There are two primary approaches to analyzing extreme values (the extreme deviations from the median of the probability distributions) in data:

(i) The first and more classical approach reduces the data considerably by taking

maxima of long blocks of data, e.g., annual maxima. The GEVD function has theoretical justification for fitting to block maxima of data [104].

(ii) The second approach is to analyze excesses over a high threshold. For this second approach the generalized Pareto distribution (GPD) function has similar justification for fitting to excesses over a high threshold [104].

4.4.1 Generalized extreme value distributions

The GEVD is a flexible three-parameter continuous probability distributions that was developed with extreme value theory to combine the Gumbel, Fréchet, and Weibull extreme values distributions into one single distribution [105, 106]. The GEV distribution has the following pdf [107]:

$$f(x;\mu,\sigma,\xi) = \frac{1}{\sigma} t(x)^{\xi+1} e^{-t(x)},$$

where

$$t(x) = \begin{cases} (1+\xi(\frac{x-\mu}{\sigma}))^{-1/\xi} & \xi \neq 0 \\ e^{-(x-\mu)/\sigma} & \xi = 0 \end{cases}$$

and $\mu \in \mathbb{R}$ is the location parameter, $\sigma > 0$ is the scale parameter, and $\xi \in \mathbb{R}$ is the shape parameter. When the shape parameter ξ is equal to 0, greater than 0, and lower than 0 [104], the GEV distribution is equivalent to Gumbel [108], Fréchet [109] and "reversed" Weibull distributions [110], respectively.

The *Gumbel distribution*, also named as the Extreme Value Type I distribution, has the following *pdf* and *cdf*:

$$f(x;\mu,\beta) = \frac{1}{\beta} e^{-(\frac{x-\mu}{\beta} + e^{-\frac{x-\mu}{\beta}})}, \tag{4.2}$$

$$F(x;\mu,\beta) = e^{-e^{-\frac{x-\mu}{\beta}}} \tag{4.3}$$

where $x \in \mathbb{R}$, μ is the location parameter, $\beta > 0$ is the scale parameter. Specially, when $\mu = 0$ and $\beta = 1$, the distribution becomes the standard Gumbel distribution. Generalizations of the Gumbel distribution, which are of flexible skewness and kurtosis due to the addition of one more shape parameter are widely used for extreme value

data as they better fit data [111]. The distribution in (4.2) has been employed as a model for extreme values [112, 113]. The distribution has a light right tail, which declines exponentially, since its skewness and kurtosis coefficients are constant.

The *Fréchet distribution*, also known as the Extreme Value Type II distribution, has the following *pdf* and *cdf*, respectively:

$$f(x;\alpha,\beta) = \frac{\alpha}{\beta}(\frac{\beta}{x})^{\alpha+1}e^{-(\frac{\beta}{x})^{\alpha}},$$
$$F(x;\alpha,\beta) = e^{-(\frac{\beta}{x})^{\alpha}}$$

where $\alpha > 0$ is the shape parameter and $\beta > 0$ is the scale parameter.

The *Weibull distribution* is known as the Extreme Value Type III distribution. The *pdf* and *cdf* of a Weibull random variable are shown as follows, respectively:

$$f(x;\lambda,k) = \begin{cases} \frac{k}{\lambda}(\frac{x}{\lambda})^{k-1}e^{-(x/\lambda)^k} & x \geq 0 \\ 0 & x < 0 \end{cases}$$

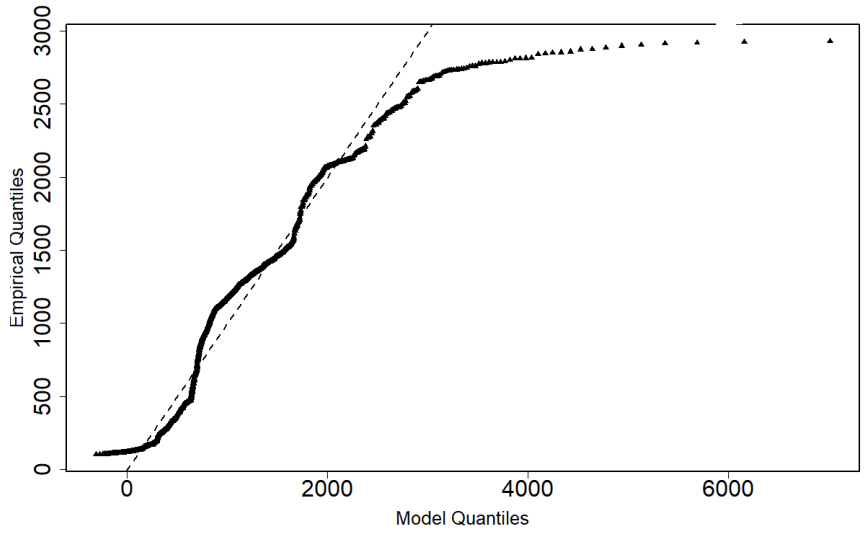
$$F(x;\lambda,k) = \begin{cases} 1 - e^{-(x/\lambda)^k} & x \geq 0 \\ 0 & x < 0 \end{cases}$$

where $\lambda > 0$ is the scale parameter and $k > 0$ is the shape parameter.

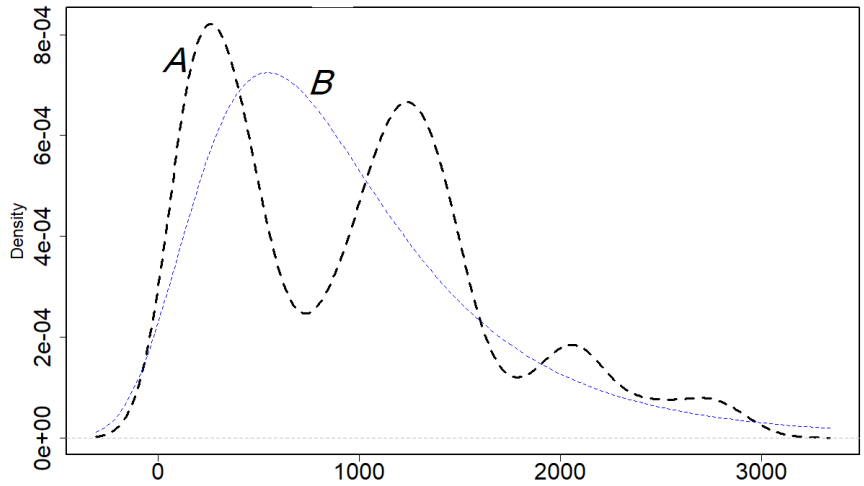
Further we show the application of the GEV model to the stock market close price using the *weekly-return* data that was calculated by

$$R(t) = \frac{(\text{maximum close price of week } t) - (\text{maximum close price of week } (t-1))}{(\text{maximum close price of week } (t-1))}.$$

The results of fitting the GEV distribution to (weekly) block maxima data is presented in Fig. 4.7 and Table 4.1 that present the Quantile-quantile plot (QQ-plot), quantiles from a sample drawn from the fitted GEV *pdf* against the empirical data quantiles with 95% confidence bands. The maximum likelihood estimators of the GEV distribution are the values of the three parameters (μ, σ, ξ) that maximize the log-likelihood. The magnitude along with positive sign of ξ indicates the fat-tailness of the weekly-return data, which is consistent with the quantile plot.



a.)



b.)

Figure 4.7. a.) A general extreme value QQ-plot with maximum likelihood estimation ; b.) Density plot of empirical data where a dashed curve A is based on the empirical data, and a dashed curve B is modeled. $N = 2036$ and bandwidth is 135.9.

Based on the statistical analysis presented above (Fig. 4.7a) , we see that the distribution of the weekly-return data can be described by a combination of different distributions. The density plot (Fig. 4.7b) having two humps validates the idea of a mixture of distributions.

Table 4.1. Parameter Estimates for the GEV fitted model with Maximum Likelihood Estimator. The 95% confidence intervals for each estimates are included.

	Location $\hat{\mu}$	Scale $\hat{\sigma}$	Shape $\hat{\xi}$
Estimated parameter	606.3260	511.1713	0.1215
95% a lower bound of the confidence interval	576.43	486.42	0.05
95% an upper bound of the confidence interval	636.22	535.92	0.19

4.4.2 How to choose a threshold

The classical approach for modeling extreme events is based on the GPD. It was proved [114] that if a threshold u is chosen and $X_1, X_2, \ldots X_n$ are observations above u, then the limiting distribution for excess over threshold is indeed GPD. In applications, the GPD is used as a tail approximation [115] of values $x - u$ exceeding the threshold u. The GPD is determined by scale and shape parameters $\sigma_u > 0$ and ξ, respectively, or in terms of threshold excess $x - u$, producing the following formula

$$G(x|u, \sigma_u, \xi) = \Pr(X < x|X > u) = \begin{cases} 1 - \left[1 + \xi\left(\frac{x-u}{\sigma_u}\right)\right]_+^{-1/\xi}, & \xi \neq 0 \\ \\ 1 - \exp\left[-\left(\frac{x-u}{\sigma_u}\right)\right]_+, & \xi = 0 \end{cases} \tag{4.4}$$

where $f_+ = \max(f, 0)$. When $\xi > 0$, it takes the form of the ordinary Pareto distribution. This case is the most relevant for financial time series, since it is heavy-tailed. For security returns or high-frequency foreign exchange returns, the estimates of ξ are usually less than 0.5. When $\xi = 0$, the GPD corresponds to the exponential distribution [116].

There are several properties of GPD [114], such as, 'threshold stability' property: if X is GPD and $u > 0$, then $X - u$ provided $X > u$ is also GPD. Therefore, a Poisson process of exceedance times with generalized Pareto excess implies the classical extreme value distributions [117]. The above suggests that generalized Pareto distribution is a practical tool for statistical estimation of the extreme values, given

a sufficiently high threshold. The rest of this chapter is devoted to a question about how high a threshold should be.

4.4.2.1 Graphical approaches to estimate threshold

One of the most common ways to determine a suitable threshold is to graphically inspect data. This approach [115] requires substantial expertise, that can be subjective and time consuming. In some cases, when dealing with several data sets, a uniform threshold may be proposed and kept fixed making the entire evaluation even more subjective.

The most common graphical tools are: mean excess plot [118], threshold stability plot [115], QQ-plot [119], Hill plot [120], return level plots [115], etc.

The mean excess plot is a tool widely used in the study of risk, insurance and extreme values. One use is in validating a generalized Pareto model for the excess distribution. The distribution of the excess over a threshold u for a random variable X with distribution function F is defined as

$$F_u(x) = \Pr\left(X - u \leq x | X > u\right). \tag{4.5}$$

This excess distribution is the foundation for peaks over threshold modeling which fits appropriate distributions to data on excesses and widespread with many application in hydrology [121, 122], actuarial science [123, 124], survival analysis [125]. This modeling is based on the GPD that is suitable for describing properties of excesses. The mean excess (ME) function is one of the most common tools to determine a suitable threshold u. The ME function of a random variable X is defined as

$$M(u) = E(X - u | X > u), \tag{4.6}$$

provided $EX_+ < +\infty$, which is also known as mean residual life function. As Ghosh [118] noted for a random variable $X \approx G_{\xi,\beta}$, $E(X) < +\infty$ if and only if $\xi < 1$ and in this case, the ME function of X is linear in u:

$$M(u) = \frac{\beta}{1-\xi} + \frac{\xi}{1-\xi}u, \tag{4.7}$$

where $0 \leq u < +\infty$ if $\xi \in [0,1)$ and $u \in \left[0, -\frac{\beta}{\xi}\right]$ if $\xi < 0$. The linearity of the ME function characterizes the GPD class [118]. Davison and Smith [117] developed a simple graphical tool that checks data against a GPD model. Let $X_1 \geq X_2 \geq \ldots \geq X_n$ be the order statistics of the data, then ME plot depicts the points $\{X_k, \hat{M}(X_k) | 1 < k \leq n\}$, where $\hat{M}$ is the empirical ME function defined as

$$\hat{M}(u) = \frac{\sum_{i=1}^{n}(X_i - u)I_{[X_i>u]}}{\sum_{i=1}^{n} I_{[X_i>u]}}, \quad u \geq 0. \tag{4.8}$$

If the ME plot is close to linear for sufficiently large values of the threshold then there is no evidence against use of a GPD model. Another problem is to obtain a natural estimation $\hat{\xi}$ of ξ. There are several methods to estimate $\hat{\xi}$, such as: (i) Least squares [115], (ii) Maximum likelihood estimation [117], (iii) the Hill estimator [120], (iv) the Pickands estimator [126], (v) quantile-quantile plot (QQ-plot) [115], (vi) the moment estimator [127].

For example, the QQ plot depicts the points
$Q_{m,n} = \left\{ \left(-\log\left(\frac{i}{m}\right), \log\left(\frac{X_i}{X_m}\right)\right) \middle| 1 \leq i \leq m \right\}$ where $m < n$ and $\xi > 0$. In case $\xi < 0$, QQ plot is the plot of points $Q'_{m,n} = \left\{ \left(X_i, G^{\leftarrow}_{\hat{\xi},1}\left(\frac{i}{n+1}\right)\right) \middle| 1 \leq i \leq n \right\}$, where $\hat{\xi}$ is an estimate of ξ based on m upper order statistics.

Recently, new graphical diagnostic tools have been introduced: a new multiple-threshold GP model with a piece-wise constant shape parameter [128]; plots measuring a surprise at different candidates to select threshold, using results of Bayesian statistics [129, 130]; structure of maximum likelihood estimators have been studied to develop diagnostic plots with more direct interpretability [131].

With this choice for a threshold, $u = 0.016$, we get 507 exceedances with the empirical life becoming close to linear above this choice of the threshold (Fig. 4.8). Similarly, we find a threshold for negative returns. In this case, all computations were repeated for absolute values of negative returns (Fig. 4.9). With this choice for the threshold $u = 0.017$, there are 462 exceedances. One can see that empirical MRL becomes almost linear above $u = 0.017$.

Ghosh and Resnick [118] noted that despite graphical diagnostic is a tool commonly accepted by practitioners, there are some problems associated with the methods mentioned above, such as: (i) an analyst needs to be convicted that $\xi < 1$ since for $\xi \geq 1$

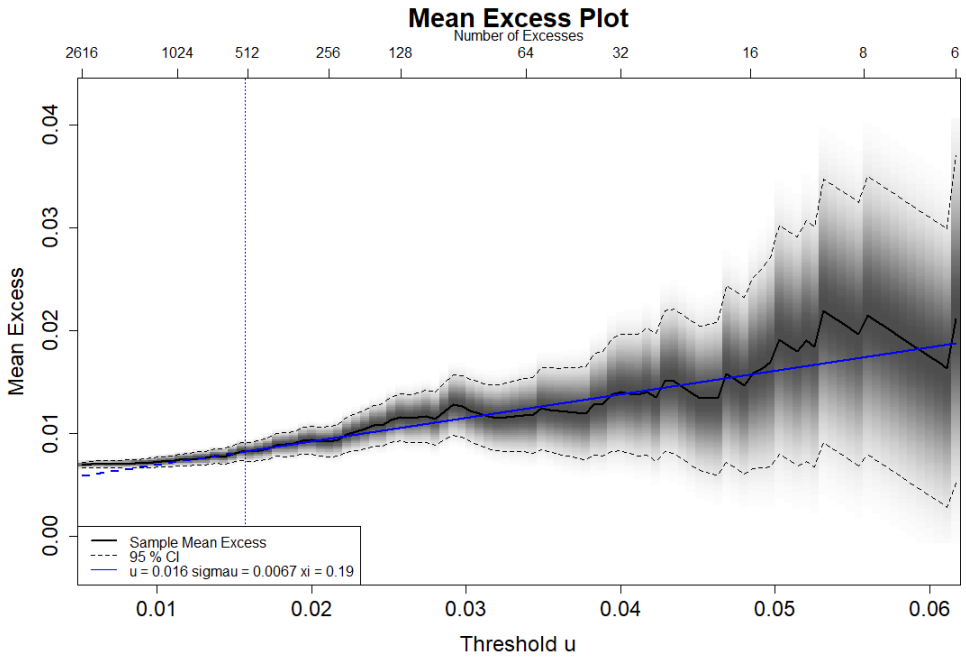


Figure 4.8. Mean residual life plot for the S&P 500 positive returns. Solid jagged line is empirical MRL with approximate pointwise Wald 95% confidence intervals as dashed lines. The threshold u is estimated at 0.016. A vertical dashed line marks this threshold.

random sets are the limits for the normalized ME plot. Such random limits lead to wrong impressions. Certain methods described above work with ξ defined on specific intervals; (ii) in case distributions are not close to GPD can mislead the mean excess diagnostics.

Based on the graphical approach, the threshold for negative return of S&P 500 was chosen at $u = -0.017$ and for positive return at $u = 0.016$. Distribution of exceedances over respective thresholds are shown in the Fig. 4.10.

The QQ-plots shown in the Fig. 4.11 demonstrate that with a few exceptions, exceedances follow the GPD.

4.4.2.2 Automatic methods to estimate thresholds

As was mentioned above, the graphical approaches as well as rules of thumb can be highly objective, time consuming, and require certain professional background. Thus

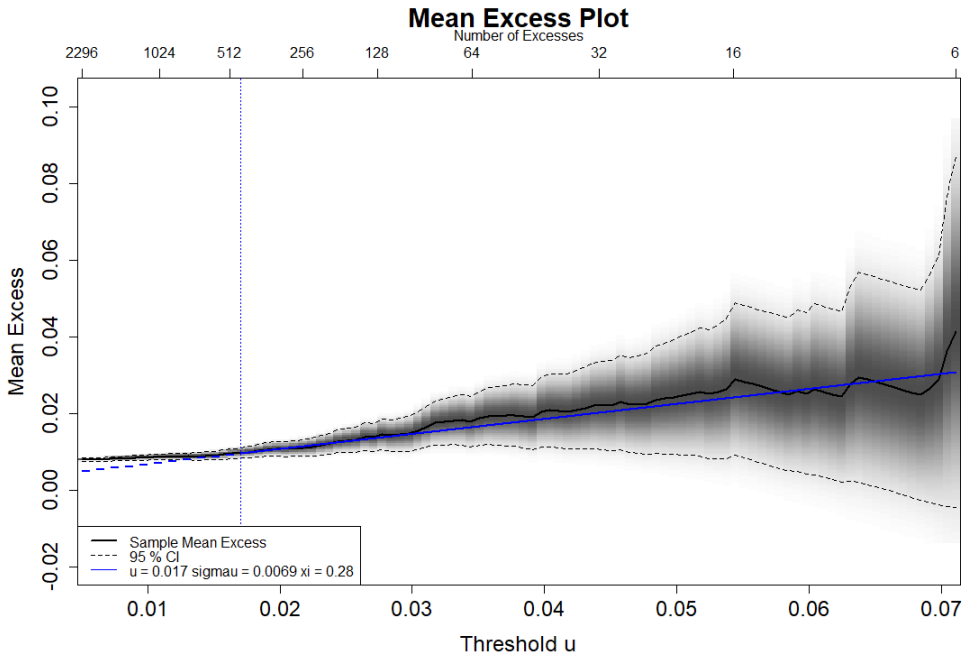


Figure 4.9. Mean residual life plot for the S&P 500 negative returns. Solid jagged line is empirical MRL with approximate point-wise Wald 95% confidence intervals as dashed lines. The threshold u is estimated at 0.017. A vertical dashed line marks this threshold.

some authors have proposed automatic selection methods that can treat chunks of Big Data: a pragmatic automated, simple and computationally inexpensive threshold selection method based on the distribution of the difference of parameter estimates when the threshold is changed [132]: it was shown that better performance is demonstrated by graphical methods and Goodness of Fit metrics that rely on pre-asymptotic properties of the GPD [133] using weighted least squares to fit linear models in the traditional mean residue life plot; the recently developed stopping rule ForwardStop [134], which transforms the results of ordered, sequentially tested hypotheses to control the false discovery rate [135] that provides reasonable error control [129].

A particular interest has a method that suggests a way to determine threshold automatically without time consuming and rather subjective visual approaches based on L-moments of GPD that summarize probability distributions, perform estimation

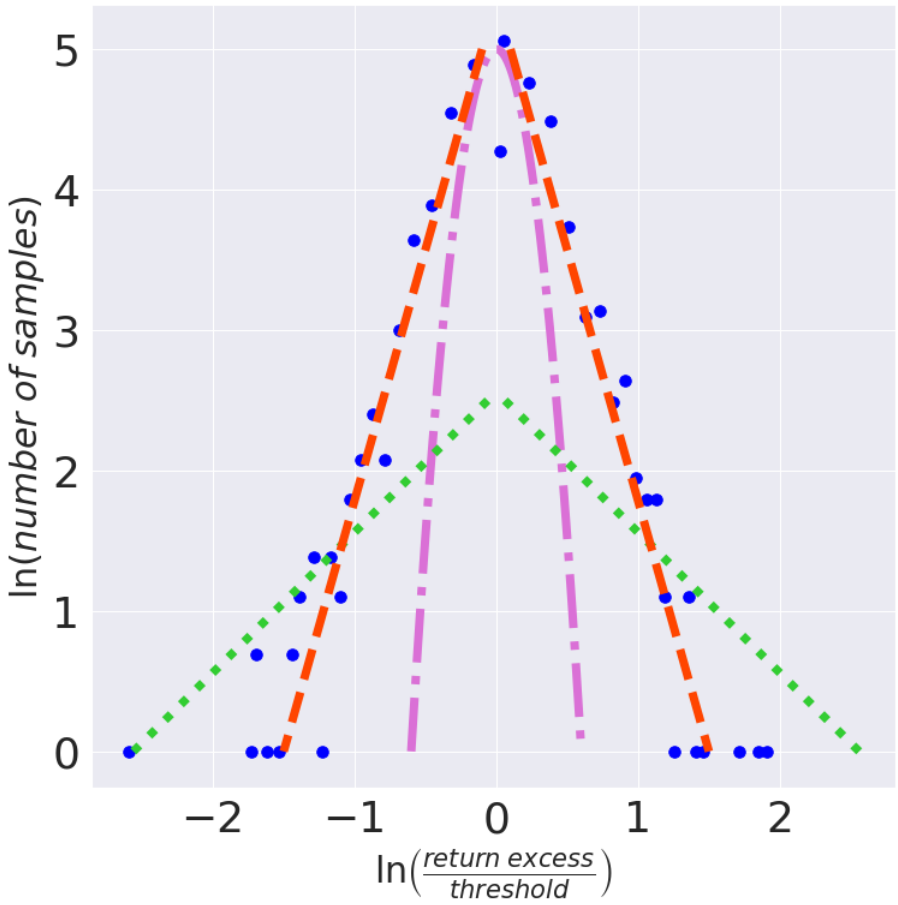


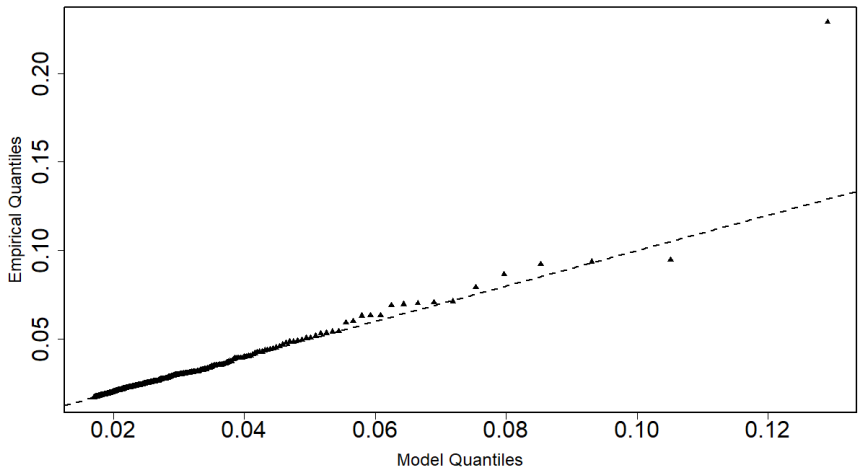
Figure 4.10. Distribution of exceedances normalized by thresholds $u = 0.016$ for positive and $u = -0.017$ for negative returns, respectively. Dotted lines represent Zipf's Law ($\propto x^{-1}$), dashdot lines represent Gaussian distribution ($\propto \exp(-x^2)$) and dashed lines represent power law ($\propto x^{-3.3}$). The curves are given for reference only.

of parameters and hypothesis testing [129].

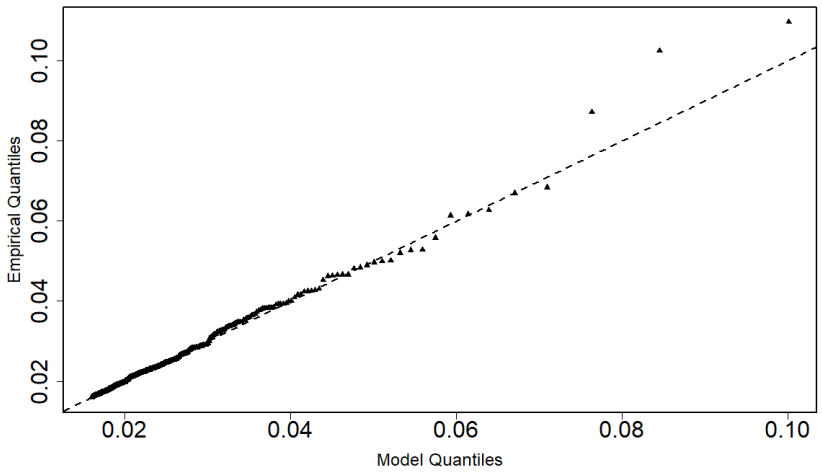
Probability weighted moments, defined by Greenwood [136], are precursors of L-moments. Sample probability weighted moments computed from data values $X_1, X_2, \ldots X_n$ arranged in increasing order, are given by

$$b_0 = \frac{1}{n}\sum_{j=1}^{n} X_j, \quad b_r = \frac{1}{n}\sum_{j=r+1}^{n} \frac{(j-1)(j-2)\ldots(j-r)}{(n-1)(n-2)\ldots(n-r)} X_j.$$

L-moments are certain linear combinations of probability weighted moments that



a.)



b.)

Figure 4.11. a.) Quantile-Quantile plot with maximum likelihood estimation for the negative threshold; b.) QQ-plot with maximum likelihood estimation for the positive threshold.

have simple interpretations as measure of location, dispersion and shape of the data

sample. The first few L-moments are defined by

$$\lambda_1 = b_0, \quad \lambda_2 = 2b_1 - b_0,$$
$$\lambda_3 = 6b_2 - 6b_1 + b_0, \quad \lambda_4 = 20b_3 - 30b_2 + 12b_1 - b_0.$$

(the coefficients are those of the shifted Legendre polynomials).

The first L-moment is the sample mean, a measure of location. The second L-moment is (a multiple of) Gini's mean difference statistic, a measure of the dispersion of the data values about their mean. By dividing the higher-order L-moments by the dispersion measure, we obtain the L-moment ratios, $\tau_r = \frac{\lambda_r}{\lambda_2}, r = 3, 4, \ldots$.

These are dimensionless quantities, independent of the units of measurement of the data. τ_3 is a measure of skewness and τ_4 is a measure of kurtosis - these are respectively the L-skewness and L-kurtosis. They take values between -1 and $+1$. For random variable with GPD with $\xi < 1$, the particular relationship between L-skewness and L-kurtosis is defined as

$$\tau_4 = \tau_3 \frac{1 + 5\tau_3}{5 + \tau_3}.$$

Given a sample $x_1, x_2, \ldots, x_n$, the Automatic L-moment Ratio Selection Method (ALRSM) works as follows [129]:

1. Define the set of candidate thresholds $\{u_i\}_{i=1}^{I}$ as $I = 20$ sample quantiles, starting at 25% by steps of 3.7%.

2. Compute the sample L-skewness and L-kurtosis for the excess over each candidate threshold $(\tau_{3,u_i}, \tau_{4,u_i})$ and determine d_{u_i} - the Eucledian distance:

$$d_{u_i} = \sqrt{(\tau_{3,u_i} - \tau_3)^2 + (\tau_{4,u_i} - g(\tau_3))^2},$$

   for $i = 1, \ldots, I$ with

$$g(\tau_3) = \tau_3 \frac{1 + 5\tau_3}{5 + \tau_3}.$$

3. The threshold after which the behavior of the tail of the underlying distribution

can be considered approximately GPD is then automatically selected as

$$u^* = \underset{u_i, 1 \le i \le I}{\operatorname{argmin}} \{d_{u_i}\},$$

that is, the level above which the corresponding L-statistics fall closest to the curve.

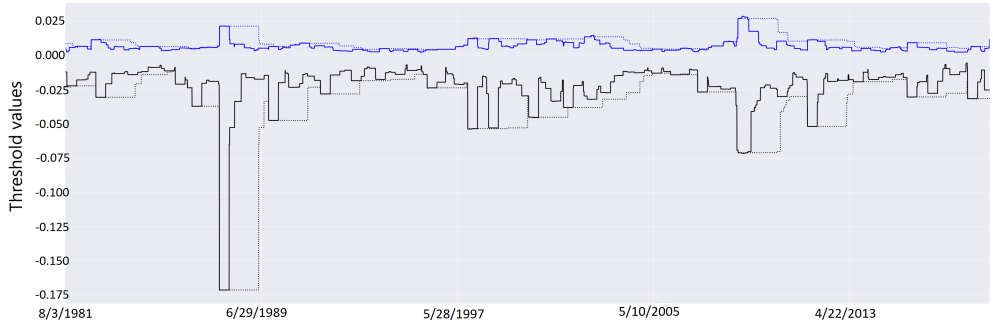


Figure 4.12. Value of the thresholds for positive and negative log-return based on L-moments. The solid black and blue lines correspond to negative and positive log return thresholds, respectively, and based on a window of 100 trading days. The dotted black and blue lines correspond to negative and positive log return thresholds, respectively, and based on a window of 400 trading days.

Using the L-moments method, we computed thresholds for S&P 500 log-return depending on an observation period, 100 trading days and 400 trading days. The Fig. 4.12 indicates that a threshold not only time dependent, but also depends on the size of data set used. Once again we cannot choose one value of the threshold that can be absolutely accurate.

4.4.2.3 Rules of thumb to choose a threshold

As earlier noted, the threshold sequence is a function of the properties of the GPD provided that a population is in the domain of attraction of the GPD. In case a distribution function F is known, derivation of the threshold selection is possible, however, in practice if F is unknown then there is no general form for the threshold sequence [115].

Practitioners often use so-called rules of thumb, many of them have little to know theoretical justification, for instance, a fixed quantile rule [137]: the upper 10% rule

or its derivative 5% rule; the square root rule $k = \sqrt{n}$ [138]. There is a procedure that tries to find a region of stability among the estimates of the extreme value index [139]. This method depends on a tuning parameter, whose choice is further analysed in [140]. Unfortunately, no theoretical analysis exists for this approach [141]. More comprehensive reviews of the threshold selection methods can be found in [115].

Even though most of methods mentioned above have no theoretical justification for an exact value of a threshold, we can find an approximate location for a threshold given a data set R that has M values of log-return, $R = [x_1, x_2, \ldots, x_M]$.

1. Let R_n be a list of n consecutive values of log-return, $R_n = [x_1, x_2, \ldots, x_n]$. Split R_n into two parts: $R_{n+} = \{x \in R_n | x \geq 0\}$ and $R_{n-} = \{x \in R_n | x < 0\}$; then sort them in an increasing order such that

$$R_{n+} = [x_1^+, x_2^+, \ldots, x_p^+], \qquad R_{n-} = [x_1^-, x_2^-, \ldots, x_q^-].$$

   where $p, q \in \mathbb{N}$ such that $p + q = n$ and n is the width of observation or a window of observation.

2. Next, we compute medians of R_{n+}, R_{n-}, call them x_{m+} and x_{m-}. These values will be a lower bound for a positive threshold and an upper bound for a negative threshold, respectively.

3. At this step, an upper bound for a positive threshold x_u and a lower bound for a negative threshold x_l are estimated using the rule of thumb, namely, the fixed quantile rule with upper 10% for positive log-return values and lower 10% for negative log-return values. With this we have

$$R_{n+} = [x_1^+, x_2^+, \ldots, x_{m+}, \ldots, x_u, \ldots, x_p^+], \quad R_{n-} = [x_1^-, x_2^-, \ldots, x_l, \ldots, x_{m-}, \ldots, x_q^-].$$

   The indices u, l can be found as $u = \lfloor p/1.111 \rfloor, \quad l = \lceil q/10 \rceil$.

4. A threshold for negative log-return values ranges from x_l to x_{m-} and a threshold for positive values is within x_{m+} and x_u, based on n observations from R_n.

Further, we shifted R_n to $R_n = [x_2, x_3, \ldots, x_{n+1}]$ to estimate new values of thresholds based on previous n observations. The process repeated until we exhausted the entire

data set R.

We chose a window of 300 days that was moving over the entire dataset producing a domain for threshold existence as shown in the Fig. 4.13. It is clear from the Fig. 4.13 that certain values of thresholds cannot sustain the entire period and must be updated from time to time.

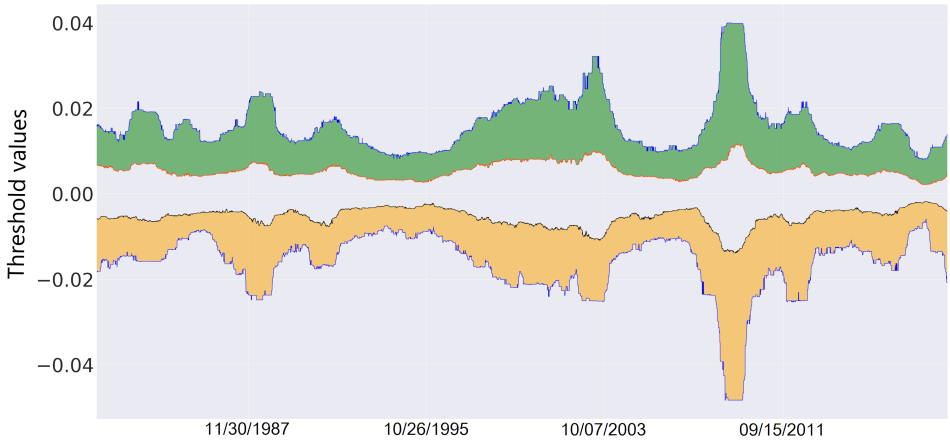


Figure 4.13. Possible threshold changing ranges from 03/11/1981 to 12/31/2018 based on 300 preceding trading days. A green strip represents positive log-return and an orange strip shows the threshold domain for negative log-return values.

Similarly to the window of 300 trading days, the Fig. 4.14 demonstrates how ranges of thresholds change as we move a window of 600 trading days across available data. Once again, some values of thresholds can exist for almost entire period, while other can exist a few months and then must be replaced with an updated value.

Based on Figs. 4.13, 4.14 we can see that a choice of the threshold depends on a size of the data set used. Moreover, the rules of thumbs alike graphical approaches require practitioners involvement in studying data before making a final choice for thresholds.

Equipped with these results, we can compute a validity period $\tau(u)$ for a threshold u. Let n be the window of observation. By moving the window over the data set containing M values of log-returns we obtain four lists with $M - n + 1$ elements in

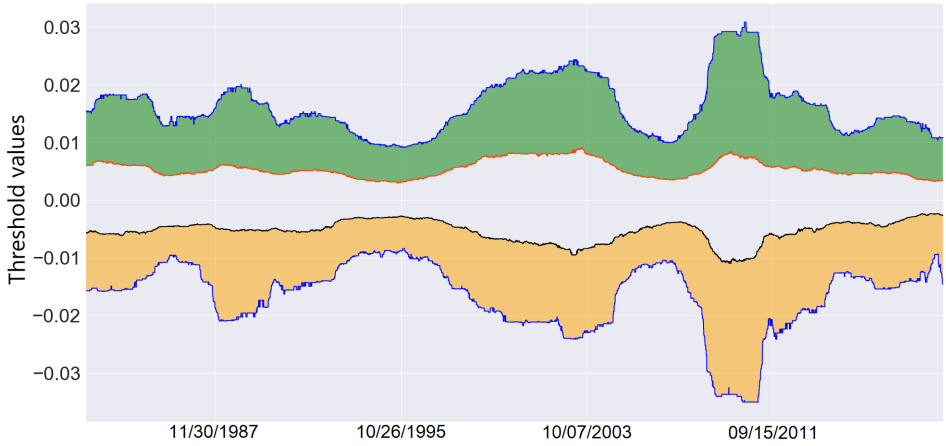


Figure 4.14. Possible threshold changing ranges from 05/18/1982 to 12/31/2018 based on 600 preceding trading days. A green strip represents positive log-returns and an orange strip shows the threshold domain for negative log-return values.

each

$$X_{m+} = [x_{m+,n}, x_{m+,n+1}, \ldots, x_{m+,M}] \qquad X_{m-} = [x_{m-,n}, x_{m-,n+1}, \ldots, x_{m-,M}]$$
$$X_u = [x_{u,n}, x_{u,n+1}, \ldots, x_{u,M}] \qquad X_l = [x_{l,n}, x_{l,n+1}, \ldots, x_{l,M}]$$

containing medians of positive and negative log-returns, upper 10% cut-offs and lower 10% cut-offs for positive and negative log-returns, respectively, as computed previously.

1. First, we compute the k threshold candidates u_i for positive log-return as $\{u_i^+ \in \mathbb{R} | \min(X_{m+}) + \frac{i}{k}(\max(X_u) - \min X_{m+}), i = 1, \ldots, k\}$. In a similar fashion, we compute a set of threshold candidates for negative log-returns, $\{u_i^- \in \mathbb{R} | \min(X_l) + \frac{i}{k}(\max(X_{m-}) - \min X_l), i = 1, \ldots, k\}$.

2. For each threshold candidate u_i found in the previous step we compute its validity duration, i.e. a number of days N_{u_i} a candidate would fall within admissible ranges. Set $N_{u_i} = 0$. For $j = n \ldots M$ if $x_{m+,j} < u_i < x_{u,j}$, then $N_{u_i} = N_{u_i} + 1$ for positive threshold candidates, similarly, if $x_{l,j} < u_i < x_{m+,j}$, then $N_{u_i} = N_{u_i} + 1$ for negative threshold candidates.

3. The probability that a threshold candidate will be changed on any given day is

$$\eta(u_i) = 1 - N_{u_i}/(M - n + 1).$$

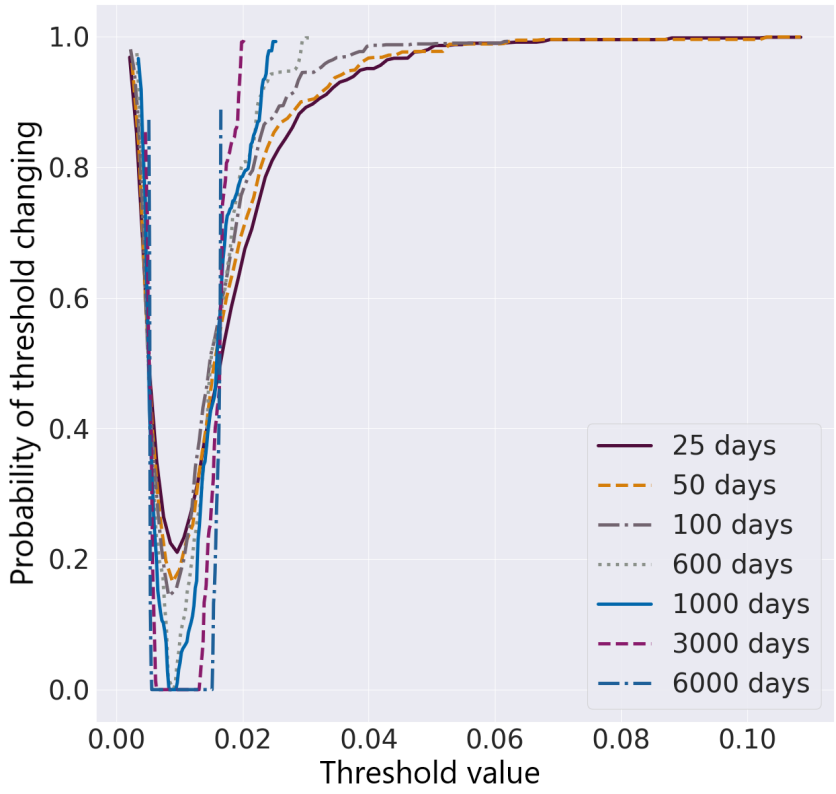


Figure 4.15. The probability that a threshold of the log-return values will be changed on any given day calculated over the different data windows ranging from 25 to 6000 trading days.

The Fig. 4.15 shows probabilities that a particular choice of the threshold can be changed on any day depending on the size of the dataset used, which is important to know especially when new data becomes available and is included for consideration. The more data we use to estimate a value of a threshold, the more likely the threshold will stay unchanged, however, as we add data, the threshold should be reconsidered. It also brings another issue: in many cases, statistical analysis is performed on a historical dataset that does not reflect a phenomena we study at the present time.

4.4.3 Statistics of extreme events under threshold uncertainty

The statistics of extreme events under threshold uncertainty can be described with the help of a simple model, in which the log-return of the index and the value of threshold are treated as random variables that yet can change inconsistently. The model that we are going to adopt and modify had been put forward by us for the first time in [142] to describe the behavior of systems close to a threshold of instability and used later in [143, 69] to model survival under uncertainty and the events of mass extinction. In the model of extreme events under threshold uncertainty, the current value of the log-return is quantified by a random number $x \in [0, 1]$ drawn accordingly some probability distribution function $\Pr\{x < z\} = F(z)$. The threshold value that might change any time once the new data become available is another random number $y \in [0, 1]$, which is drawn from another probability distribution function, $\Pr\{y < z\} = G(z)$. We assume that the rate of daily variations of the log-return values is greater than or equal to that of the threshold values, ultimately determining whether the current log-return value is extreme or not. In fact, it is the relative rate of random updates of x and y described in our model by the probability of inconsistency $\eta \geq 0$, that actually determines the statistics of extreme events.

At time $t = 0$, the value of log-return x is chosen with respect to the probability distribution function F, and the value of the threshold y is chosen with respect to the probability distribution function G. If $y \geq x$, the event is regular and the process keeps going to time $t = 1$. At time $t \geq 1$, either, with probability $\eta \geq 0$, the value of log-return x is drawn anew from the probability distribution function F, but the threshold keeps the value y it had at time $t - 1$, or with probability $1 - \eta$, the value of log-return x is updated anew from the probability distribution function F, and the level of supply y is updated either with respect to the probability distribution function G. As long as the value of threshold is not exceeded ($x \leq y$), the event is classified as regular, but the event is extreme whenever $x > y$.

The value of probability $\eta > 0$ can be interpreted as the reciprocal characteristic time interval, during which the threshold level remains unchanged, and vice versa the probability that a threshold of the log-return values will stay unchanged on any given day can be calculated as the inverse of the time interval during which the threshold value stays put. The probability that a threshold of the log-return values will stay

unchanged on any given day calculated over the different data windows ranging from 25 to 6000 trading days is shown in Fig. 4.15.

We are interested in the probability distribution $P_\eta(t)$ of the interval duration t between the sequential extreme events for some probability distribution functions F and G and a given value of the probability $\eta \geq 0$. A straightforward computation [142, 143], shows that, independently upon the value of η, the initial probability of choosing the level of demand below the supply level (to start the subsistence process) is $\int_0^1 dG(z)F(z)$. The general formulas for $P_\eta(t)$ can be found in [142, 143].

When $\eta = 0$, the values of log-return and the threshold value are updated coherently the resulting probability function,

$$P_{\eta=0}(t) \;=\; \left[\int_0^1 dG(z)F(z)\right]^t \int_0^1 dG(z)\,(1-F(z))\,, \tag{4.9}$$

decays exponentially fast with t, for any choice of the probability distribution functions F and G. In particular, if the threshold level and log-returns are drawn uniformly at random, $dF(z) = dG(z) = dz$, over the interval $[0, 1]$, the occurrence of an extreme event is statistically equivalent to simple flipping a fair coin, for which head and tail come up equiprobably,

$$P_{\eta=0}(t) \;=\; \frac{1}{2^{(t+1)}}. \tag{4.10}$$

On the contrary, when the threshold level is kept unchanged, $\eta = 1$, the statistics of intervals between the sequential extreme events,

$$P_{\eta=1}(t) \;=\; \int_0^1 dG(z)F(z)^t\,(1-F(z))\,, \tag{4.11}$$

decays asymptotically algebraically as $t \gg 1$ [142, 143]. For example, in the special case of uniformly random updates of the threshold and log-return values, the probability function (4.11) decays algebraically as

$$P_{\eta=1}(t) \;=\; \frac{1}{(t+1)(t+2)} \simeq \frac{1}{t^2}. \tag{4.12}$$

For a general family of invariant measures of a map of the interval $[0, 1]$ with a fixed

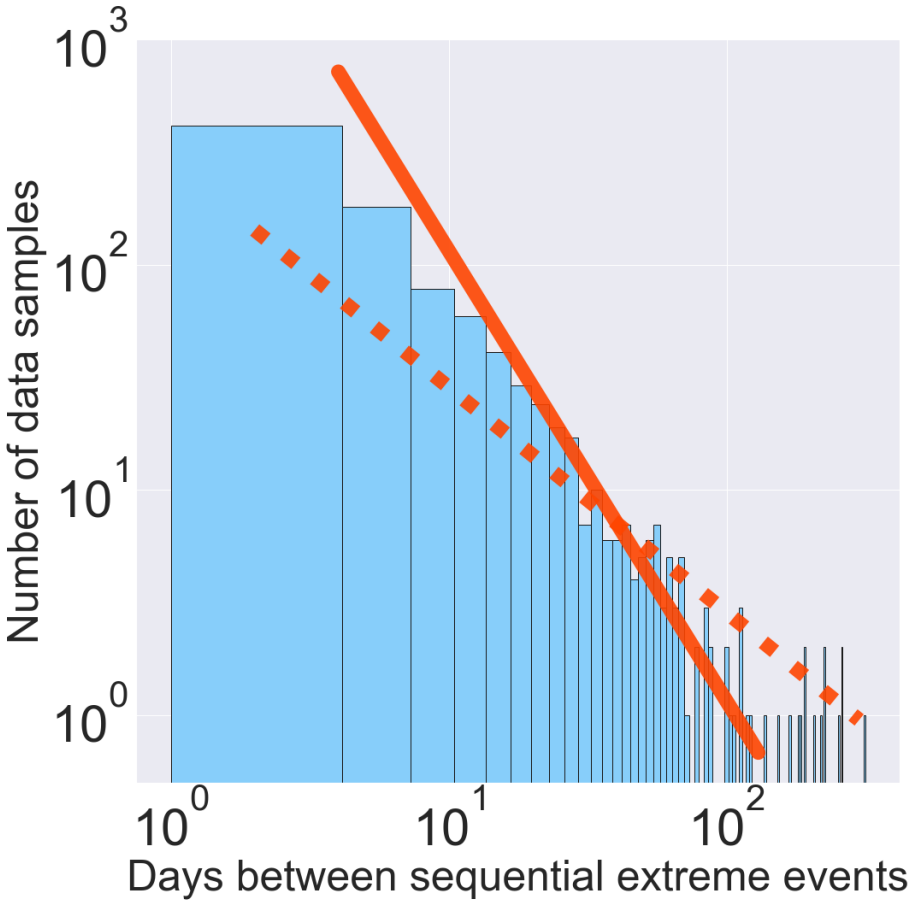


Figure 4.16. The statistics of time intervals (in days) between the sequential extreme events for the fixed threshold values, $u = 0.016$ and $u = -0.017$, for positive and negative fluctuations of the log-return respectively. The solid line $\propto t^{-2}$ corresponding to the asymptotic quadratic decay (4.12) is given for reference.

neutral point defined by the probability distributions F and G, absolutely continuous with respect to the Lebesgue measure, i.e.,

$$\begin{aligned} dF(z) &= (1+\alpha)z^{\alpha}\,dz, \qquad \alpha > -1, \\ dG(z) &= (1+\beta)(1-z)^{\beta}\,dz, \quad \beta > -1. \end{aligned} \tag{4.13}$$

Eq.(4.11) gives the probability function that exhibits the power law asymptotic decay

for $t >> 1$ [142, 143, 69]:

$$P_{\eta=1}(t) \;\simeq\; \frac{(1+\beta)\,\Gamma(2+\beta)\,(1+\alpha)^{-1-\beta}}{t^{2+\beta}}\,\left(1+0\left(\frac{1}{t}\right)\right). \tag{4.14}$$

The asymptotic decay of (4.14) seems to be algebraic,

$$P_{\eta=1}(\tau) \;\simeq\; \frac{1}{\tau^{2+\beta}}, \tag{4.15}$$

for $\beta > -1$, for any choice of the distributions F and G although it is mainly the character of the probability function G that determines the rate of decay of $P_{\eta=1}(t)$ with time. In the limiting case when the support of the probability distribution $G(x)$ determining the choice of the supply level is concentrated close to $x = 1$, i.e., is zero everywhere in the interval $[0, 1]$, except for a small interval of length ε up to 1, the *Zipf power law* asymptote $\propto t^{-1-\varepsilon}$, $\varepsilon > 0$, follows directly from (4.14) [142, 143, 69]. A possible modeling function for such a bountiful probability distribution, forming a thin spike as $x \to 1$, can be chosen in the form,

$$G_\varepsilon(x) \;=\; 1-(1-x)^\varepsilon, \quad \varepsilon > 0, \tag{4.16}$$

with the probability density in the interval $[0, 1[$,

$$dG_\varepsilon(x) \;=\; \frac{\varepsilon\, dx}{(1-x)^{1-\varepsilon}}. \tag{4.17}$$

The straight line shown in Fig. 4.16 represents the hyperbolic decay of time intervals between the extreme events.

4.5 Defining Emergency Scales by thresholds uncertainty.

According to the triality of the nature of extreme events (Fig. 4.1), once we assume statistics for log- returns, or ranges for threshold values are defined using different methods shown in Sec. 4.4.1, we introduce uncertainty to the threshold value that depends on the method itself and the amount of data we use. We measure uncertainty to justify an emergency scale to represent the extreme events.

Using the rule of thumb (Sec. 4.4.2.3), we determined the ranges for the threshold

values depending on the window of observation (Fig. 4.13) for both positive and negative log return values. For each admissible positive choice of a threshold u, we determine the probabilities $\eta(u)$ that the threshold u can be changed on any given day (Fig. 4.15), degrees of uncertainty have been assessed by the means of the Shannon's entropy for each threshold value and the window of observation [69] (4.18). The case with the negative values of the threshold is analogous.

$$H(\eta) = \begin{cases} -\eta \log \eta - (1-\eta)\log(1-\eta), & 0 < \eta < 1 \\ 0, & \eta = 0, 1 \end{cases} \tag{4.18}$$

where $\eta(u)$ is the probability that the threshold will be changed on any given day (Fig. 4.15).

We observed the Red Queen State and three emergency scales can be readily interpreted.

(i) If amount of data is not sufficient (a solid line corresponding to the 18-day window of obervation in Fig. 4.17), then the uncertainty curve forms a skewed profile attaining a single maximum of 0.69295 for the threshold value 0.008272. In this situation, an observer's perception of the events reminds *Red Queen* from "Through The Looking-Glass and What Alice Found There" by Lewis Carroll [92] who said "When you say hill, I could show you hills, in comparison with which you'd call that a valley". Our understanding of the events whether they are extreme or not is very limited and uncertainty is blurry. As events become more severe, our uncertainty that the events are extreme decreases. The observer realizes that the events are extreme, but a precise point at which the events turn to be severe cannot be determined. This case is called the *Red Queen State*.

(ii) As the window of observation becomes larger, 25 days, for instance, the uncertainty curve exhibits two maxima indicating that the amount of data is sufficient. This happens because $H(\eta)$ attains its maximum for $\eta = 1/2$ and the η curve admits a value $1/2$ twice (Fig. 4.15). As we further extend the window the curve torrents into sharp peaks (Fig. 4.17). The latter ones clearly separate the threshold values into three regions: three levels of emergency. The locations

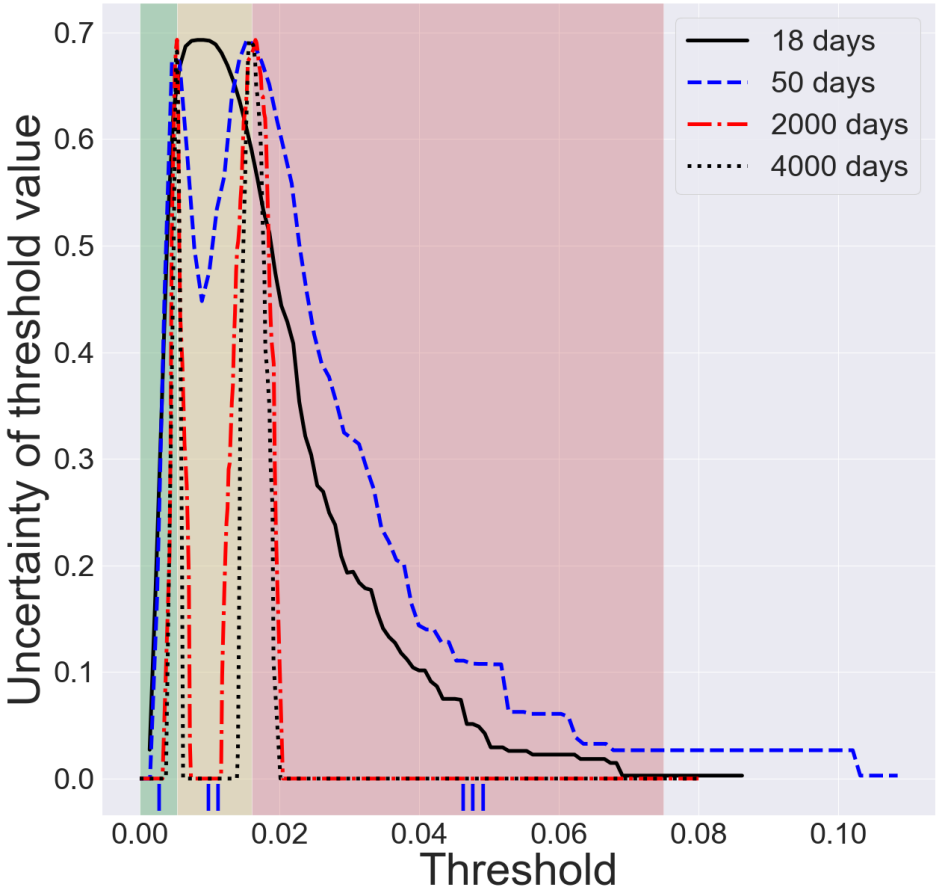


Figure 4.17. Degree of uncertainty of different values of the thresholds based on the amount of trading days taken into account. Three shaded regions mark three scales of emergency: I - subcritical, II - critical, III - extreme and the solid curve represents the Red Queen State.

of peaks of the curves are summarized in Table 4.2. The location of two peaks is not sensitive to the window of observation.

Emergency scales for S&P 500.

(I) *Subcritical.* In the region I, the threshold values are small to raise concern about extreme events. Then we can observe a spike with the degree of uncertainty attaining its first maximum. This extremum indicates a transition to the next kind of uncertainty. For the window of 2000 days, the region I lies in the interval $[0, 0.00534)$ (see Fig. 4.17, Table 4.2).

Table 4.2. Location of extrema points of the uncertainty curves from Fig. 4.17.

Window	First maximum		Second maximum	
	Threshold	Uncertainty	Threshold	Uncertainty
18 days	0.00827	0.69296	–	–
50 days	0.00461	0.67411	0.01532	0.69290
2000 days	0.00534	0.69315	0.01672	0.69034
4000 days	0.00533	0.68217	0.01580	0.69314

(II) *Critical.* In the region II, the interval $[0.00534, 0.01672)$ for the 2000-day window, uncertainty is conceptualized with a question whether a magnitude of the event is already critical, extreme or not yet. Further, we see another jump of uncertainty reaching 0.69034. At this point we are certain that events are not regular anymore.

(III) *Extreme.* In the case of the window of 2000 days, the interval $[0.01672, \infty)$ constitutes the region III. We consider all events in this region extreme with our uncertainty decreasing as threshold values increase.

With the analysis presented above, we define an emergency scale of three levels based on the three regions of the threshold values corresponding to three peaks of the uncertainty curve. This emergency scale is not sensitive to the size of the window of sufficient amount of data considered.

4.6 Conclusion

The S&P500 times series in the period from January 2, 1980 till December 31, 2018 exhibits an asymmetrical skewness of the distribution with the right and left power law tails. Multifractal detrended fluctuation analysis of log return time series for S&P 500 index reveals a scale invariant structure for the fluctuations on both small- and large scale magnitudes, as well as its short- and long-range dependence on different time scales. Moreover, the segments with small fluctuations have a random walk like structure whereas segments with large fluctuations have a noise like structure.

We have reviewed different methods of threshold selection and studied the extreme events presented in the time series using different statistical approaches. We found that the distribution of the weekly-return data can be described by a combination

of different distributions. Based on a graphical approach for threshold selection, we chose separate thresholds for the positive and negative values of the log return, 0.016 and -0.017, respectively. With this choice, we registered 507 instances of extreme events corresponding to raise of market and 462 extreme events related to market declines. With a few exceptions, exceedances over (under for negative log return values) the threshold follow the GPD.

The rule of thumb showed that a threshold value depends on the width of observation window, and the threshold can change at any moment, once new data become available. Uncertainty of the threshold values can be determined by the probability of changing the threshold on any given day.

The moment we make an assumption about statistics of distributions or the dataset is fixed, it leads to uncertainty of the threshold value which can be resolved by the emergency scales rigid to variation on the size of the dataset.

We suggested a statistical model that describes registration frequency of extreme events under threshold uncertainty. Our model fits well the statistics of occurrence of the extreme values in the S&P 500 time series.

CHAPTER 5
FLIGHT DATA REFINEMENT AND REDUCTION.

5.1 Introduction

Since the first flight data recorder (VADR) was invented in 1960's which purpose was to restore last minutes of flights in case of a crash or less serious extreme events, many manufactures worked hard to use data generated by VADR for health assessment of the airplanes.

MH-60 Black Hawk is also equipped with a voice and data recorder (VADR) that is capable of storing up to two hours of flight data, storing information from about 1500 sensors at the highest frequency of 16 Hz.

A Health and Usage Monitoring System (HUMS) records the status of critical systems and components on helicopters so that the early detection of progressive defects, or indications of them, is possible and thus rectification can be achieved before they have an immediate effect on operational safety. The on-board equipment stores data on a PCMCIA Card. For analysis, the card is downloaded after flight and maintenance analysis can then be performed on a ground-based computer. These systems were first deployed in the early 1990's as a response to the relatively poor continuing airworthiness record and their introduction led to, and continues to support, significant improvements in both safety and reliability.

The Unit Level Logistics System - Aviation (ULLS-A) is an automated aviation logbook and maintenance and logistics management system, which allows Army aviation units to perform and track maintenance and supply operations in accordance with The Army Maintenance Management Systems-Aviation.

All the three systems mentions above aim to help the ground team to access the health of a helicopter. However, there are certain problems associated to collecting and processing data generated by VADR, HUMS, and ULLS-A. The VADR unit has very reliable but limited amount of memory able to undergo high temperature, enormous pressure and staing underwater for extended periods of time. Every 2 hours, the new coming data overwites the previously recorded entries. On the other hand, HUMS can store larger amounts of data that need to be uploaded to a ground station. While transferring data to the ground station is not difficult, there are not

certain regulations that mandate such transfer. Also, HUMS timestamps depend on a partilucal clock readings when such tranfers occur. Lastly, ULLS-A logbook is maintained by the pilots and is subject to human errors. Usually, actual flight data differs from those in the logbooks.

In addition to the above issue, it is worth mentioning that HUMS and VADR files are generated every time an aircraft is turned on, even for maintenance or testing purposes. Therefore, there is a portion of generated files that do not contain any actual flight information.

Schematially, the process of fusing data from the above mentioned sources is demonstrated in Fig. 5.1.

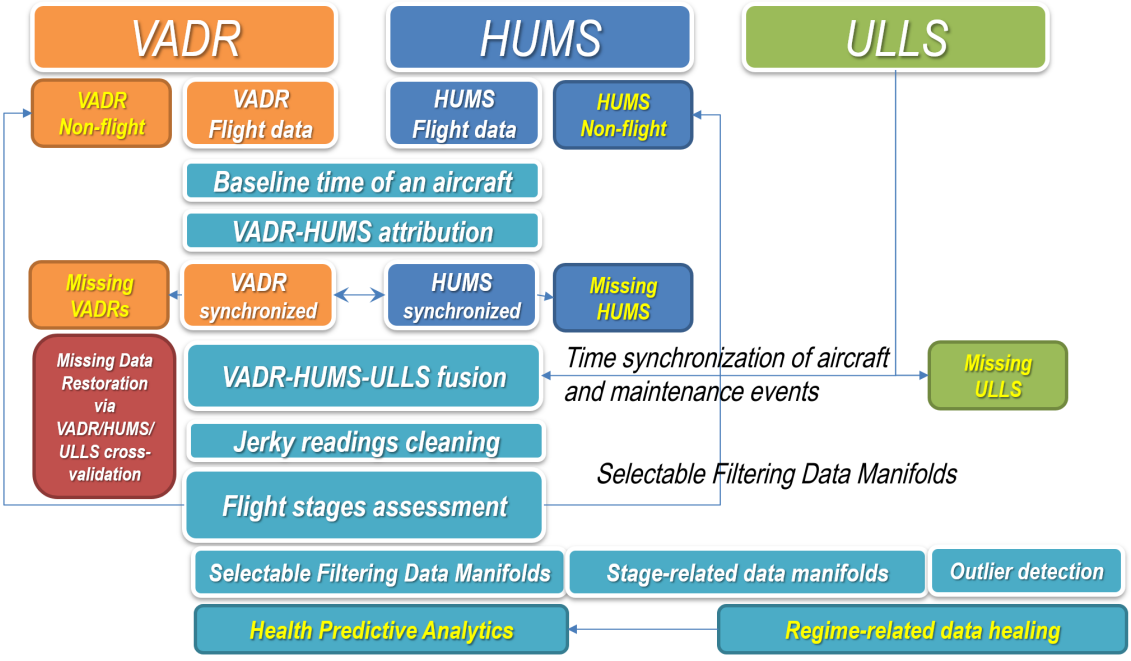


Figure 5.1. Data Refinement and Reduction flowchart.

We created an algorithm to separate files containing flight information from files that were generated by maintenance and accessment startups. Further, we propesed a reliable automated process to synchronize time between HUMS and VADR files. In

additon, the VADR and HUMS files are grouped into pairs that relate to the same flights. And finally, the files were grouped into triplets of files from HUMS, VADR, and ULLS-A for every actual flight. No only we fused data together, but many files containing zero flight information are set aside reducing the amount of analyzing data, and speeding up computations.

Once, the data sets are preprocessed, we considered each time series generated by respected sensors in the phase space for cleaning and further analysis.

5.2 Automated Non-flight Data Files Filtering Algorithm

As was mentioned previously, one of the main steps in preprocessing data, was to separate the files generated by HUMS and VADR into two categories: those that contain actual flight and others that were generated during the maintenance and testing procedures (Fig. 5.2).

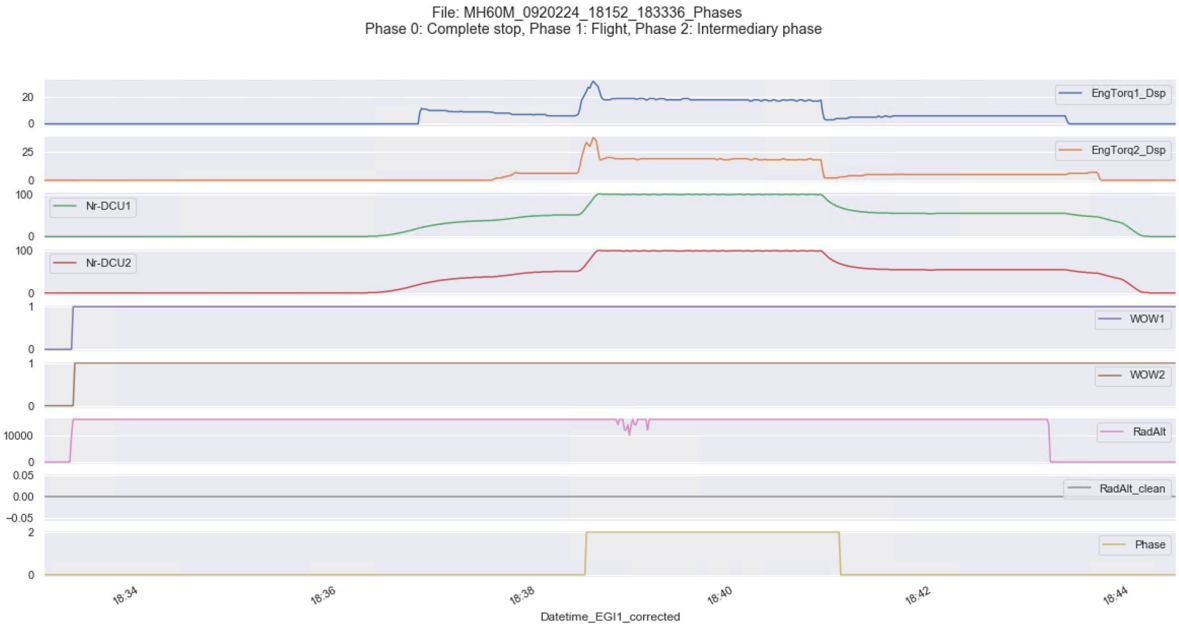


Figure 5.2. An example of a file that does not contain any actual flight information.

There are two sensonrs (WOW-1, WOW-2) installed on the wheels meauring the weight on wheels. However, they are not reliable in detecting whether the helicopter

is in the air or on the ground. Note, that during the very first few minuted of the file (Fig. 5.2) both WOW-1 and WOW-2 read 0, meaning there is no weight on the wheels and hence the helicopter is in the air. However, the engine torque (EngTorq1_Dsp and EngTroq2_Dsp) is zero implying that the Black Hawk is on the ground. Later, at around 18:34, weight on the wheels switches to a correct value 1 demonstrating that the helicopter is on the ground, but the radar altimeter (RadAlt) shows that the aircraft is approximately 17000 ft above the ground.

For the identification, we used 10 standard regime trigger time-series from VADR and HUMS files, such as the Air Speed, Rotor readings, Pitch, Roll, and Yaw rates, as well as the Weight-on-wheels. The problem of flight identification was solved in two steps:

(1) Using the *K-Means* alogorithm , we classified each second of the file data into three categories:
Stage 0 - Complete stop on the ground (partial rotor speed);
Stage 1 - Flight;
Stage 2 - Operation, or on the ground with 100% Nr.

(2) The stages were extrapolated to the rest of the data set with the *Support Vector Machine (SVM).*

The flight file along with the stages is shown in Fig. 5.3.

Our algorithm correctly identifies the flight stage, the complete stop on the ground with partial rotor activity, and the operation on the ground with 100% rotor activity. The automated assessment of flight stages is important for the data reduction and refinement, as it allows to perform the regime wise filtering of the data and creating the data manifolds. (Fig. 5.4). The regimes 80Kts, 120Kts, and 145Kts can be detected only during flights and are in the perfect agreement with the Stage 1 in our case.

After successfully training the SVM with accuracy of 100% for 80 Knots, 120 Knots, and 145 Knots - regimes, we calculated durations of every regime in each file. Our

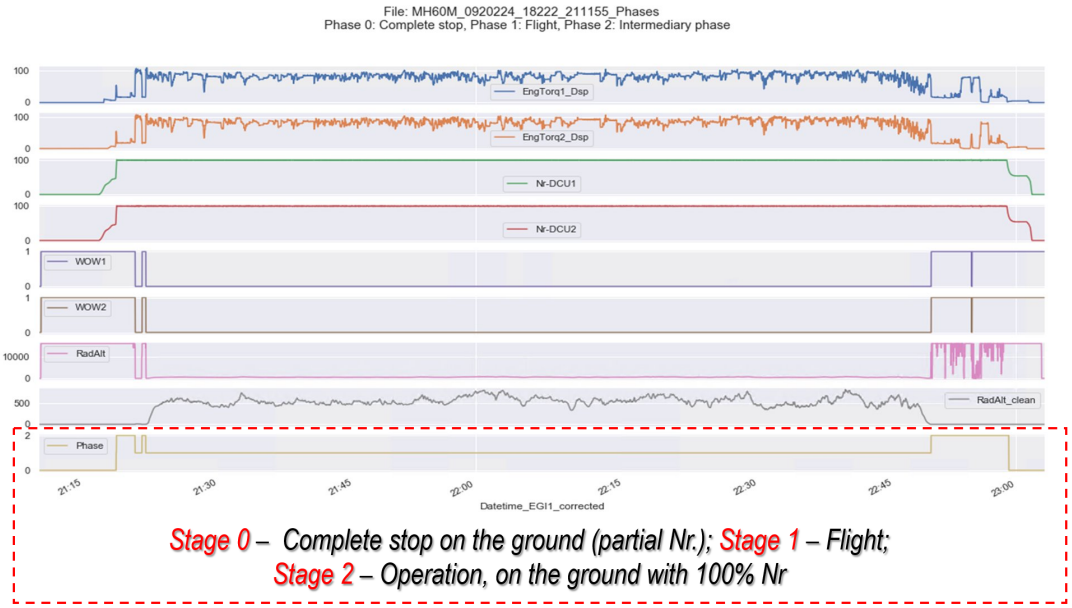


Figure 5.3. Machine Learning Algorithms (K-Means and SVM) applied to multiple functional time series for flight stage detection.

results are summarized in the following general table made available for all project partners.

Although our results perfectly match with the Army-defined flight operational regimes, our work was by no means redundant, because of army – defined regimes are known only for some time steps with varying time lags between them. But we need an exact and reliable flight stage identification for each and every time step for filtering the data regime-wise and constructing the relevant data manifolds.

It is worth mentioning that reliable assessment of flight stages and separation of flight and non-flight files allows for the massive functional data reduction. For example, for the tail number MH60M, we got 1,929 HUMS files. If we are interested in the analysis of flight data, we might focus at 520 files only that means the data reduction up to 73% in the number of files. Vice versa, if we are interested in the analysis of Conditional indicator readings while on the ground, we can effectively disregard the flight data that means the 70% reduction in the data volume (Fig. 5.5).

1,929 HUMS files

HUMS_file	HUMS_RGM_file	Phase0	Phase1	Phase2	FPG100	Hover	80Kts	120Kts	145Kts
N:\Raw Data\HUMS 2020-05-21\08-20152\191…	N:\Raw Data\HUMS 2020-05-21\08-20152\191217_13…	1	0	0	0	0	0	0	0
N:\Raw Data\HUMS 2020-05-21\08-20152\191…	N:\Raw Data\HUMS 2020-05-21\08-20152\191217_13…	1	0	0	0	0	0	0	0
N:\Raw Data\HUMS 2020-05-21\08-20152\191…	N:\Raw Data\HUMS 2020-05-21\08-20152\191217_13…	0.601342	0	0.398658	0.306897	0	0	0	0
N:\Raw Data\HUMS 2020-05-21\08-20152\191…	N:\Raw Data\HUMS 2020-05-21\08-20152\191217_13…	1	0	0	0	0	0	0	0
N:\Raw Data\HUMS 2020-05-21\08-20152\191…	N:\Raw Data\HUMS 2020-05-21\08-20152\191217_13…	0.15608	0.200171	0.643748	0.223465	0.330327	0.0487871	0.0219136	0.0101001
N:\Raw Data\HUMS 2020-05-21\08-20152\191…	N:\Raw Data\HUMS 2020-05-21\08-20152\191217_13…	1	0	0	0	0	0	0	0
N:\Raw Data\HUMS 2020-05-21\08-20152\191…	N:\Raw Data\HUMS 2020-05-21\08-20152\191217_13…	1	0	0	0	0	0	0	0
N:\Raw Data\HUMS 2020-05-21\08-20152\191…	N:\Raw Data\HUMS 2020-05-21\08-20152\191217_13…	1	0	0	0	0	0	0	0
N:\Raw Data\HUMS 2020-05-21\08-20152\191…	N:\Raw Data\HUMS 2020-05-21\08-20152\191217_13…	1	0	0	0	0	0	0	0
N:\Raw Data\HUMS 2020-05-21\08-20152\191…	N:\Raw Data\HUMS 2020-05-21\08-20152\191217_13…	1	0	0	0	0	0	0	0
N:\Raw Data\HUMS 2020-05-21\08-20152\191…	N:\Raw Data\HUMS 2020-05-21\08-20152\191217_13…	1	0	0	0	0	0	0	0
N:\Raw Data\HUMS 2020-05-21\08-20152\191…	N:\Raw Data\HUMS 2020-05-21\08-20152\191217_13…	1	0	0	0	0	0	0	0
N:\Raw Data\HUMS 2020-05-21\08-20152\191…	N:\Raw Data\HUMS 2020-05-21\08-20152\191217_13…	0.636101	0	0.363899	0.317255	0	0	0	0
N:\Raw Data\HUMS 2020-05-21\08-20152\191…	N:\Raw Data\HUMS 2020-05-21\08-20152\191217_13…	0.750962	0	0.249038	0.239946	0	0	0	0
N:\Raw Data\HUMS 2020-05-21\08-20152\191…	N:\Raw Data\HUMS 2020-05-21\08-20152\191217_13…	1	0	0	0	0	0	0	0
N:\Raw Data\HUMS 2020-05-21\08-20152\191…	N:\Raw Data\HUMS 2020-05-21\08-20152\191217_13…	0.0276346	0.920899	0.0514662	0.0375342	0.00281305	0.0122167	0.395194	0.0892943
N:\Raw Data\HUMS 2020-05-21\08-20152\191…	N:\Raw Data\HUMS 2020-05-21\08-20152\191217_13…	0.0394591	0.925431	0.0351099	0.0270574	0.0029231	0.00427222	0.412607	0.0651327
N:\Raw Data\HUMS 2020-05-21\08-20152\191…	N:\Raw Data\HUMS 2020-05-21\08-20152\191217_13…	0.0825626	0.824129	0.0933079	0.0598002	0.00570761	0.00181606	0.721365	0.00272409
N:\Raw Data\HUMS 2020-05-21\08-20152\191…	N:\Raw Data\HUMS 2020-05-21\08-20152\191217_13…	0.599469	0	0.400531	0.470097	0	0	0	0
N:\Raw Data\HUMS 2020-05-21\08-20152\191…	N:\Raw Data\HUMS 2020-05-21\08-20152\191217_13…	0.882141	0	0.117859	0.0894871	0	0	0	0
N:\Raw Data\HUMS 2020-05-21\08-20152\191…	N:\Raw Data\HUMS 2020-05-21\08-20152\191217_13…	0.0905125	0.591226	0.318261	0.252386	0.0297632	0.155355	0.0252537	0
N:\Raw Data\HUMS 2020-05-21\08-20152\191…	N:\Raw Data\HUMS 2020-05-21\08-20152\191217_13…	0.27557	0.489821	0.23461	0.118822	0.0575058	0.0937644	0.0182448	0
N:\Raw Data\HUMS 2020-05-21\08-20152\191…	N:\Raw Data\HUMS 2020-05-21\08-20152\191217_13…	1	0	0	0	0	0	0	0
N:\Raw Data\HUMS 2020-05-21\08-20152\191…	N:\Raw Data\HUMS 2020-05-21\08-20152\191217_13…	1	0	0	0	0	0	0	0

Figure 5.4. Perfect matching of the flight stages with Army-defined flight operational regimes.

5.3 Synchronization of Flight and Maintenance data systems

Upon completion of the first preprocessing stage, we separated files containing flight data from files that bear little to none useful information. However, flight data come from different sources: VADR, HUMS, and ULLS-A which are not synchronized that therefore as the next step we need to determine correspondence between data sources.

We begin by restoring the actual time line in the VADR data sets. VADR data is being recorded once the engine is switched on. However, time data provided by two GPS units (EGI-1, EGI-2) onboard becomes available only few minutes later, also sometimes the connection with the GPS satellites can be lost or the GPS units may fail.

- First, we developed the CLEAR_INNER_TIME algorithm for cleaning erroneous, repeating, and missing timestamps along the series of Conditional Indicator readings in the VADR files

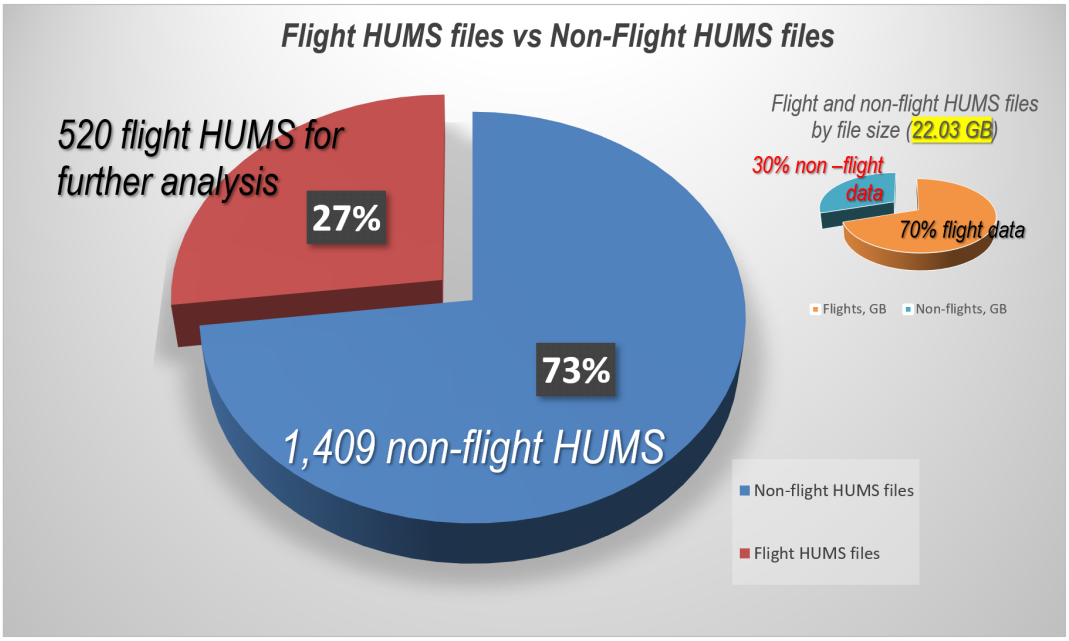


Figure 5.5. Distribution of flight and non-flight HUMS files.

- Second, We developed the CORR_TIME_GPS algorithm for Interpolating missing and correcting erroneous GPS time readings in the VADR files.

After all time readings were cleaned, corrected, and interpolated, we were able to merge all relevant VADR files and create operational aircraft history based on the corrected EGI time (Fig. 5.6).

Low Data Organization Quality is the major obstacle for the reliable file attribution and synchronization in the Flight and Maintenance data systems. There is neither one -to-one correspondence between the VADR and HUMS files, nor one -to-one correspondence between the HUMS and ULLS files:

- A HUMS file might correspond to a part of a VADR file
- Several VADR files can be attributed to a single HUMS file
- HUMS and VADR readings are desynchronized in time and mismatched in amplitude

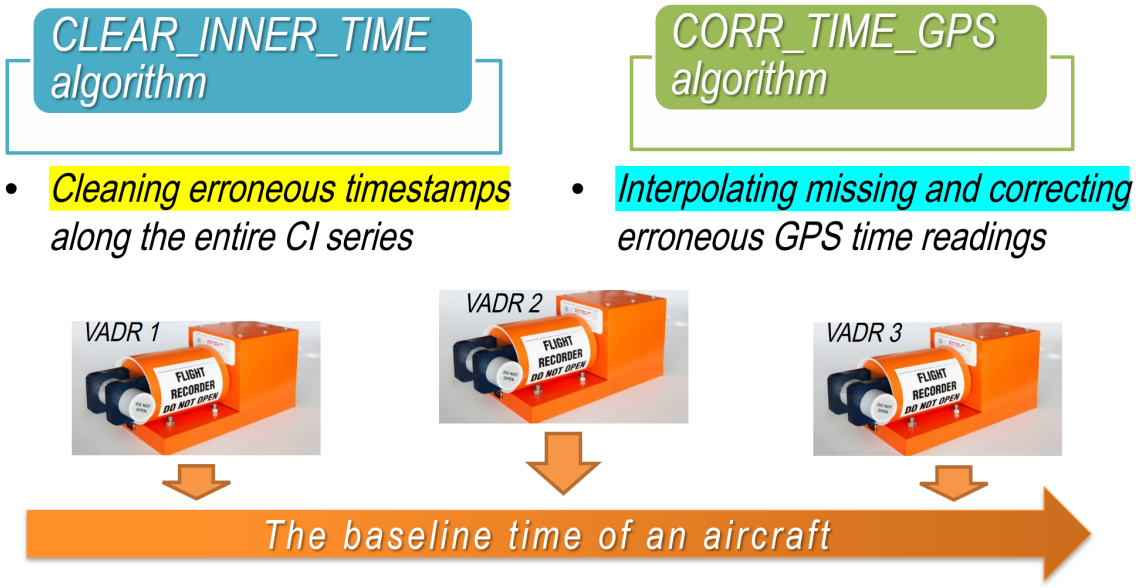


Figure 5.6. Baseline Time of an Aircraft.

Finally, there might be missing files and incomplete data collections (Fig. 5.7).

	VADR	HUMS	ULLS
VADR	100%	63%	54%
HUMS	63%	100%	29%
ULLS	54%	29%	100%

Figure 5.7. File attribution problem. Matching parts of HUMS, VADR data sets.

To solve the VADR-HUMS Attribution and Synchronization problems reliably and efficiently, we have developed and used a series of algorithms based on three principles.

First, sampling frequencies in the HUMS file readings vary, ranging from 0.1 to 10 Hz. The VADR files have the standard sampling frequencies of 16 Hz.

For the attribution of VADR and HUMS files, and only for that, we perform DOWNSAMPLING of VADR and HUMS time-series uniformly to the sampling frequency of 1 Hz.

It is important to mention that downsampling reduces the number of points in the time series, and therefore speeds the attribution algorithms running over the huge annual aircraft operational history up to 10 times (Fig. 5.8).

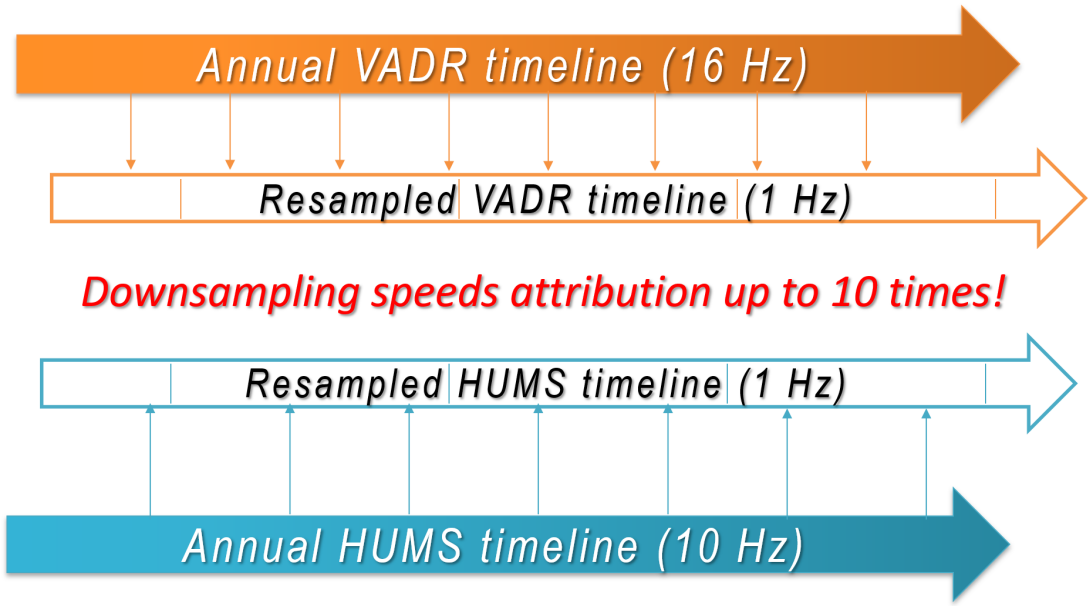


Figure 5.8. VADR-HUMS attribution and synchornization: Step 1 Downsampling.

Our second "knowhow" is the triple iterative synchronization refinement approach to HUMS -VADR attribution and synchronization. We trace an initial similarity match with the use of ultrafast methods, and then refine our hypothesis with fine methods, matching the files exactly.

On the first stage, we use the MASS, Mueen's Ultrafast Similarity Search Algorithm [144], scaled to trillions of observations under Dynamic Time Warping and Early

Abandoning technique. The algorithm is ultrafast, especially if the modern versions of this algorithm using the graphic processors are used. We implement this algorithm for superficial similarity matching over the annual operational history of the aircraft.

On the second stage, we use the SSIM, Maximizing Structural Similarity algorithm [145] to justify and refine the initial hypothesis on the VADR-HUMS matching. The SSIM algorithm is fine but relatively slow, so that we use this algorithm to justify the discovered matching and approximating the time lag between the HUMS and VADR time series in a range of a minute.

Finally, on the third stage, we use the minimization of the Mean Squared Error for the fine tuning of time lags between the VADR and HUMS time series on the time scale of one second. The Mean Squared Error works fine but is unstable if used on long time scales (Fig. 5.9).

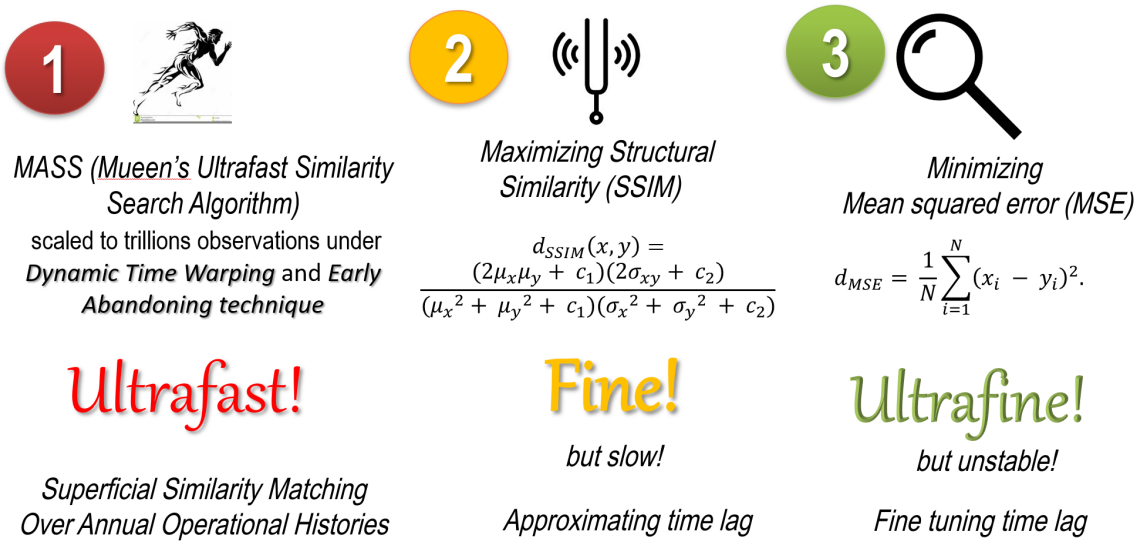


Figure 5.9. VADR-HUMS attribution and synchornization: Step 2 Triple iterative synchronization refinement.

Our last "knowhow" technology is the two-way move of the algorithm, in which we find the 1-to-1 VADR – HUMS matching on the forward move and discover all remaining VADR files corresponding to the identified HUMS file on the backward move.

We start with the VADR file and use the MASS ultrafast similarity search algorithm to superficially match the VADR file with a point in the annual HAMS history.

Then, we use the SSIM algorithm for approximating a time lag between the files up to a minute. Finally, we refine the time lag by minimizing the Mean Squared Error between the VADR and HUMS time series.

After the matching point is found, we check whether the discovered HUMS file ends simultaneously with the matched VADR file. If it is not the case, there might be other VADR files corresponding to the given HUMS file, and then we start the backward move.

Again, we start with the use of the MASS algorithm to match the last identified point in the HUMS file against all available VADR files. After preliminary matching is done, we use the SSIM algorithm to find the corresponding time lag. If the HUMS file is not finished yet, we repeat the previous steps until all correspondences with other VADR files are found.

Starting from a VADR file, our algorithm takes from 4 – to 5 seconds to uncover the whole cluster of VADR-HUMS correspondences even if no graphic processor involved (Fig. 5.10).

It is important to mention that as HUMS and VADR readings are desynchronized in time and mismatched in amplitude, the use of a single algorithm is not enough for reliable attribution of barely similar HUMS and VADR files.

For example, if we minimize the Mean Squared Error between the VADR and HUMS time series alone, we might get an amazing time lag of 310 days, 1 hour, 25 minutes, and 10 seconds. However, if we check the MASS and SSIM similarity values for this match, we will find that these time series are dissimilar (Fig. 5.11).

A reliable decision on the correspondence between the VADR and HUMS files should be taken on the base of three parameters simultaneously.

Our three-stage matching algorithm produces a table of file correspondences and time lags, especially useful for large databases when visual or other pairwise comparisons of files are not possible.

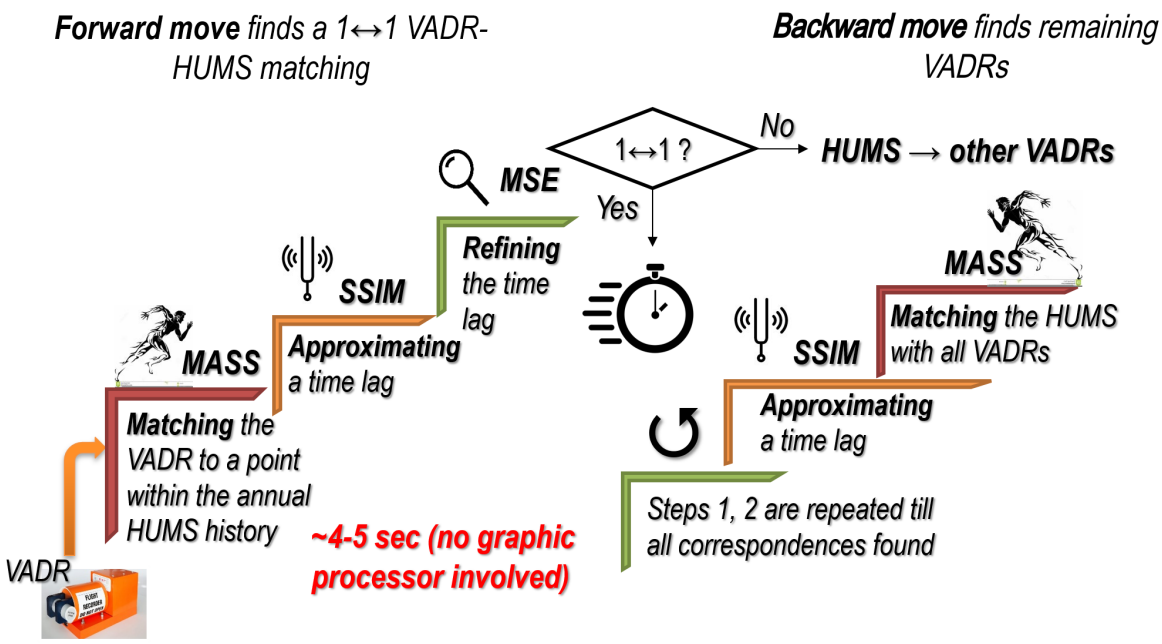


Figure 5.10. VADR-HUMS attribution and synchornization: Step 3 Two-way move of the algorithm.

Vadr_Files	File_size_MB	Hums_Air	R_Air_missing_sam	Hums_Air_corresp	Hums_Air_Dist	Hums_Air_SSIM	ms_Air_SSIM_epsil	lums_Air_Time_La
N:\Raw Data\avx_vad…	398.283	N:\Raw Data\HUMS …	No	1	0.850186	0.391428	0.917831	0 days 00:01:26
N:\Raw Data\avx_vad…	6.064	Too many zeros to …	No					
N:\Raw Data\avx_vad…	282.88	N:\Raw Data\HUMS …	No	1	9.53818	0.0934673	0.115489	-310 days +01:25:10

Figure 5.11. VADR-HUMS attribution and synchornization. Intermediate results using MASS algorithm and SSIM.

The correspondence table available for all partners contains information on the beginning and the end of matching, synchronization time lag, and overall similarity index of all related files.

5.4 Missing data restoratiion via VADR/HUMS/ULLS Cross-validation

Fusion of the Flight and Maintenance data systems inevitably involves not only technical, but also a human factor into analysis.

FORWARD RUN	VADR File	Beginning of flight (VADR)	End of flight (VADR)	Matching HUMS File	Beginning of match	Eng of Match	Time delta	SSIM
2	N:\Raw Data\avx_vadr\avx2\MH60M_1320567_18026_162544.csv	1/26/2018 16:25	1/26/2018 18:54	N:\Raw Data\HUMS 2020-05-21\13-20567\191216_195359\MH60M-AED_13-20567_2018_0126_155953Z\BM001_XM.txt	1/26/2018 16:25	1/26/2018 18:37	0 days 00:02:07.000000000	0.9625723

BACKWARD RUN	HUMS File	Beginning of flight (VADR)	End of flight (VADR)	Matching VADR File	Beginning of match	Eng of Match	Time delta	SSIM
6	N:\Raw Data\HUMS 2020-05-21\13-20567\191216_195359\MH60M-AED_13-20567_2018_0126_155953Z\BM001_XM.txt	1/26/2018 16:03	1/26/2018 18:37	N:\Raw Data\avx_vadr\avx2\MH60M_1320567_18026_162544.csv	1/26/2018 16:25	1/26/2018 18:37	0 days 00:02:07.000000000	0.9625723
	Union of time intervals (duration of combined flight data)	1/26/2018 16:03	1/26/2018 18:54					

Tail Number 0820152

Figure 5.12. VADR-HUMS attribution and synchornization. Final results.

- Pilots may combine several consecutive flights into a single entry in a ULLS logbook (Fig. 5.16a).
- Pilots might change during the same VADR/HUMS flight (Fig. 5.16b).
- Duration of flights in an ULLS file may be up to 30%-40% off their actual lengths.
- Finally, Some HUMS, VADR, or ULLS files may just be missing (Fig. 5.16c).

In the diagram, we have presented the matching over the Flight and Maintenance data systems. The heights of the vertical bars indicate the numbers of matched files in different days over the annual aircraft history.

In some days, the flight hours indicated in ULLS and Flight files match exactly, but in the most of cases the numbers of flight and maintenance files do not coincide and, moreover, for some months, either flight, or maintenance data is missing.

For instance, ULLS data was provided only for the period of a Fiscal year, from 01/01/2018 to 09/30/2018. Some chunks of HUMS, VADR files were also missing (Fig. 5.13).

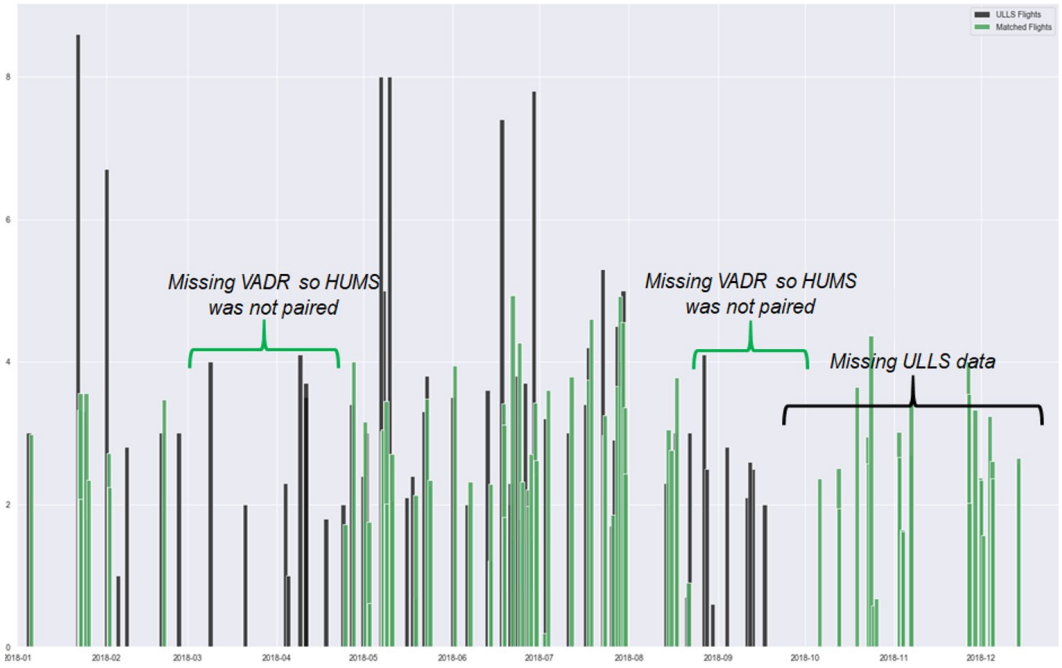
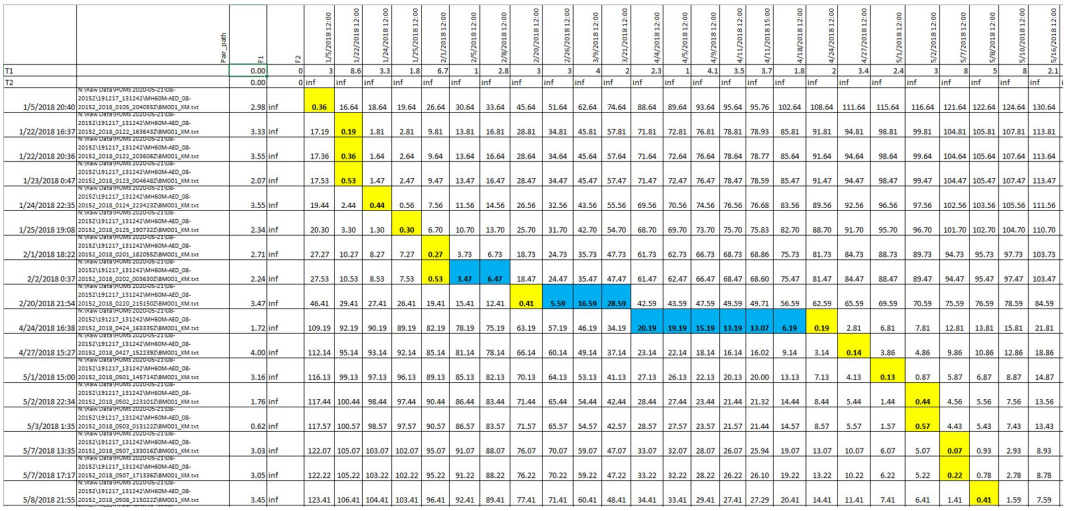


Figure 5.13. Distribution of Flight and Maintenance data sets of the year of 2018.

To perform the Fusion of the Flight and Maintenance data system files, we modified the Needleman–Wunsch algorithm originally developed in bioinformatics to align protein and nucleotide sequences [146].

Upon defining a metric on the sets of ULLS and VADR files, we computed a "cost matrix" that contains 'distances' between every ULLS and VADR files represented by the table. We align the files by reading this table from the left top corner, one cell down, to the right or diagonally - at each step choosing the next cell that has the lowest 'cost'. The resulting alignment path, colored in yellow and blue, is shown in the figure. If a cell has a value less than 1, then the files are related, if not, then we cannot match those VADR and ULLS files.

After we run the alignment algorithm to match VADR and ULLS flights, we had some ULLS flights unassigned. Then, we repeated this algorithm for remaining ULLS records and over unassigned HUMS flights. All results were logged in a table (available



	Pair_path	F1	F2	1/5/2018 12:00	1/22/2018 12:00	1/24/2018 12:00	1/25/2018 12:00	2/1/2018 12:00	2/5/2018 12:00	2/8/2018 12:00	2/26/2018 12:00	2/28/2018 12:00	3/9/2018 12:00	3/21/2018 12:00	4/4/2018 12:00	4/5/2018 12:00	4/9/2018 12:00	4/11/2018 12:00	4/11/2018 15:00	4/18/2018 12:00	4/24/2018 12:00	4/27/2018 12:00	5/1/2018 12:00	5/22/2018 12:00	5/27/2018 12:00	5/28/2018 12:00	5/10/2018 12:00	5/16/2018 12:00
T1		0.00	0	3	8.6	3.3	1.8	6.7	1	2.8	3	3	4	2	2.3	1	4.1	3.5	3.7	1.8	2	3.4	2.4	3	8	5	8	2.1
T2		0.00	0	inf	inf	inf	inf	inf	inf	inf	inf	inf	inf	inf	inf	inf	inf	inf	inf	inf	inf	inf	inf	inf	inf	inf	inf	inf
1/5/2018 20:40	N:\Raw Data\HUMS 2020\05-21\08-20152\191217_131242\MH60M-AED_08-20152_2018_0109_204655ZBM001_XM.txt	2.98	inf	0.36	16.64	18.64	19.64	26.64	30.64	33.64	45.64	51.64	62.64	74.64	88.64	89.64	93.64	95.64	95.76	102.64	108.64	111.64	115.64	116.64	121.64	122.64	124.64	130.64
1/22/2018 16:37	N:\Raw Data\HUMS 2020\05-21\08-20152\191217_131242\MH60M-AED_08-20152_2018_0122_163843ZBM001_XM.txt	3.33	inf	17.19	0.19	1.81	2.81	9.81	13.81	16.81	28.81	34.81	45.81	57.81	71.81	72.81	76.81	78.81	78.93	85.81	91.81	94.81	98.81	99.81	104.81	105.81	107.81	113.81
1/22/2018 20:36	N:\Raw Data\HUMS 2020\05-21\08-20152\191217_131242\MH60M-AED_08-20152_2018_0122_203606ZBM001_XM.txt	3.55	inf	17.36	0.36	1.64	2.64	9.64	13.64	16.64	28.64	34.64	45.64	57.64	71.64	72.64	76.64	78.64	78.77	85.64	91.64	94.64	98.64	99.64	104.64	105.64	107.64	113.64
1/23/2018 0:47	N:\Raw Data\HUMS 2020\05-21\08-20152\191217_131242\MH60M-AED_08-20152_2018_0123_004648ZBM001_XM.txt	2.07	inf	17.53	0.53	1.47	2.47	9.47	13.47	16.47	28.47	34.47	45.47	57.47	71.47	72.47	76.47	78.47	78.59	85.47	91.47	94.47	98.47	99.47	104.47	105.47	107.47	113.47
1/24/2018 22:35	N:\Raw Data\HUMS 2020\05-21\08-20152\191217_131242\MH60M-AED_08-20152_2018_0124_223423ZBM001_XM.txt	3.55	inf	19.44	2.44	0.44	0.56	7.56	11.56	14.56	26.56	32.56	43.56	55.56	69.56	70.56	74.56	76.56	76.68	83.56	89.56	92.56	96.56	97.56	102.56	103.56	105.56	111.56
1/25/2018 19:08	N:\Raw Data\HUMS 2020\05-21\08-20152\191217_131242\MH60M-AED_08-20152_2018_0125_190732ZBM001_XM.txt	2.34	inf	20.30	3.30	1.30	0.30	6.70	10.70	13.70	25.70	31.70	42.70	54.70	68.70	69.70	73.70	75.70	75.83	82.70	88.70	91.70	95.70	96.70	101.70	102.70	104.70	110.70
2/1/2018 18:22	N:\Raw Data\HUMS 2020\05-21\08-20152\191217_131242\MH60M-AED_08-20152_2018_0201_182055ZBM001_XM.txt	2.71	inf	27.27	10.27	8.27	7.27	0.27	3.73	6.73	18.73	24.73	35.73	47.73	61.73	62.73	66.73	68.73	68.86	75.73	81.73	84.73	88.73	89.73	94.73	95.73	97.73	103.73
2/2/2018 0:37	N:\Raw Data\HUMS 2020\05-21\08-20152\191217_131242\MH60M-AED_08-20152_2018_0202_003630ZBM001_XM.txt	2.24	inf	27.53	10.53	8.53	7.53	0.53	3.47	6.47	18.47	24.47	35.47	47.47	61.47	62.47	66.47	68.47	68.60	75.47	81.47	84.47	88.47	89.47	94.47	95.47	97.47	103.47
2/20/2018 21:54	N:\Raw Data\HUMS 2020\05-21\08-20152\191217_131242\MH60M-AED_08-20152_2018_0220_215152ZBM001_XM.txt	3.47	inf	46.41	29.41	27.41	26.41	19.41	15.41	12.41	0.41	5.59	16.59	28.59	42.59	43.59	47.59	49.59	49.71	56.59	62.59	65.59	69.59	70.59	75.59	76.59	78.59	84.59
4/24/2018 16:38	N:\Raw Data\HUMS 2020\05-21\08-20152\191217_131242\MH60M-AED_08-20152_2018_0424_163335ZBM001_XM.txt	1.72	inf	109.19	92.19	90.19	89.19	82.19	78.19	75.19	63.19	57.19	46.19	34.19	20.19	19.19	15.19	13.19	13.07	6.19	0.19	2.81	6.81	7.81	12.81	13.81	15.81	21.81
4/27/2018 15:27	N:\Raw Data\HUMS 2020\05-21\08-20152\191217_131242\MH60M-AED_08-20152_2018_0427_152239ZBM001_XM.txt	4.00	inf	112.14	95.14	93.14	92.14	85.14	81.14	78.14	66.14	60.14	49.14	37.14	23.14	22.14	18.14	16.14	16.02	9.14	3.14	0.14	3.86	4.86	9.86	10.86	12.86	18.86
5/1/2018 15:00	N:\Raw Data\HUMS 2020\05-21\08-20152\191217_131242\MH60M-AED_08-20152_2018_0501_145714ZBM001_XM.txt	3.16	inf	116.13	99.13	97.13	96.13	89.13	85.13	82.13	70.13	64.13	53.13	41.13	27.13	26.13	22.13	20.13	20.00	13.13	7.13	4.13	0.13	0.87	5.87	6.87	8.87	14.87
5/2/2018 22:34	N:\Raw Data\HUMS 2020\05-21\08-20152\191217_131242\MH60M-AED_08-20152_2018_0502_223101ZBM001_XM.txt	1.76	inf	117.44	100.44	98.44	97.44	90.44	86.44	83.44	71.44	65.44	54.44	42.44	28.44	27.44	23.44	21.44	21.32	14.44	8.44	5.44	1.44	0.44	4.56	5.56	7.56	13.56
5/3/2018 1:35	N:\Raw Data\HUMS 2020\05-21\08-20152\191217_131242\MH60M-AED_08-20152_2018_0503_013122ZBM001_XM.txt	0.62	inf	117.57	100.57	98.57	97.57	90.57	86.57	83.57	71.57	65.57	54.57	42.57	28.57	27.57	23.57	21.57	21.44	14.57	8.57	5.57	1.57	0.57	4.43	5.43	7.43	13.43
5/7/2018 13:35	N:\Raw Data\HUMS 2020\05-21\08-20152\191217_131242\MH60M-AED_08-20152_2018_0507_133016ZBM001_XM.txt	3.03	inf	122.07	105.07	103.07	102.07	95.07	91.07	88.07	76.07	70.07	59.07	47.07	33.07	32.07	28.07	26.07	25.94	19.07	13.07	10.07	6.07	5.07	0.07	0.93	2.93	8.93
5/7/2018 17:17	N:\Raw Data\HUMS 2020\05-21\08-20152\191217_131242\MH60M-AED_08-20152_2018_0507_171336ZBM001_XM.txt	3.05	inf	122.22	105.22	103.22	102.22	95.22	91.22	88.22	76.22	70.22	59.22	47.22	33.22	32.22	28.22	26.22	26.10	19.22	13.22	10.22	6.22	5.22	0.22	0.78	2.78	8.78
5/8/2018 21:55	N:\Raw Data\HUMS 2020\05-21\08-20152\191217_131242\MH60M-AED_08-20152_2018_0508_215022ZBM001_XM.txt	3.45	inf	123.41	106.41	104.41	103.41	96.41	92.41	89.41	77.41	71.41	60.41	48.41	34.41	33.41	29.41	27.41	27.29	20.41	14.41	11.41	7.41	6.41	1.41	0.41	1.59	7.59

Figure 5.14. Application of Needleman algorithm to fusion of Flight and Maintenance data systems.

on the AWS cloud). If an ULLS flight has a matching pair (VADR, HUMS), then, the time lag between VADR and HUMS is provided.

Matching data contains the beginning, ending, and duration of matching, along with the synchronization time lags.

Our algorithm has detected all instances when

- The ULLS records showing that pilots might combine several VADR, HUMS flights into one entry (Fig. 5.16a).
- Pilots might change in the middle of operation, meaning that there could be several entries in ULLS logbooks, for the same VADR-HUMS files pair (Fig. 5.16b).
- If a HUMS file is missing, only VADR files get associated to ULLS records or vice versa (Fig. 5.16c).

ULLS_flight	ULLS_duration	VADR_file	HUMS_file	Union_beg	Union_end	Duration	Timedelta
1/5/2018 15:00	3	MH60M_0820152_18005_205633	MH60M-AED_08-20152_2018_0105_204055Z	1/5/2018 20:40	1/5/2018 23:39	2.98	0 days 00:00:19.000000000
1/22/2018 11:00	8.6	MH60M_0820152_18022_164002	MH60M-AED_08-20152_2018_0122_163643Z	1/22/2018 16:37	1/22/2018 19:57	3.33	0 days 00:01:05.000000000
1/22/2018 11:00	8.6	MH60M_0820152_18022_203738	MH60M-AED_08-20152_2018_0122_203606Z	1/22/2018 20:36	1/23/2018 0:10	3.55	0 days 00:01:06.000000000
1/22/2018 11:00	8.6	MH60M_0820152_18023_004839	MH60M-AED_08-20152_2018_0123_004648Z	1/23/2018 0:47	1/23/2018 2:51	2.07	0 days 00:01:06.000000000
1/24/2018 16:00	3.3	MH60M_0820152_18024_223544	MH60M-AED_08-20152_2018_0124_223423Z	1/24/2018 22:35	1/25/2018 2:08	3.55	0 days 00:01:09.000000000
1/25/2018 12:45	1.8	MH60M_0820152_18025_191101	MH60M-AED_08-20152_2018_0125_190732Z	1/25/2018 19:08	1/25/2018 21:28	2.34	0 days 00:01:09.000000000
2/1/2018 12:40	6.7	MH60M_0820152_18032_182311	MH60M-AED_08-20152_2018_0201_182055Z	2/1/2018 18:22	2/1/2018 21:04	2.71	0 days 00:01:26.000000000
2/1/2018 12:40	6.7	MH60M_0820152_18032_220854		2/1/2018 22:08	2/2/2018 0:04	1.94	
2/1/2018 12:40	6.7	MH60M_0820152_18033_003809	MH60M-AED_08-20152_2018_0202_003630Z	2/2/2018 0:37	2/2/2018 2:52	2.24	0 days 00:01:25.000000000
2/5/2018 12:30	1	MH60M_0820152_18036_183822		2/5/2018 18:38	2/5/2018 19:33	0.91	
2/8/2018 17:00	2.8	MH60M_0820152_18039_231403		2/8/2018 23:14	2/9/2018 1:54	2.67	
2/20/2018 16:20	3	MH60M_0820152_18051_221541	MH60M-AED_08-20152_2018_0220_215150Z	2/20/2018 21:54	2/21/2018 1:22	3.47	0 days 00:01:58.000000000
3/9/2018 12:30	4		MH60M-AED_08-20152_2018_0309_165337Z	3/9/2018 16:54	3/9/2018 22:06	5.20	
3/21/2018 10:45	2		MH60M-AED_08-20152_2018_0321_150754Z	3/21/2018 15:08	3/21/2018 17:33	2.41	
4/4/2018 20:00	2.3		MH60M-AED_08-20152_2018_0405_003917Z	4/5/2018 0:40	4/5/2018 1:15	0.59	
4/4/2018 20:00	2.3		MH60M-AED_08-20152_2018_0405_020004Z	4/5/2018 2:00	4/5/2018 3:56	1.94	
4/5/2018 16:00	1		MH60M-AED_08-20152_2018_0405_210641Z	4/5/2018 21:07	4/5/2018 22:27	1.34	
4/9/2018 10:30	4.1		MH60M-AED_08-20152_2018_0409_145212Z	4/9/2018 14:53	4/9/2018 19:31	4.63	
4/11/2018 17:30	3.7		MH60M-AED_08-20152_2018_0411_144958Z	4/11/2018 14:50	4/11/2018 18:58	4.13	
4/18/2018 6:50	1.8		MH60M-AED_08-20152_2018_0418_113305Z	4/18/2018 11:33	4/18/2018 13:39	2.09	
4/24/2018 10:30	2	MH60M_0820152_18114_164037	MH60M-AED_08-20152_2018_0424_163335Z	4/24/2018 16:38	4/24/2018 18:21	1.72	0 days 00:03:43.000000000
4/24/2018 10:30	2	MH60M_0820152_18114_184109		4/24/2018 18:40	4/24/2018 18:50	0.17	
4/27/2018 11:00	3.4	MH60M_0820152_18117_155928	MH60M-AED_08-20152_2018_0427_152239Z	4/27/2018 15:27	4/27/2018 19:27	4.00	0 days 00:03:50.000000000
5/1/2018 10:48	2.4	MH60M_0820152_18121_153938	MH60M-AED_08-20152_2018_0501_145714Z	5/1/2018 15:00	5/1/2018 18:10	3.16	0 days 00:03:56.000000000
5/2/2018 17:30	3	MH60M_0820152_18123_142425		5/3/2018 14:23	5/3/2018 14:30	0.12	
5/2/2018 17:30	3	MH60M_0820152_18123_014439	MH60M-AED_08-20152_2018_0503_013122Z	5/3/2018 1:35	5/3/2018 2:12	0.62	0 days 00:03:57.000000000
5/2/2018 17:30	3	MH60M_0820152_18123_010217		5/3/2018 1:01	5/3/2018 1:13	0.20	
5/2/2018 17:30	3	MH60M_0820152_18122_223504	MH60M-AED_08-20152_2018_0502_223101Z	5/2/2018 22:34	5/3/2018 0:20	1.76	0 days 00:03:57.000000000
5/2/2018 17:30	3	MH60M_0820152_18122_213337		5/2/2018 21:33	5/2/2018 22:04	0.51	
5/7/2018 9:00	8	MH60M_0820152_18127_204934		5/7/2018 20:48	5/7/2018 22:37	1.81	
5/7/2018 9:00	8	MH60M_0820152_18127_233122		5/7/2018 23:30	5/7/2018 23:35	0.07	

Figure 5.15. Final results of fusion of the Flight and Maintenance data systems.

a.)

1/22/2018 11:00	8.6	MH60M_0820152_18022_164002	MH60M-AED_08-20152_2018_0122_163643Z	1/22/2018 16:37	1/22/2018 19:57	3.33	0 days 00:01:05.000000000
1/22/2018 11:00	8.6	MH60M_0820152_18022_203738	MH60M-AED_08-20152_2018_0122_203606Z	1/22/2018 20:36	1/23/2018 0:10	3.55	0 days 00:01:06.000000000
1/22/2018 11:00	8.6	MH60M_0820152_18023_004839	MH60M-AED_08-20152_2018_0123_004648Z	1/23/2018 0:47	1/23/2018 2:51	2.07	0 days 00:01:06.000000000

b.)

6/21/2018 11:30	2	MH60M_0820152_18172_170539	MH60M-AED_08-20152_2018_0621_164225Z	6/21/2018 16:48	6/21/2018 21:44	4.93	0 days 00:04:50.000000000
6/21/2018 14:30	2.3	MH60M_0820152_18172_170539	MH60M-AED_08-20152_2018_0621_164225Z	6/21/2018 16:48	6/21/2018 21:44	4.93	0 days 00:04:50.000000000

c.)

4/9/2018 10:30	4.1		MH60M-AED_08-20152_2018_0409_145212Z	4/9/2018 14:53	4/9/2018 19:31	4.63	
4/11/2018 17:30	3.7		MH60M-AED_08-20152_2018_0411_144958Z	4/11/2018 14:50	4/11/2018 18:58	4.13	

Figure 5.16. a.) Entries in which all flights we combines by a pilot to a single entry found by the algorithm; b.) Entries in which one flight we split by a pilot to several entries found by the algorithm; c.) Entries in which flight data was stored only by either HUMS or VADR.

Even if some VADR or HUMS files were missing, after running the algorithm, we managed to use either HUMS data or VADR data or both if available to restore the actual flight history.

The developed algorithm helps us not only for matching ULLS, HUMS, and VADR files, but also for restoring missing flight history by patching the missing or corrupted data and filtering functionally irrelevant data, significantly reducing the data volumes (Fig. 5.18).

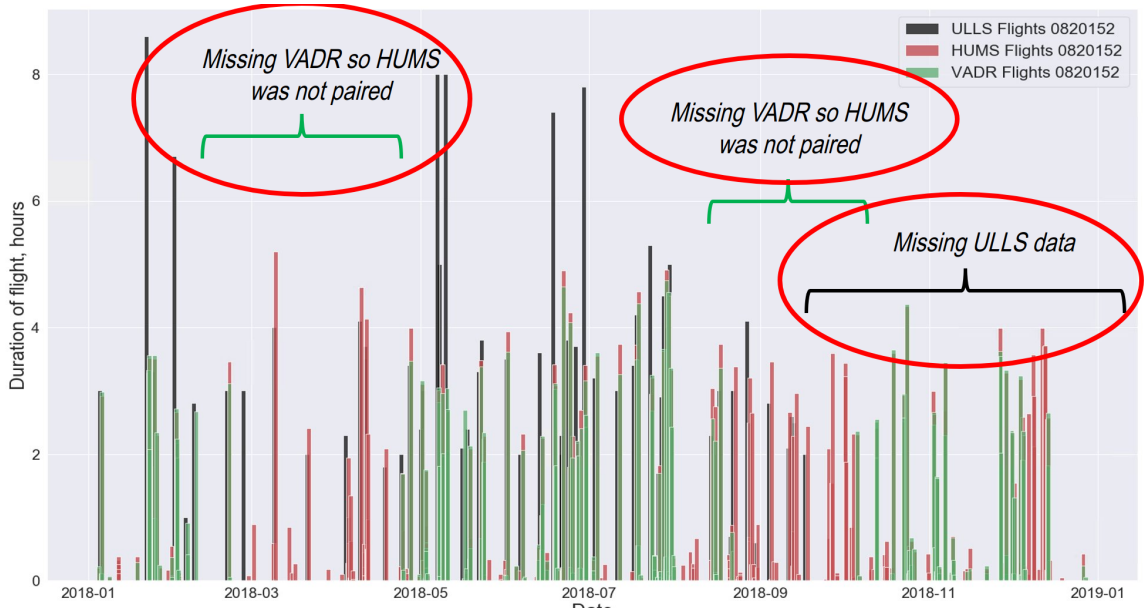


Figure 5.17. Distribution of HUMS (red bars), VADR (green bars), and ULLS-A (black bars) files on a time line.

5.5 Time series healing and data manifolds.

In the previous steps, we restored flight history and defined correspondence between three data collecting systems: HUMS, VADR, and ULLS-A. At this point we shift our attention to the times series in those files. There is a plethora of different methods that can be efficiently used to clean time series [147]. We propose a different approach by submering time series into a phase space.

Fig. 5.19 shows the time series generated by a fuel pump sensor. There are plenty of erroneous values. Many common cleaning methods including neural networks do not efficiently solve the problem due to a high volume of erroneous readings.

Under our approach, the time series is taken to the two-dimensional phace space (position vs velocity) as shown in Fig. 5.20). The points of the time series corresponding to the rapid drops and increases are aling with the line $y = x$ and $x = 0$ (excluding the origin). In general, such outliers lie on the line $y = \nu x$ where ν is the frequency of data. In our case, frequency is 1Hz. Regular ponts that form a regular data manifold are inside the green rectangle (Fig. 5.20).

a.)

b.)

VADR	Beg of flight	End of flight	HUMS	From	To	Timedelta	SSIM	Union_beg	Union_end
N:\Raw Data\avx_vadr\avx2\MH60M_1320567_18026_162544.csv	1/26/2018 16:25	1/26/2018 18:54	N:\Raw Data\HUMS 2020-05-21\13-20567\191216_195359\MH60M-AED_13-20567_2018_0126_155953Z\BM001_XM.txt	1/26/2018 16:25	1/26/2018 18:37	0 days 00:02:07.000000	0.962572254	1/26/2018 16:03	1/26/2018 18:54
VADR file that we are trying to match to some HUMS file	Beginning EGI time of the VADR file **(t2)**	Ending EGI time of the VADR file **(t4)**	HUMS file that matches VADR file under our assumptions	The EGI time point at which matching begins **(t2)**	The EGI time point at which matching ends **(t3)**	Timedelta between VADR and original HUMS time	Similarity index	The EGI time point at which union of VADR and HUMS begins **(t1)**	The EGI time point at which union of VADR and HUMS ends **(t4)**

Figure 5.18. a.) Complete time series of a flight restored from both VADR and HUMS; b.) An example of a table containing all information about restored flight, including time delta between HUMS and VADR and total duration of the flight.

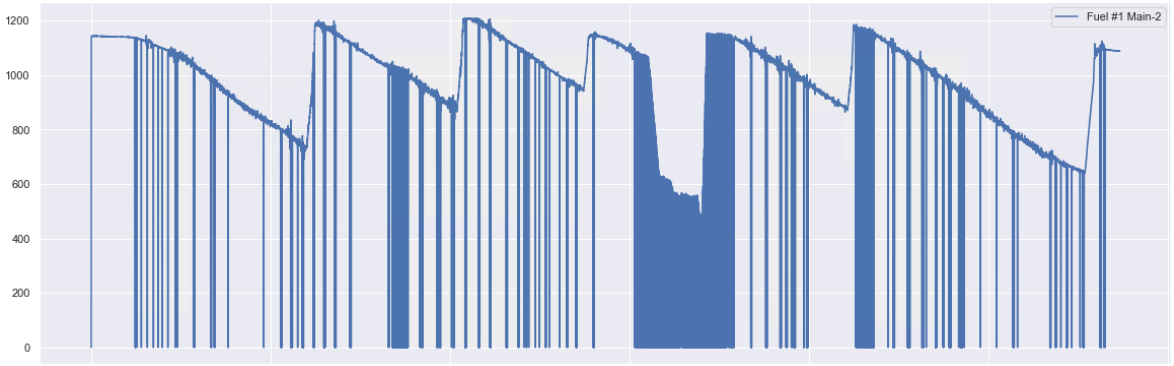


Figure 5.19. Gallons of fuel per hour readings generated by a fuel pump (Original data readings).

Once the outliers were identified in the phase space, the irregular point can be taken back to the regular manifold producing the clean time series as shown in Fig. 5.21).

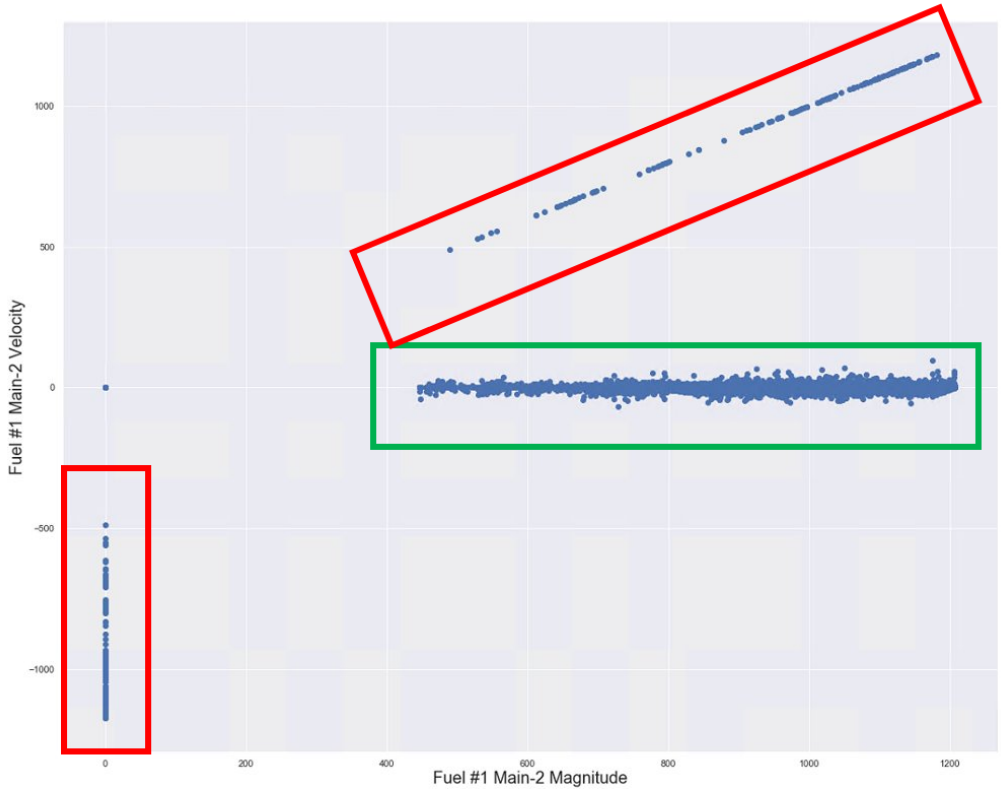


Figure 5.20. Gallons of fuel per hour readings generated by a fuel pump (Original data readings) in the phase space. The points in the green rectangle are the regular point of the data manifold. The points inside the red rectangles are the outliers generated by the drops in the readings.

We obtained similar results with other time series, for example, Air Speed provided by the pitot tubes installed on the MH-60 Black Hawk. If the velocity is low, the difference in pressures is very small and errors in the instrument could be greater than the measurement. So pitot tubes do not work very well for very low velocities. Such limitations results in the so-called jerky readings as demonstrated in Fig. 5.22).

Upon taking the Air Speed time series to the phase space, one can see almost the same data manifold. Again, all the erroneous poitns lie on the line $y = x$. We can send the outliers back to the main manifold (orange time series in Fig. 5.22).

VADR and HUMS collect readings from 1500 and 64 sensors, respectively. However, for the majority of time series we can observe four classes of data manifolds (Fig.

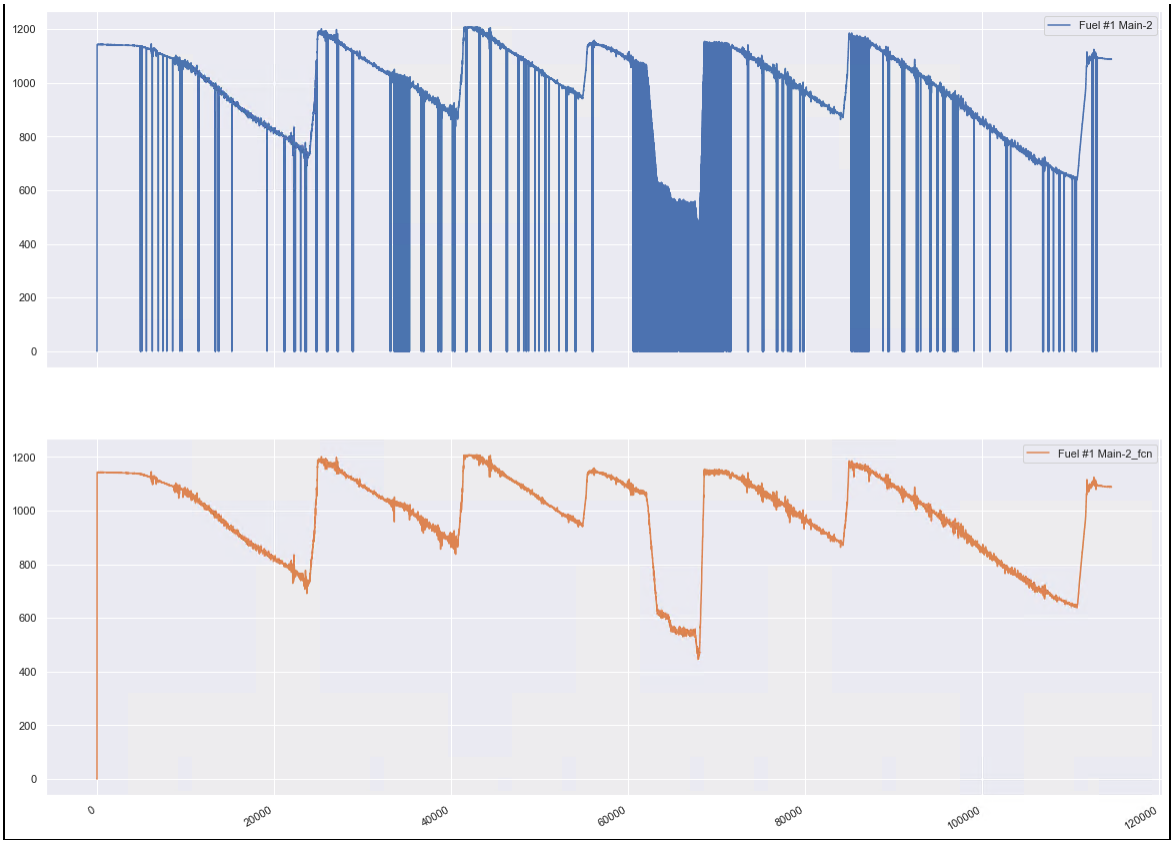


Figure 5.21. Gallons of fuel per hour readings after returning irregular points back to the regular data manifold.

5.23). Taking the time series to the phase space reveals the data manifolds make it possible to learn the data effectively.

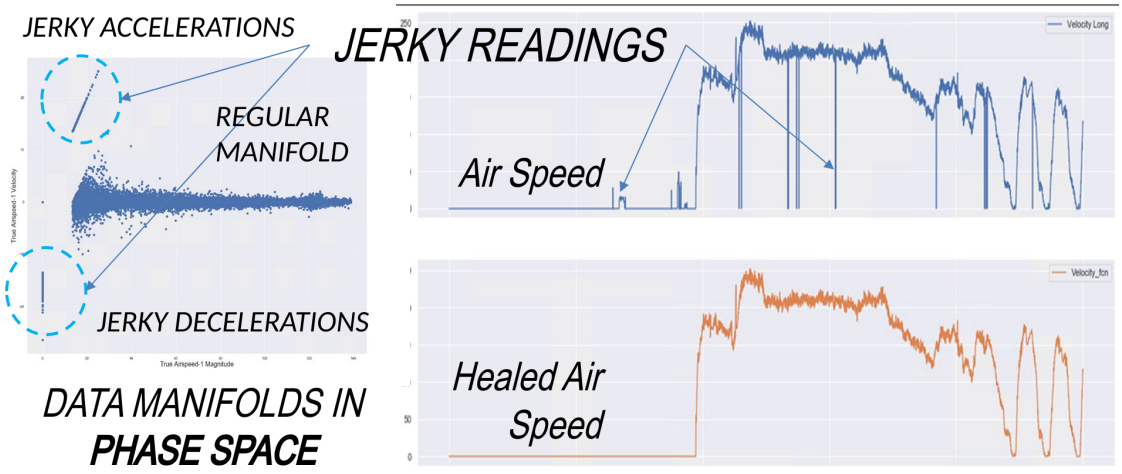


Figure 5.22. Air speed original readings (blue time series) along with the associated data manifold in the phase space, and the air speed readings after cross validation (orange time series).

5.6 Automated Assessment of Flight Operational Stages

The next step in our work was the Automated Assessment of Flight Operational Stages required for Selectable Filtering Data Manifolds.

Each Flight Operational Stage requires an individual approach to the data refinement and analysis, which should be performed even when formally the Flight Operational Regime is not defined.

We develop an automated data Flight Stage annotation procedure to extend the standard Regime assessment over all time stamps. We have stratified the data into the Regime layers to foster the data refinement and analysis. The assessment ground in our approach are the time series for 10 standard Regime Triggers (such as Air Speed, Rotor Speed, Pitch, Roll, Yaw, and the Weight on Wheels) used for the regime assessment in HUMS systems (Fig. 5.24).

Our algorithm for Automated Flight Operational Regimes Assessment constitutes a Support Vector Machine, an automated learning procedure over 10 time-series of regime triggers supervised by the available HUMS Regime readings. When the Flight Operational Regime assessed by the standard Army routines is known, we use these

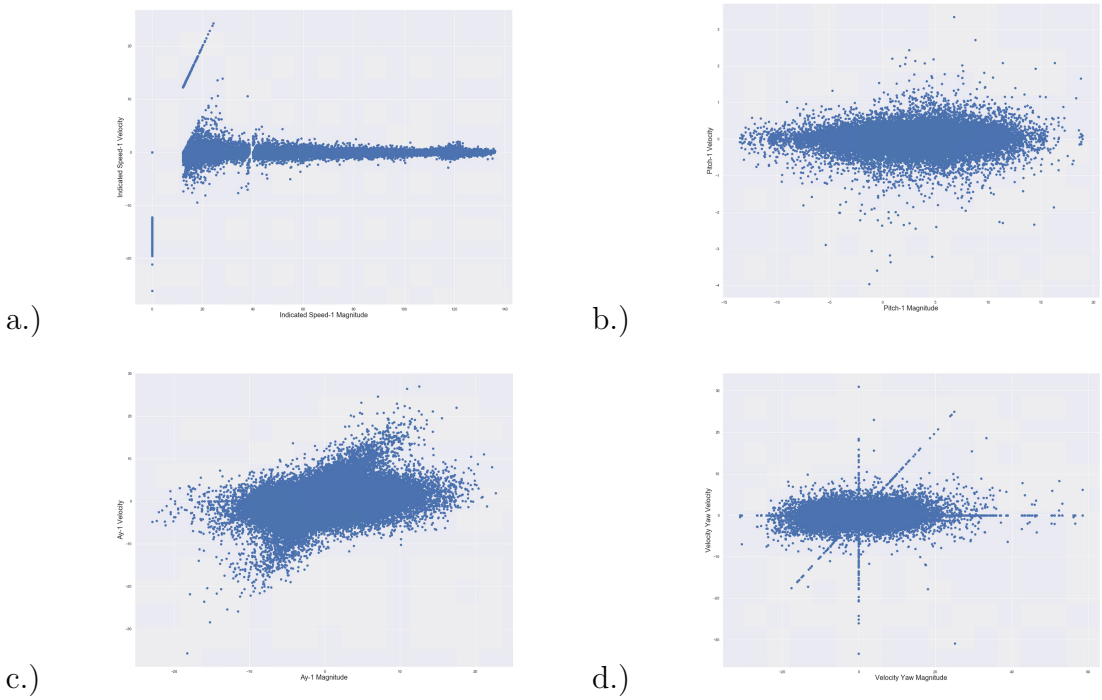


Figure 5.23. a.The data monifold for Indicated Speed- 1; b.) The data monifold for Pitch-1 c.) The data monifold for Acceleration in y-direction; d.) The data monifold for Velocity Yaw.

1.	AirSpd_ADC-1	Airspeed_true_ADC#1
2.	AirSpd_ADC-2	Airspeed_true_ADC#2
3.	Nr-DCU1	Rotor Speed (Nr)
4.	Nr-DCU2	Rotor Speed (Nr)
5.	PitAng-EGI1	Body Pitch Angle (EGI1)
6.	PitRat-EGI1	Y Body Pitch Rate (EGI1)
7.	RollAng-EGI1	Body Roll Angle (EGI1)
8.	RolRat-EGI1	X Body Roll Rate (EGI1)
9.	WOW1	Weight On Wheels
10.	YawRat-EGI1	Z Body Yaw Rate (EGI1)

Figure 5.24. Ten time series used as the regime triggers.

values of Regime triggers to learn the correspondent features and then make the Regime assessment for the intermediate timesteps. Accuracy of Machine Learning predictions is perfect over the annual data, with exact matching with Army -defined regimes. On the upper diagram, you see the original Regime readings reported in the

HUMS file of a flight, and on the lower diagram we show the results of the automated assessments of the Operational Stages made by our Support Vector Machine (Fig. 5.25).

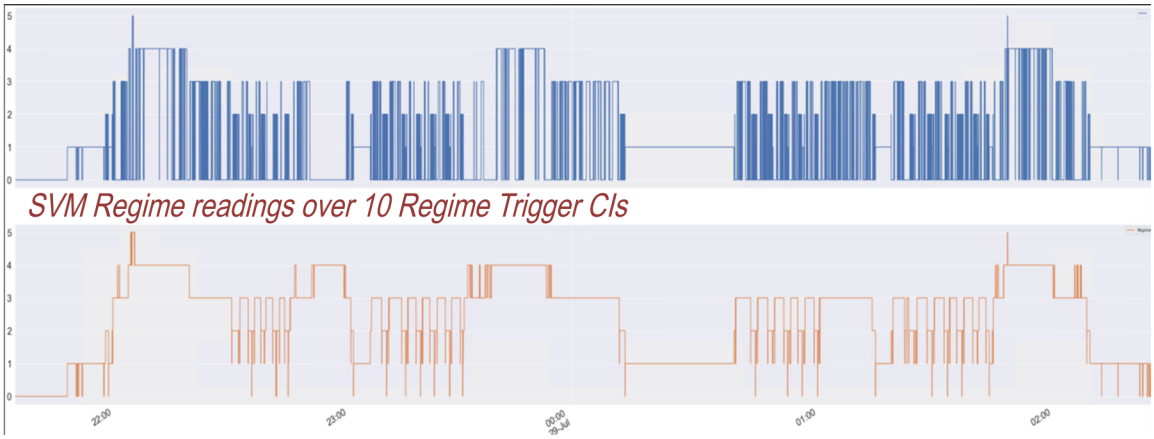


Figure 5.25. Automatically identified Regimes using the regime triggers. Blue time series is for original regime readings and the orange time series shows the regimes identified by the Support Vector Machine.

Aircraft maneuvers in different operational flight Regimes call for the further stratification of data into sub-regimes that can be identified by means of the unsupervised Machine Learning algorithms in view of further analysis. Simply speaking, a 30-degree turn at 80 Kts and the same turn at 145 Kts are the different things. On the diagram at the right-hand side, you see the heart-like form – the data manifold representing the annual kinematic readings on the Roll, Pitch, and Yaw rate of an aircraft in the Regime "3" (corresponding to the Army Regime of 80 Kts). The color-coded submanifolds of the data manifold at 80 Kts are characterized by the coherent variations of the Roll, Pitch, and Yaw rate, representing the rotation of the aircraft while performing certain maneuvers. The sub-regime outliers are identified as scattered points laying outside the regular manifold. It is worth to mention that this cardiac-shaped form has a cavity, and some outliers are located inside the cavity, as well. Each outlier can nevertheless be identified as belonging to some "home-manifold" (of the same

color) and healed by sending it back to their home manifolds. In our approach the different data streams are cleaning each other (Fig. 5.26).

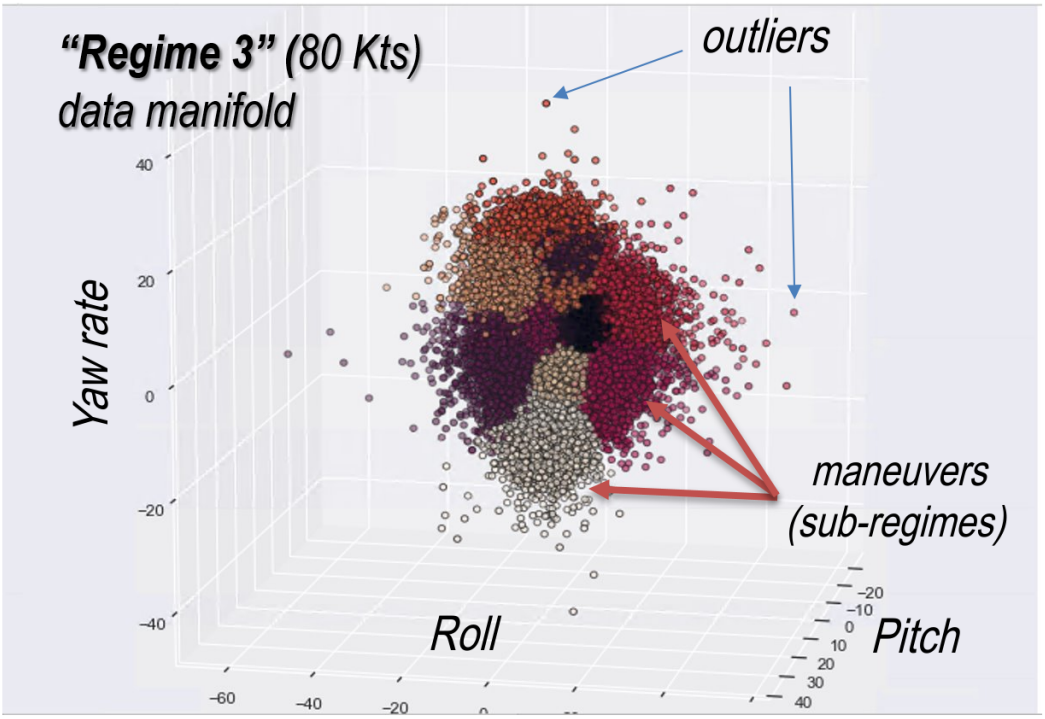


Figure 5.26. The color-coded sub-manifold of the data manifold of the annual aircraft kinematic readings in Regime 3.

The major deliverables of our project are the algorithms for

- Flight/ Non-flight data files separation allowing for up to 73% functional reduction in the numbers of files and up to 30% reduction in the data volume.
- Fusion of independent/ selectable Flight and Maintenance data systems (including VADR, HUMS, and ULLS files) with 97% of matching for ULLS files
- Automated Operational Flight Stages assessment matching precisely with army-defined operational flight regimes.

- restoration of missing and corrupted Flight Data via VADR-HUMS-ULLS data cross-validation. In our work, we had occasions to restore up to 37% of missing readings using the files cross-validation. Finally,
- Selectable Filtering Data Manifolds - Identification, automated health predictive analysis, and healing jerky readings. A flight data file might contain up to 8% of readings identifiable as outliers.

Our work opens the way for developing the Automated Health State Predictive Analytics algorithms for the future Self-Aware Aircrafts.

CHAPTER 6
CONCLUSION

With widespread applications of Artificial Intelligence, the idea of self-aware data is rapidly gaining interest of the scientific community. Over the past decades there have been the great amount of algorithms developed to significantly improve the capabilities of the perception, understanding, decision-making, and control for autonomous systems.

In this dissertation we proposed a strategy (Fig. 1.1)to evaluate the self-awareness for data sets of different nature: from geo-spatial data acquired from the Open Street Map and Financial time series of the S&P 500 all the way to the Flight Army data generated by MH-60 Black Hawk helicopters.

The strategy lies its foundation in the determining a dynamical model for the given data. There could be many different dynamical models that reveal different characteristics of the data set one deals with. The main goal of the dynamical model is not only to describe the behavior of the data, but equip the data with suitable geometry, distances.

Once the geometry is defined, we introduce merge the dynamical model to a phase space in which the model evolves. Depending on the field from which the data originated, there could be numerous ways to define the phase spaces that respect the patterns present in the data which further lets us to increase the dimensionality of data without observing the severe consequences of the curse of dimensionality. While studying S&P 500, the financial time series were submerged into the "Return vs Roughness" phase space; in case of the time series generated by the engineering systems of the HM-60 Black Hawk helicopters, "Position vs Velocity" phase space was more suitable to perform data analysis.

The central part of the strategy belongs to the notion of the data manifolds. Such approach lets us to separate the manifold into regions of different predictablities:

- the most visited states (regular points) have higher predictability,
- the parts of the manifold that are visited rarely (irregular points).

We were able to use uncertainty to develop a way to separate the erroneous data

points from extreme events. The erroneous points can be healed via cross-validation and brought back to the main (regular) regions of the data manifold. Extreme events were studied further in Chapter (reference here). We assess the degree of uncertainty by the Shannon's entropy calculated on the probability that the threshold changes at any given time.

At the final stage, we were able to split uncertainty into three components:

1) $\mathcal{D}(p)$ measures our capability to predict the future state from the past states.

2) $\mathcal{U}(p)$ measures our capability to predict the forthcoming state from the last occurred.

3) $\mathcal{E}(p)$ measures the portion of entropy we are unable to predict.

For each of the data sets we dealt with, the above steps were fully automated, excluding human being judgments and reducing an expert interference when it comes to data analysis. Data can "understand" itself to some degree.

BIBLIOGRAPHY

[1] Rosenblatt, F. The perceptron: a probabilistic model for information storage and organization in the brain. Psychological review, 65(6), 386 (1958).

[2] Mahesh, B. Machine Learning Algorithms-A Review. International Journal of Science and Research (IJSR).[Internet], 9, 381-386 (2020)

[3] J. O. Kephart and D. M. Chess, "The vision of autonomic computing," Computer, vol. 36, no. 1, pp. 41–50, Jan. 2003.

[4] S. Kounev, "Engineering of self-aware IT systems and services: State-of-the-art and research challenges," in Computer Performance Engineering. EPEW (Lecture Notes in Computer Science), vol. 6977, N. Thomas, Eds. Berlin, Germany: Springer, 2011.

[5] S. Kounev, X. Zhu, J. O. Kephart, and M. Kwiatkowska, "Model-driven algorithms and architectures for self-aware computing systems (Dagstuhl seminar 15041)," Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, Dagstuhl, Germany, Tech. Rep., 2015, vol. 5, no. 1, doi: 10.4230/DagRep.5.1.164.

[6] Li, Yubin, et al. "A self-aware data compression system on FPGA in Hadoop." 2015 International Conference on Field Programmable Technology (FPT). IEEE, 2015.

[7] Bauer, A., Züfle, M., Herbst, N., Zehe, A., Hotho, A., & Kounev, S. Time series forecasting for self-aware systems. Proceedings of the IEEE, 108(7), 1068-1093, (2020).

[8] Allaire, D., Biros, G., Chambers, J., Ghattas, O., Kordonowy, D., & Willcox, K. Dynamic data driven methods for self-aware aerospace vehicles. Procedia Computer Science, 9, 1206-1210, (2012).

[9] Du, Z., Guo, Q., Zhao, Y., Zhi, T., Chen, Y., & Xu, Z. Self-aware neural network systems: A survey and new perspective. Proceedings of the IEEE, 108(7), 1047-1067, (2020).

[10] Raymer, M. G., Ml Beck, and D. McAlister. "Complex wave-field reconstruction using phase-space tomography." Physical review letters 72.8 (1994).

[11] Srinivasan, N., M. T. Wong, and S. M. Krishnan. "A new phase space analysis algorithm for cardiac arrhythmia detection." Proceedings of the 25th Annual

International Conference of the IEEE Engineering in Medicine and Biology Society (IEEE Cat. No. 03CH37439). Vol. 1. IEEE, 2003.

[12] Habershon, Scott, et al. "Ring-polymer molecular dynamics: Quantum effects in chemical dynamics from classical trajectories in an extended phase space." Annual review of physical chemistry 64 (2013): 387-413.

[13] Schön, J. Christian, and Martin Jansen. "First step towards planning of syntheses in solid-state chemistry: determination of promising structure candidates by global optimization." Angewandte Chemie International Edition in English 35.12 (1996): 1286-1304.

[14] Jarrold, M. F., J. E. Bower, and K. Creegan. "Chemistry of semiconductor clusters: A study of the reactions of size selected Si+ n (n= 3–24) with C2H4 using selected ion drift tube techniques." The Journal of Chemical Physics 90.7 (1989): 3615-3628.

[15] Folland, Gerald B. Harmonic Analysis in Phase Space.(AM-122), Volume 122. Princeton university press, 2016.

[16] Dalcanton, Julianne J., and Craig J. Hogan. "Halo cores and phase-space densities: observational constraints on dark matter physics and structure formation." The Astrophysical Journal 561.1 (2001): 35.

[17] Kuo, Frances Y., and Ian H. Sloan. "Lifting the curse of dimensionality." Notices of the AMS 52.11 (2005): 1320-1328.

[18] Chakraborty, Rudrasis, and Baba C. Vemuri. "Statistics on the Stiefel manifold: Theory and applications." The Annals of Statistics 47.1 (2019): 415-438.

[19] Moore, Andrew W., and Mary S. Lee. "Efficient algorithms for minimizing cross validation error." Machine Learning Proceedings 1994. Morgan Kaufmann, 1994. 190-198.

[20] Behrens, Cibele N., Hedibert F. Lopes, and Dani Gamerman. "Bayesian analysis of extreme events with threshold estimation." Statistical Modelling 4.3 (2004): 227-244.

[21] Rahmstorf, Stefan, and Dim Coumou. "Increase of extreme events in a warming world." Proceedings of the National Academy of Sciences 108.44 (2011): 17905-17909.

[22] Huser, Raphaël, and Anthony C. Davison. "Space—time modelling of extreme events." Journal of the Royal Statistical Society: Series B: Statistical Methodology (2014): 439-461.

[23] Longin, François, ed. Extreme events in finance: A handbook of extreme value theory and its applications. John Wiley & Sons, 2016.

[24] Farazmand, Mohammad, and Themistoklis P. Sapsis. "Extreme events: Mechanisms and prediction." Applied Mechanics Reviews 71.5 (2019).

[25] Fama, Eugene F. "Random walks in stock market prices." Financial analysts journal 51.1 (1995): 75-80.

[26] Meerschaert, Mark M., and Enrico Scalas. "Coupled continuous time random walks in finance." Physica A: Statistical Mechanics and its Applications 370.1 (2006): 114-118.

[27] Hassan, Ahmed, and Dragomir Radev. "Identifying text polarity using random walks." Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics. 2010

[28] Li, Changyang, et al. "Robust saliency detection via regularized random walks ranking." Proceedings of the IEEE conference on computer vision and pattern recognition. 2015.

[29] Xia, Feng, et al. "Random walks: A review of algorithms and applications." IEEE Transactions on Emerging Topics in Computational Intelligence 4.2 (2019): 95-107.

[30] Blanchard, Ph., Volchenkov, D., *Mathematical Analysis of Urban Spatial Networks*, Springer Series *Understanding Complex Systems*, Berlin / Heidelberg (2009)

[31] Volchenkov, D. "*Grammar Of Complexity: From Mathematics to a Sustainable World*", in World Scientific Series "*Nonlinear Physical Science*" (2018)

[32] Volchenkov, D., Blanchard, Ph., "Markov Chain Methods For Analyzing Urban Networks" *Journal of Statistical Physics*, 1572-9613 (Online), Vol. **13**2, Pages 1051-1069 (2008)

[33] Hillier, B., Hanson, J., *The Social Logic of Space* (1993, reprint, paperback edition ed.). Cambridge: Cambridge University Press (1984)

[34] Ortega-Andeane, P., Jiménez-Rosas, E., Mercado-Doménech, S., & Estrada-Rodrýguez, C., "Space syntax as a determinant of spatial orientation perception." *Int. J. of Psychology*, **40** (1), 11-18 (2005)

[35] Chown, M. "Equation can spot a failing neighborhood", *New Scientist* **2628**, London (2007)

[36] Smart Growth America Report (2014) "*Measuring Sprawl 2014*"; available at `https://www.smartgrowthamerica.org/app/legacy/documents/measuring-sprawl-2014.pdf`

[37] Hillier, B., *Space is the Machine: A Configurational Theory of Architecture*. Cambridge University Press. ISBN 0-521-64528-X (1999)

[38] Hansen, W.G., "How accessibility shapes land use." *J. of the Am. Inst. Planners* **25**, 73 (1959)

[39] Wilson, A.G., *Entropy in Urban and Regional Modeling*, Pion Press, London (1970)

[40] Batty, M., *A New Theory of Space Syntax*, UCL Centre For Advanced Spatial Analysis Publications, CASA Working Paper **75** (2004)

[41] Prisner, E., *Graph Dynamics*, Boca Raton (FL): CRC Press (1995)

[42] Horn, R.A., Johnson, C.R., *Matrix Analysis*, Cambridge University Press (1990)

[43] Burda, Z., Duda, J., Luck, J.M. & Waclaw, B., "Localization of the Maximal Entropy Random Walk" *Physical Review Letters* **102**(16), 160602., 10.1103/PhysRevLett.102.160602 (2009)

[44] Burda, Z., Duda, J., Luck, J.M. & Waclaw, B., "The various facets of random walk entropy" *Acta Physica Polonica B* **41** (5), pp. 949-987 (2010)

[45] Lovász, L., "Random Walks On Graphs: A Survey." *Bolyai Society Mathematical Studies* **2**: *Combinatorics, Paul Erdös is Eighty*, 1 Keszthely (Hungary) (1993)

[46] Shlesinger, M.F., "First encounters" *Nature* **450**(1) 40 (2007)

[47] Graves, L., (Ed.) "*A History of Lubbock*", Lubbock: West Texas Museum Association (1962).

[48] Ewing R., Meakins G., Hamidi S., Nelson A.C. (2014). "Relationship between urban sprawl and physical activity, obesity, and morbidity - Update and refinement". *Health Place.* **26**: 118 - 126.

[49] Berrigan D., Tatalovich Z., Pickle L.W., Ewing R., Ballard-Barbash R. (2014). "Urban sprawl, obesity, and cancer mortality in the United States: cross-sectional analysis and methodological challenges". *International Journal of Health Geographics* **13**: 3.

[50] James P., Troped P.J., Hart J.E., Joshu C.E., Colditz G.A., Brownson R.C., Ewing R., Laden F. (2013). "Urban sprawl, physical activity, and body mass index: Nurses' Health Study and Nurses' Health Study II". *Am J Public Health* **103**(2): 369 - 75.

[51] Lee I.M., Ewing R., Sesso H.D. (2009). "The built environment and physical activity levels: the Harvard Alumni Health Study". *Am J Prev Med.* **37**(4): 293 - 298.

[52] Ewing R., Brownson R.C., Berrigan D. (2006). Relationship between urban sprawl and weight of United States youth. *Am J Prev Med.* **31**(6): 464 - 74.

[53] Ewing R., Schmid T., Killingsworth R., Zlot A., Raudenbush S. (2003). "Relationship between urban sprawl and physical activity, obesity, and morbidity". *Am J Health Promot.* **18**(1): 47 - 57.

[54] Volchenkov, D., Ph. Blanchard, "Intelligibility and first passage times in complex urban networks", *Proc. R. Soc. A* **464**, 2153–2167 (2008).

[55] Volchenkov, D., Blanchard, Ph., "*Introduction to Random Walks and Diffusions on Graphs and Databases*", Springer Series in *Synergetics* , Vol. **10**, Berlin / Heidelberg (2011).

[56] `https://www.trulia.com/real_estate/Lubbock-Texas/crime/`

[57] Graham, B. (1949, 1954, 1959, 1965) *The Intelligent Investor* by by Harper & Row Publishers Inc, New York.

[58] Fama, E. (1965). "The Behavior of Stock Market Prices". Journal of Business. 38: 34-105.

[59] Nolte, D.D. (2010). "The tangled tale of phase space". *Physics Today* **63**(4): 33-38.

[60] Radzewicz A., (2013), "Real estate market system theory approach phase space", *Real Estate Management and Valuation*, vol. **21**, no. 4, pp. 87-95.

[61] Can-Zhong Yao, Qing-Wen Lin (2017). "Recurrence plots analysis of the CNY exchange markets based on phase space reconstruction". *The North American Journal of Economics and Finance*, Vol. **42**, Pages 584-596.

[62] Ulam, S. M. (1964) Problems in Modern Mathematics, New York: Interscience.

[63] Galbraith, J. (1988 edition) *The Great Crash 1929*, Houghton Mifflin Co. Boston.

[64] Han, Te Sun, Kobayashi, Kingo (2002). *Mathematics of Information and Coding*. American Mathematical Society.

[65] Cover, T. M., Thomas, J. A. (1991). *Elements of Information Theory*. Wiley, 576 pp.

[66] Lee, P. M. (2012). *Bayesian Statistics*. Wiley. ISBN 978-1-1183-3257-3.

[67] DelSole, T. (2004). "Predictability and Information Theory. Part I: Measures of Predictability". *J. Atmos. Sci.*, **61**, 2425–2440.

[68] Kleeman, R. (2002). "Measuring dynamical prediction utility using relative entropy". *J. Atmos. Sci.*, **59**, 2057 – 2072.

[69] Volchenkov, D., (2018). "*Grammar Of Complexity: From Mathematics to a Sustainable World*", in World Scientific Series "*Nonlinear Physical Science*".

[70] James, R.G., Ellison, Ch.J., and Crutchfield, J.P. (2011) Anatomy of a bit: Information in a time series observation *Chaos* **21**, 037109.

[71] Mahdavi Damghani, B. (2013). "The Non-Misleading Value of Inferred Correlation: An Introduction to the Cointelation Model". *Wilmott*. 2013 (**1**): 50-61.

[72] Kaneko,K., Tsuda, I. (2011) *Complex Systems: Chaos and Beyond: A Constructive Approach with Applications*, Springer Series in *Science& Business Media* .

[73] Lin, H.W., Tegmark, M. (2017) "Criticality in Formal Languages and Statistical Physics" *Entropy*, **19**, 299.

[74] Smith, V.L., Suchanek, G.L., Williams, A.W. (1988) "Bubbles, Crashes, and Endogenous Expectations in Experimental Spot Asset Markets". *Econometrica*. The Econometric Society. **56** (5): 1119-1151.

[75] Sethna, J.P. (2006). *Statistical mechanics : entropy, order parameters, and complexity* Oxford: Oxford University Press.

[76] Kaplan, S. Tolley's Handbook of Disaster and Emergency Management: Pronciples and Best Practices; Disaster and Emergency Management Systems, Lexis Nexis UK, (2004).

[77] Richter Scale / Mercalli Scale https://www.usgs.gov/media/images/modified-mercalli-intensity-mmi-scale-assigns-intensities

[78] Beaufort Wind Scale https://www.spc.noaa.gov/faq/tornado/beaufort.html

[79] Saffir-Simpson Hurricane Wind Scale https://www.nhc.noaa.gov/aboutsshws.php

[80] Fujita Tornado Damage Scale https://www.spc.noaa.gov/faq/tornado/f-scale.html

[81] Homeland Security Advisory System https://en.wikipedia.org/wiki/Homeland_Security_Advisory_System

[82] U.S. Climate Extremes Index (CEI) https://www.ncdc.noaa.gov/extremes/cei/introduction

[83] Rohn, E., Blackmore, D. "A Unified Localizable Emergency Events Scale". *International Journal of Information Systems for Crisis Response and Management (IJISCRAM)*, **1(4)**, 1-14, (2009).

[84] Pisarchik, Alexander N. and Jaimes-Reátegui, Rider and Sevilla-Escoboza, Ricardo and Huerta-Cuellar, G. and Taki, Majid. "Rogue Waves in a Multistable System". *Phys. Rev. Lett.*, **107**, 274101, (2011).

[85] Pisarchik, A.N., Grubov, V.V., Maksimenko, V.A. et al. "Extreme events in epileptic EEG of rodents after ischemic stroke". *Eur. Phys. J. Spec. Top.* **227**, 921–932, (2018).

[86] Plotnick, L., Gomez, E., White, C., Turoff, M. "Furthering Development of a Unified Emergency Scale Using Thurstone's Law of Comparative Judgment: A Progress Report ABSTRACT", https://www.dhs.gov/xlibrary/assets/hsas_unified_scale_feedback.pdf, (2007).

[87] Richter magnitude scale https://en.wikipedia.org/wiki/Richter_magnitude_scale

[88] 2010 Haiti Earthquake https://en.wikipedia.org/wiki/2010_Haiti_earthquake

[89] 2019 Ridgecrest earthquakes https://en.wikipedia.org/wiki/2019_Ridgecrest_earthquakes

[90] Trading Halt Definition James Chen - https://www.investopedia.com/terms/t/tradinghalt.asp

[91] J. Lee, Y. Fany and S. A. Sisson, "Bayesian threshold selection for extremal models using measures of surprise", *arXiv:1311.2994v2 [stat.ME]*, (2014).

[92] Carroll, L. "Through The Looking-Glass and What Alice Found There", Chicago, *W.B. Conkey Co.*, (1900).

[93] Hudson, R., Gregoriou, A., "Calculating and Comparing Security Returns is Harder than you Think: A Comparison between Logarithmic and Simple Returns", *International Review of Financial Analysis*, 38, 151-162, (2015).

[94] Onnela, J.-P., Chakraborti, A., Kaski, K., Kertész, J., Dynamic asset trees and Black Monday. *Physica A: Statistical Mechanics and Its Applications*, **(1-2)**, 247–252, (2003).

[95] Birru, J., Figlewski, S., "Anatomy of a meltdown: The risk neutral density for the S&P 500 in the fall of 2008", *Journal of Financial Markets*, **15(2)**, 151–180, (2012).

[96] Beran, J., *Statistics for Long-Memory Processes*, Chapman & Hall/CRC, Roca Raton, FL, (1994).

[97] Ihlen E.A., "Introduction to multifractal detrended fluctuation analysis in matlab", *Front Physiol.* **3**:141 (2012).

[98] Thompson, J.R., Wilson, J.R., "Multifractal detrended fluctuation analysis: Practical applications to financial time series", *Mathematics and Computers in Simulation*, Vol. **126**, Pages 63-88, (2016).

[99] Harte, D., *Multifractals*. London: Chapman & Hall.(2001).

[100] Calvet, L., Fisher, A., "Multifractality in asset returns: theory and evidence", *Rev. Econ. Stat.*, **84**, pp. 381-406 (2002).

[101] J.W. Kantelhardt, S.A. Zschiegner, E. Koschielny-Bunde, S. Havlin, A. Bunde, H.E. Stanley, "Multifractal detrended fluctuation analysis of nonstationary time series", *Physica A* **316** (2002) 87–114.

[102] Smirnov, V., Volchenkov, D., "Five Years of Phase Space Dynamics of the Standard & Poor's 500", *Applied Mathematics and Nonlinear Sciences* **4**(1) 203–216 (2019)

[103] Zhi-Qiang Jiang, Wen-Jie Xie, Wei-Xing Zhou, and Didier Sornette, "Multifractal analysis of financial markets: a review", *Reports on Progress in Physics* 10 September 2019 *https://doi.org/10.1088/1361-6633/ab42fb*

[104] Coles, Stuart *An Introduction to Statistical Modeling of Extreme Values*. Springer, London, 2001.

[105] Fisher RA, Tippett LHC. *Limiting forms of the frequency distribution of the largest and smallest member of a sample*. Proceedings of the Cambridge Philosophical Society, 24:180-190, 1928.

[106] Gnedenko BV. *Sur la distribution limite du terme maximum d'une série aléatoire*. Annals of Mathematics 44:423-453, 1943.

[107] Xinyan Zhang and Jinguo Lian. *Analysis of Extreme Value at Risk to Amazon Stocks*. International Journal of Engineering Research and Development, 14(2):62-71, 2018.

[108] Nadarajah, S., Kotz, S., "The beta Gumbel distribution", *Mathematical Problems in Engineering*, **4**, 323–332, (2007).

[109] Abbas, K., Yincai, T., "Comparison of estimation methods for Frechet distribution with known shape", *Caspian Journal of Applied Sciences Research*, **1**(10), 58-64, (2012).

[110] Rinne, H., The Weibull distribution: a handbook, Chapman and Hall/CRC, (2008).

[111] Eliane C. Pinheiro and Silvia L. P. Ferrari. *A comparative review of generalizations of the Gumbel extreme value distribution with an application to wind speed data*. Journal of Statistical Computation and Simulation, 86(11), 2015.

[112] Castillo E., Hadi A.S., Balakrishnan N., and Sarabia J. M.. *Extreme Value and Related Models with Applications in Engineering and Science*. New Jersey: John Wiley & Sons, 2005.

[113] Ferrari S.L.P. and Pinheiro E.C. *Small-sample one-sided testing in extreme value regression models*. AStA Advances in Statistical Analysis, 100(1): 79-97, 2016.

[114] Pickands, J. Statisical inference using extreme order statistics. *Ann. Statist.*, **3**, 119-131, (1975)

[115] Scarrott, C., MacDonald, A. A review of extreme value threshold estimation and uncertainty quantification. *RevStat - Statistical Journal*, **10(1)**, 33-60, (2012).

[116] Gencay, R., Selcuk, F., Ulugulyagci, A. EVIM: A Software Package for Extreme Value Analysis in MATLAB. *Studies in Nonlinear Dynamics and Econometrics*, **5**(3), 1-29, (2001).

[117] Davison, A.C., Smith, R.L. Models for exceedance over high thresholds (with discussion). *J.R. Statist. Soc. B*, **52**, 237-254, (1990).

[118] Ghosh, S., Resnick, S.I. A discussion on mean excess plots. *Stochastic Processes and their Applications*, **120**, (2010).

[119] Kratz, M., Resnick, S. I., "The QQ-estimator and heavy tails", *Stochastic Models*, **12(4)**, 699-724, (1996).

[120] Drees, H., De Haan, L., Resnick, S., "How to make a Hill plot", *The Annals of Statistics*, **28(1)**, 254-274, (2000).

[121] Todorovic, P., Rousselle, J. Some problems of flood analysis. *Water Resources Research*, **7**(5), 1144-1150, (1971).

[122] Todorovic, P., Zelenhasic, E. (1970). A stochastic model for flood analysis. *Water Resources Research*, **6**(6), 1641-1648.

[123] Hogg, R., Klugman, S. *Loss Distribution*, Wiley, New York, (1984).

[124] Embrechts, P., McNeil, A., Frey, R. *Quantitative Risk Management: Conepts, Techniques, and Tools*, Princeton University Press, New Jersey, (2005).

[125] Guess, F., Proschan, F. *Mean Residual Life: Theory and Applications*, Defence Technical Information Center, (1985).

[126] Drees, H., "Refined estimators of the extreme value index", *The Annals of Statistics*, 2059-2080, (1995).

[127] Dekkers, A. L., Einmahl, J. H., De Haan, L., "A moment estimator for the index of an extreme-value distribution", *The Annals of Statistics*, **17(4)**, 1833-1855, (1989).

[128] Northrop, P.J., Coleman, C.L. Improved threshold diagnostic plots for extreme value analyses. *Extremes*, **17**, 289-303, (2014).

[129] Lomba, J.S., Alves, M.I. L-moments for automatic threshold selection in extreme value analysis. *arXiv:1905.08726v1 [stat.ME]*, (2019).

[130] Manurung, A., Wigena, A.H., Djuraidah, A. GPD threshold estimation using measure of surprise. *International Journal of Sciences: Basic and Applied Research*, **42**(3), 16-25, (2018).

[131] Wadsworth, J. Exploiting Structure of Maximum Likelihood Estimators for Extreme Value Threshold Selection. *Technometrics*, **58**(1), 116-126, (2016).

[132] Thompson, P., Cai, Y., Reeve, D., Stander, J. Automated threshold selection methods for extreme wave analysis. *Coastal Engineering*, **56**, 1013-1021, (2009).

[133] Langousis, A., Mamalakis, A., Puliga, M., Deidda, R. Threshold detection for the generalized Pareto distribution: Review of representative methods and application to the NOAA NCDC daily rainfall database. *Water Resources Research*, **52**, 2659-2681, (2016).

[134] G'Sell, M., Wager, S., Chouldechova, A., Tibshirani, R. Sequential selection procedures and false discovery rate control. *Journal of the Royal Statistical Society: Series B*, **78**(2), 423-444, (2016).

[135] Bader, B., Yan, J., Zhang, X. Automated threshold selection for extreme value analysis via ordered goodness-of-fit tests with adjustment for false discovery rate. *The Annals of Applied Statistics*, **12**(1), 310-329, (2018).

[136] Greenwood, J.A. Probability weighted moments: Definition and relation to parameters of several distributions expressible in inverse form. *Water Resources Research*, **15**(5), 1049-1054, (1979).

[137] DuMouchel, W.H. Estimating the stable index α in the order to measure tail thickness: A critique. *Ann. Statist.*, **11**, 1019-1031, (1983).

[138] Ferreira, A., De Haan, L., Peng, L. On optimising the estimation of high quantiles of a probability distribution. *Statistics*, **37**, 401-434, (2003).

[139] Reiss, R.D., Thomas M. *Statistical Analysis of Extreme Values: With Applications to Insurance, Finance, Hydrology and Other Fields*, Birkhauser, Boston, (2017).

[140] Neves, C., Alves, M.I.F. Reiss and Thomas' automatic selection of the number of extremes. *Comp. Statist. Data. Anal.*, **47**, 689-704, (2004).

[141] Schneider, L.F., Krajina, A., Krivobokova, T. Threshold Selection in Univariate Extreme Value Analysis. *arXiv:1903.02517v1 [stat.ME]*, (2019).

[142] E. Floriani, Volchenkov, D., R. Lima, "A System close to a threshold of instability", *J. of Physics A : Math. General* **36**, 4771-4783 (2003).

[143] Volchenkov, D., "*Survival under Uncertainty An Introduction to Probability Models of Social Structure and Evolution*", Springer Series: *Understanding Complex Systems* (2016).

[144] Rakthanmanon, T., Campana, B., Mueen, A., Batista, G., Westover, B., Zhu, Q., Keogh, E. (2012, August). Searching and mining trillions of time series subsequences under dynamic time warping. *In Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining* (pp. 262-270).

[145] Chen, M. J., Bovik, A. C. (2011). Fast structural similarity index algorithm. *Journal of Real-Time Image Processing*, **6**(4), 281-287.

[146] Needleman, S. B., Wunsch, C. D. (1970). A general method applicable to the search for similarities in the amino acid sequence of two proteins. *Journal of Molecular Biology*, **48**(3), 443–453. doi:10.1016/0022-2836(70)90057-4

[147] Wang, X. and Wang, C. Time series data cleaning: A survey. *Ieee Access*, **8**, pp.1866-1881 (2019).

APPENDIX: COMPUTER CODE

Multi-scale Analysis of Urban Spatial Structures acquired from *OpenStreetMap*

```
# -*- coding: utf-8 -*-
"""
Spyder Editor

This is a temporary script file.
"""

from lxml import etree
import pandas as pd
import numpy as np
import ast
import scipy.sparse as sp
from scipy import linalg
from scipy.sparse.linalg import eigs, eigsh
import plotly.express as px
from numba import jit
import numba
import plotly.express as px
from scipy.sparse import csr_matrix



def reduce_components(adj_mtx, nodes):
    """
    """
    n_components, labels = sp.csgraph.connected_components(csgraph=adj_mtx,
                           directed=False, return_labels=True)
    if n_components > 1:
        print('There are {} components'.format(n_components))
        u, c = np.unique(labels, return_counts = True)
        adj_mtx = np.delete(adj_mtx, np.where(labels == u[np.argmin(c)]),
         axis = 0)
        adj_mtx = np.delete(adj_mtx, np.where(labels == u[np.argmin(c)]),
         axis = 1)
        nodes = np.delete(nodes, np.where(labels == u[np.argmin(c)]))
```

```
        return reduce_components(adj_mtx, nodes)
    else:
        return adj_mtx, list(nodes)


def reduce_components2(adj_mtx, nodes):
    """
    """

    n_components, labels = sp.csgraph.connected_components(csgraph=adj_mtx,
                           directed=False, return_labels=True)
    if n_components > 1:
        print('There are {} components'.format(n_components))
        pos = np.bincount(labels).argmax()
        ind_to_remove = np.where(labels != pos)[0]
        adj_mtx = np.delete(adj_mtx, ind_to_remove, axis = 0)
        adj_mtx = np.delete(adj_mtx, ind_to_remove, axis = 1)
        nodes = np.delete(nodes, ind_to_remove)
        return reduce_components(adj_mtx, nodes)
    else:
        return adj_mtx, list(nodes)



def adj_mtrx(graphml_path):
    """Return a connected graph represented by an adjacency matrix
    """
    tree = etree.parse(graphml_path)
    root = tree.getroot()

    #Extract nodes and edges
    nodes_stc = root.findall(".//{http://graphml.graphdrawing.org/xmlns}node")
    print('There are {} nodes'.format(len(nodes_stc)))
    edges_stc = root.findall(".//{http://graphml.graphdrawing.org/xmlns}edge")
    print('There are {} edges'.format(len(edges_stc)))

    #Create a list of nodes
    nodes = [n.get('id') for n in nodes_stc]

    #Create an initial adjacency matrix
```

```
adj_mtx = np.zeros((len(nodes), len(nodes)), dtype = np.int8)

#Fill the matrix
for edge in edges_stc:
    source, target = edge.get('source'), edge.get('target')
    i, j = nodes.index(source), nodes.index(target)
    adj_mtx[i, j] = adj_mtx[j, i] = 1

if np.allclose(adj_mtx, adj_mtx.T, rtol = 0.01, atol = 0.001):
    print('The adjacency matrix is symmetric')


#Reduce the connected components to 1
adj_mtx, nodes = reduce_components2(adj_mtx, nodes)

#Find the coordinates of all the nodes in adj matrix as pandas df
all_coord = [["", 0.0, 0.0] for i in range(len(nodes))]
for i in range(len(nodes)):
    all_coord[i][0] = nodes[i]
for node in nodes_stc:
    try:
        nd_num = node.get('id')
        i = nodes.index(nd_num)
        all_coord[i][1] =
         node.find("{http://graphml.graphdrawing.org/
         xmlns}data[@key='d5']").text
        all_coord[i][2] =
         node.find("{http://graphml.graphdrawing.org/
         xmlns}data[@key='d4']").text
    except ValueError:
        pass


df = pd.DataFrame(all_coord, columns = ['Node', 'Long', 'Lat'])
df['Long'] = df['Long'].astype(float, errors = 'raise')
df['Lat'] = df['Lat'].astype(float, errors = 'raise')

return adj_mtx, df
```

```
@jit(nopython = True)
def t1_mtx(adj_mtx):
    deg = adj_mtx.sum(axis = 0, dtype = numba.float64)
    t_mtx = np.zeros((adj_mtx.shape), dtype = numba.float64)
    n, m = np.nonzero(adj_mtx)
    for p in range(len(n)):
        i = n[p]
        j = m[p]
        t_mtx[i, j] = adj_mtx[i, j] / deg[i]


    return t_mtx


def my_eig(t_mtx, k, which):
    vals, vecs = eigs(t_mtx, k=k, which = which)
    vals, vecs = np.real(vals), np.real(vecs)
    vecs[:, 0] = abs(vecs[:, 0])

    return vals, vecs

@jit(nopython = True)
def iso_index(vals, vecs):
    row, col = vecs.shape
    perron = vecs[:, 0]
    r = 1/(perron**2)
    f = np.zeros(row, dtype = numba.float64)
    for i in range(row):
        f[i] = r[i] * ((vecs[i, 1:] ** 2 ) / (1 - vals[1: ])).sum()


    return 10*np.log10( f / np.min(f))


@jit(nopython = True)
def int_index(vals, vecs):
    row, col = vecs.shape
    perron = vecs[:, 0]
    r = 1/(perron**2)
    f = np.zeros(row, dtype = numba.float64)
```

```
    for i in range(row):
        f[i] = r[i] * ((vecs[i, 1:] ** 2 ) / (1 - vals[1: ])).sum()


    return 10*np.log10( np.min(f) / f)

#@jit(nopython = True)
def tinf_mtx(adj_mtx):
    adj_mtx = adj_mtx.astype(float)
    val, perron = eigs(adj_mtx, k=1, which = 'LR')
    val = np.real(val)
    perron = np.abs(np.real(perron)).flatten()
    ti_mtx = np.zeros((adj_mtx.shape), dtype = float)
    n, m = np.nonzero(adj_mtx)
    for p in range(len(n)):
        i = n[p]
        j = m[p]
        ti_mtx[i, j] = (adj_mtx[i, j] * perron[j]) / (val * perron[i])

    return ti_mtx




def ent_pressure(adj_mat, df):
    df['Long'] = pd.to_numeric(df['Long'])
    df['Lat'] = pd.to_numeric(df['Lat'])
    #v, w = linalg.eigh(adj_mat)
    adj_mat = adj_mat.astype('float64')
    v, w = eigsh(adj_mat, 3, which='LM')
    df['Perron'] = abs(w[:, np.argmax(v)])
    df['Ent_press'] = -np.log2(1-(df['Perron'] / df['Perron'].sum()))
    df['Rel_fugacity'] = (1 / df['Perron']) / ((1 / df['Perron']).sum())
    #for mapb in ["open-street-map", "stamen-terrain", "carto-darkmatter", \
    #             "carto-positron", "stamen-watercolor", "stamen-toner"]:
    for mapb in ["open-street-map"]:
         fig = px.scatter_mapbox(df, lat="Lat", lon="Long", \
          color = "Ent_press",
                hover_name="Node", hover_data=["Long", "Lat"], \
                zoom=12, height=900)
```

```
        fig.update_layout(mapbox_style=mapb)
        fig.update_layout(margin={"r":0,"t":0,"l":0,"b":0})
        #fig.show()
        fig.write_html("Lubbock_Entropic_pressure2_{}.html".format(mapb))

    return v, w




def first_pass_commute_time(t_mtrx):
    """
    """
    a = []
    m, n = t_mtrx.shape
    b = np.zeros((m, n), dtype = np.float64)

    w, v = np.linalg.eig(t_mtrx)
    v[:, 0] = np.abs(v[:, 0])
    rec_time = 1 / (v[:, 0] ** 2)
    for i in range(m):
        s = np.sum((v[i, 1: ] ** 2) / (1 - w[1 : ]))
        a.append(rec_time[i] * s)

    for i in range(m):
        for j in range(n):
            tmp = 0
            for s in range(1, m):
                tmp += (1/(1-w[s]))*((v[i,s]/v[i,0])-(v[j,s]/v[j,0]))**2
            b[i, j] = tmp

    return a, b
```

Five Years of Phase Space Dynamics of the Standard & Poor's 500

Return_roughness.py

```
# -*- coding: utf-8 -*-
"""
Created on Fri Aug 10 14:53:47 2018

@author: vesmirno
"""

import numpy as np
import pandas as pd

data = np.nan_to_num(pd.read_csv('10_yr_by_comp_NEW.csv').values, copy=True)
dates = data[ : ,0]
names = np.delete(list(pd.read_csv('10_yr_by_comp_NEW.csv')), 0)

#Drop the dates
data = np.delete(data, 0, 1)
print(data)
print(data.shape)

(i, j) = data.shape

#Dates will be dropped

returns = np.zeros((i, j), dtype = np.float32)

for row in range(1, i):
    for column in range(j):
      returns[row, column] = (data[row, column] - data[row - 1, column])
       / data[row, column]


pd.DataFrame(returns, index = dates, columns = names).to_csv(
'Log_return_10_yr_EXCEL.csv', sep = ',')
np.savetxt('Log_return_10_yr_NUMPY.csv', returns,
delimiter = ',', fmt = '%10.5f')

roughness = np.zeros((i, j), dtype = np.float32)
```

```
for row in range(2, i):
    for column in range(j):
      roughness[row, column] = returns[row, column] - returns[row - 1, column]


pd.DataFrame(roughness, index = dates, columns = names).to_csv(
'Roughness_10_yr_EXCEL.csv', sep = ',')
np.savetxt('Roughness_10_yr_NUMPY.csv', roughness,
delimiter = ',', fmt = '%10.5f')
```

Mutual_information.py

```
# -*- coding: utf-8 -*-
"""
Created on Fri Aug 24 19:26:38 2018

@author: Benjamin
"""

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from scipy.interpolate import splprep, splev
import bisect
import scipy.sparse.linalg as linalg

returns = np.delete(np.nan_to_num(pd.read_csv('Log_return_10_yr_EXCEL.csv').
values, copy=True), 0, 1)
roughness = np.delete(np.nan_to_num(pd.read_csv('Roughness_10_yr_EXCEL.csv').
values, copy=True), 0, 1)

original_data = pd.read_csv('10_yr_by_comp_NEW.csv')

names = np.delete(list(pd.read_csv('Log_return_10_yr_EXCEL.csv').
columns.values), 0)

#Compute transition matrix on the given data

def transitionNoWeights(returnMatrix, roughnessMatrix, numBins):
    """ This function computes a transition matrix disregarding
    the weights of companies. The size of the transition matrix
    is NumBins ** 2 (square matrix)
    """

    days, corps = returnMatrix.shape
    bins_count = numBins
    ret_hist, ret_bin_edges = np.histogram(returnMatrix,
     bins = bins_count,   density = False)
    rough_hist, rough_bin_edges = np.histogram(roughnessMatrix,
     bins = bins_count, density=False)
```

```
    trans = np.zeros((bins_count ** 2, bins_count ** 2),
     dtype = np.float64)

    np.savetxt('Return_coordinates.csv', ret_bin_edges,
     delimiter = ',')
    np.savetxt('Roughness_coordinates.csv', rough_bin_edges,
     delimiter = ',')


    for firm in range(0, corps):
        firm_ret = returnMatrix[2: , firm]
        firm_rough = roughnessMatrix[2: , firm]
        #Create a transition matrix
        length = len(firm_ret)

    #Lets fill the matrix with integers

        for i in range(0, length - 1):
            n = bisect.bisect_left(ret_bin_edges, firm_ret[i]) - 1
            m = bisect.bisect_left(rough_bin_edges, firm_rough[i]) - 1
            k = bisect.bisect_left(ret_bin_edges, firm_ret[i + 1]) - 1
            l = bisect.bisect_left(rough_bin_edges, firm_rough[i + 1]) - 1
            trans[m * bins_count + n, l * bins_count + k] += 1

    return trans



#Now we have a transition matrix that contains tons of zero entries
#I am writing a function that creates walks after a given number of days


def daysTransit(transitMatrix, days):
    """ This function returns THEORETICAL transtition matrix with
    transitions after a given number of days
    """

    (r, c) = transitMatrix.shape
    degInitial = np.ones(r)
    degDaysMinus1 = np.matmul(np.linalg.matrix_power(transitMatrix,
```

```
     days - 1), degInitial)
    degDays = np.matmul(np.linalg.matrix_power(transitMatrix, days),
     degInitial)
    Rxy = np.zeros((r,c))

    del r
    del c

    (p, q) = np.nonzero(transitMatrix)

    for nz in range(len(p)):
        i = p[nz]
        j = q[nz]
        Rxy[i, j] = (transitMatrix[i, j] * degDaysMinus1[j]) / degDays[i]


    del p
    del q
    return Rxy

#This completes the definition of a function


def empiricalDays(returnMatrix, roughnessMatrix, numBins, step):
    """ This function computes the emperical transition matrix with
        the given day step (cannot be less than 1) and number of bins
    """

    days, corps = returnMatrix.shape
    bins_count = numBins
    ret_hist, ret_bin_edges = np.histogram(returnMatrix, bins = bins_count,
     density = False)
    rough_hist, rough_bin_edges = np.histogram(roughnessMatrix,
     bins = bins_count, density=False)
    Pxy = np.zeros((bins_count ** 2, bins_count ** 2), dtype = np.int)


    for firm in range(corps):
        firm_ret = returnMatrix[2: , firm]
        firm_rough = roughnessMatrix[2: , firm]
```

```
        #Create a transition matrix
        length = len(firm_ret)

        #Lets fill the matrix with integers

        for i in range(0, length - step):
            n = bisect.bisect_left(ret_bin_edges, firm_ret[i]) - 1
            m = bisect.bisect_left(rough_bin_edges, firm_rough[i]) - 1
            k = bisect.bisect_left(ret_bin_edges, firm_ret[i + step]) - 1
            l = bisect.bisect_left(rough_bin_edges, firm_rough[i + step]) - 1
            Pxy[m * bins_count + n, l * bins_count + k] += 1

    return Pxy


def distanceKulLeibl(theorMatrix, epmiricalMatrix):
    """ Compute Kullback-Leibler distance between empirical transition matrix
        and computer using degree vectors
    """

    if (theorMatrix.shape == epmiricalMatrix.shape):
        print('Matrices are compatible')
    else:
        print('Input Matrices are wrong')


    D = []
    #Find nozero entries to reduce the number of computations
    vrmMatrix = np.add(theorMatrix, epmiricalMatrix)
    i, j = np.nonzero(vrmMatrix)

    for x in i:
        for y in j:
            if (theorMatrix[x, y] == 0 or epmiricalMatrix[x, y] == 0):
                pass
            else:
                D.append(epmiricalMatrix[x, y] * np.log2(epmiricalMatrix[x, y]
                 theorMatrix[x, y]))
```

```
    del vrmMatrix
    del i
    del j

    return np.sum(D)


#Write a function to plot a trajectory of a company in a phase space.

def corporateTrajectory(returnMatrix, roughnessMatrix, corp, days):
    """ Plot a trajectory of a company motions in a return roughness
    phase space
    """
    (c, d) = returnMatrix.shape
    if corp < 2 or corp > d:
        print('Corp number is out of bounds. Must be between 2 and {}'.
         format(d))
    else:
        x = list(returnMatrix[2: , corp])
        y = list(roughnessMatrix[2:, corp])
        mnx = np.amin(x)
        mxx = np.amax(x)
        mny = np.amin(y)
        mxy = np.amax(y)

        for i in range(0, len(x), days):
            fig, ax = plt.subplots(1, 1, figsize=(20,20))
            plt.subplot(1, 1, 1)
            tck, u = splprep([x[i : i + days],y[i : i + days]], u=None,
             s=0.0, per=1)
            u_new = np.linspace(u.min(), u.max(), 1000)
            x_new, y_new = splev(u_new, tck, der=0)
            plt.plot(x[i : i + days], y[i : i + days], 'ko',
             markersize = 16)
            plt.plot(x_new, y_new, 'b--')
            plt.xlim(-0.06, 0.06)
            plt.ylim(-0.06, 0.06)
            plt.tick_params(axis='both', which='major', labelsize=45)
            plt.xticks(rotation=70)
            plt.yticks(rotation=20)
```

```
            gradient = np.linspace(0, 1, 100).reshape(1, -1)
            plt.imshow(gradient , extent=[-0.07, 0.06, -1, 1],alpha = 0.6,
             aspect='auto', cmap='RdYlGn')
            plt.xlabel('Return', fontsize = 80)
            plt.ylabel('Roughness', fontsize = 80)

            plt.title('{} company From {} to {}'.format(names[corp],
             original_data['Unnamed: 0'][i], original_data['Unnamed: 0']
             [i + days]), y=1.08, fontsize = 50)
            plt.tight_layout()
            plt.savefig('Trajectory_{}_{}_to_{}'.format(names[corp], i,
             i + days))
            plt.gcf().clear()
            plt.close("all")


def normalMatrix(transitMatrix):
    """
    This function returns a normalized matrix.
    """

    row_sums = transitMatrix.sum(axis=1)
    normal = np.copy(transitMatrix.astype(np.float64))
    m, n = transitMatrix.shape
    for i in range(m):
        if row_sums[i] == 0:
            continue
        else:
            for j in range(n):
                normal[i, j] = float(transitMatrix[i, j]) / float(row_sums[i])


    return(normal)

def DEU(transitMatrixNorm, epsilon):
    """
    Compute Information Decoposition for a given transition matrix.
    Return three lists.
    """
    #Compute eigenvalues and eigenvectors
```

```
toll = epsilon
val, vec = linalg.eigs(np.transpose(transitMatrixNorm),
 k=1, which = 'LM')

vec.real[abs(vec.real) < toll] = 0.0
fixed_vec = np.absolute(np.real(vec))

norm_vec = np.zeros(len(fixed_vec))

(m, n) = transitMatrixNorm.shape

s = np.sum(fixed_vec)
for i in range(len(fixed_vec)):
    norm_vec[i] = fixed_vec[i] / s

H_list = []
for pi in range(len(norm_vec)):
    if norm_vec[pi]  == 0.0:
        pass
    else:
        H_list.append(norm_vec[pi] * np.log2(norm_vec[pi]))

H = -np.sum(H_list)
H = np.round(H, decimals = 4)

#Now we need to compute the little h
h_list = []
for row in range(m):
    vrm = []
    for column in range(n):
        if transitMatrixNorm[row, column] == 0:
            pass
        else:
            vrm.append(transitMatrixNorm[row, column] * np.log2
             (transitMatrixNorm[row, column]))
    h_list.append(norm_vec[row] * np.sum(vrm))


h = np.round(-np.sum(h_list), decimals = 4)
D = np.round(H - h, decimals = 4)
```

```
    #Multiply matrices (transpose * transpose)
    transit_sqr = np.matmul(transitMatrixNorm, transitMatrixNorm)

    #Compute U
    U_list = []
    for row in range(m):
        vrm = []
        for column in range(n):
            if transitMatrixNorm[row, column]==0 and transit_sqr[row,
             column]==0:
                pass
            elif transitMatrixNorm[row, column] == 0:
                vrm.append(-(transit_sqr[row, column] * np.log2
                 (transit_sqr[row, column])))
            elif transit_sqr[row, column] == 0:
                vrm.append(transitMatrixNorm[row, column] * np.log2
                 (transitMatrixNorm[row, column]))
            else:
                vrm.append(transitMatrixNorm[row, column] * np.log2
                 (transitMatrixNorm[row, column])-
                           transit_sqr[row, column] * np.log2
                            (transit_sqr[row, column]))
        U_list.append(norm_vec[row] * np.sum(vrm))


    U = np.round(np.sum(U_list), decimals = 4)
    E = np.round(h - U, decimals = 4)

    return D, E, U


def plotDEU(returnMatrix, roughnessMatrix, numBins, epsilon, days):
    """
    Plot DEU stack plot based on the emperical transition matrix.
    """

    D_list = []
    E_list = []
    U_list = []
```

```
    for day in range(1, days):
        A = normalMatrix(empiricalDays(returnMatrix, roughnessMatrix,
         numBins, day))
        D, E, U = DEU(A, epsilon)
        D_list.append(D)
        E_list.append(E)
        U_list.append(U)


    x = [x for x in range(1, days)]
    labels = ["D ", "E", "U"]

    fig, ax = plt.subplots()
    ax.stackplot(x, D_list, E_list, U_list, labels=labels)
    ax.legend(loc=1)
    #plt.show()
    plt.savefig('Information decomposition for {} days.png'.format(days),
     dpi = 600)
    print("D_list {}".format(D_list))
    print("E_list {}".format(E_list))
    print("U_list {}".format(U_list))



def corpTransitMtx(returns, roughness, numBins, firm):
    """
    Compute transition matrix for a particular company
    """
    corp_ret = returns[2: , firm]
    corp_rough = roughness[2: , firm]

    bins_count = numBins

    corp_ret_hist, corp_ret_bin_edges = np.histogram(corp_ret,
                              bins = bins_count,   density = False)

    corp_rough_hist, corp_rough_bin_edges = np.histogram(corp_rough,
                              bins = bins_count, density=False)
```

```
    #Create a transition matrix

    trans = np.zeros((bins_count ** 2, bins_count ** 2))

    length = len(corp_ret)

    #Lets fill the matrix with integers

    for i in range(0, length - 1):
        n = bisect.bisect_left(corp_ret_bin_edges, corp_ret[i]) - 1
        m = bisect.bisect_left(corp_rough_bin_edges, corp_rough[i]) - 1
        k = bisect.bisect_left(corp_ret_bin_edges, corp_ret[i + 1]) - 1
        l = bisect.bisect_left(corp_rough_bin_edges, corp_rough[i + 1]) - 1
        trans[m * bins_count + n, l * bins_count + k] += 1


    return trans


def corpDEUPlot(D, E, U, corp):
    """
    Plot DEU information decomposition for each company
    """
    names = np.delete(list(pd.read_csv('Log_return_EXCEL.csv')), 0)
    fig = plt.figure(figsize=(30,30))


    plt.subplot(1,1,1)
    plt.title('Information decomposition for {}'.format(names[corp]),
     y=1.08, fontsize = 30)
    plt.pie([D, E, U], labels = ['D={}'.format('{0:.3f}'.format(D)),
     'E={}'.format('{0:.3f}'.format(E)), 'U={}'.format('{0:.3f}'.format(U))],
     autopct='%1.1f%%', textprops={'fontsize': 30})
    plt.savefig('Information decomposition for {}.png'.format(names[corp]))

    plt.gcf().clear()
    plt.close("all")
```

```
def histOccur(returns, roughness, numBins, norm=True):
    """
    Compute the the list of occurances in each cell of phase space of
    the size numBin**2
    """

    days, corps = returns.shape
    ret_hist, ret_bin_edges = np.histogram(returns, bins = numBins,
     density = False)
    rough_hist, rough_bin_edges = np.histogram(roughness, bins = numBins,
     density=False)
    hist = np.zeros((numBins ** 2), dtype = np.int64)

    for firm in range(0, corps):
        firm_ret = returns[2: , firm]
        firm_rough = roughness[2: , firm]
        #Create a transition matrix
        length = len(firm_ret)

    #Lets fill the matrix with integers

        for i in range(0, length):
            n = bisect.bisect_left(ret_bin_edges, firm_ret[i]) - 1
            m = bisect.bisect_left(rough_bin_edges, firm_rough[i]) - 1
            hist[m * numBins + n] += 1



    if norm == True:
        return np.true_divide(hist, np.sum(hist))
    else:
        return hist




def Itdays(transition, hist):
    """
    Computes the I for the given transition matrix (must be normalized)
     so is the hist vector
```

```
    """

    I = 0
    row, column = transition.shape
    for i in range(row):
        for j in range(column):
            if (transition[i, j] == 0 or hist[i] == 0 or hist[j] == 0):
                pass
            else:
                I += transition[i, j] * np.log2(transition[i, j]/
                 (hist[i]* hist[j]))


    return I




def Ilist(returns, roughness, numBins, days):
    """
    Create the list with I values for the given period of time
    """

    hist = histOccur(returns, roughness, numBins, norm = True)
    I_list = []

    for i in range(days):
        tmp = Itdays(normalMatrix(empiricalDays(returns, roughness,
         numBins, i)), hist)
        I_list.append(tmp)
        print("Day {}".format(i))

    return I_list



def plot_mtrx(hist, numBins, title, plot=True):
    """
    Plot matrix of cells in phase space
```

```
    """

    mtrx = np.zeros((numBins, numBins))

    for row in range(numBins):
        for column in range(numBins):
            mtrx[numBins - row-1,column] = hist[row * numBins + column]


    if plot == True:
        plt.imshow(mtrx, cmap='magma', vmin = 0, vmax = 1,
         interpolation = 'bilinear')
        plt.savefig('J_matrix_{}_days.png'.format(title), dpi = 300)
    else:
        np.savetxt('J_matrix_{}_days.csv'.format(title), mtrx,
         delimiter = ',')




def Jlist(returns, roughness, numBins, days):
    """
    Create the list of J values for the given day
    """
    hist = histOccur(returns, roughness, numBins, norm = True)
    J_list = np.zeros((numBins **2))
    mtrx = normalMatrix(empiricalDays(returns, roughness, numBins, days))

    for row in range(numBins ** 2):
        for column in range(numBins ** 2):
            if (mtrx[row, column] == 0 or hist[column] == 0 or hist[row] == 0):
                pass
            else:
                J_list[row] += hist[row] * (mtrx[row, column] / hist[column]) * np


    return J_list
```

```
def weight_list(data, weights):
    """
    Create a list with weights for the given data set
    """
    original_data = pd.read_csv(data)
    corps = list(original_data.columns.values)
    corps.pop(0)

    #Create a list of weights
    weight = np.nan_to_num(pd.read_csv(weights), copy=True)

    vesa = list(weight[0 : ,2])
    wght_list = [None] * len(corps)
    names = list(weight[0 : ,1])

    for comp in corps:
        if comp in names:
            n = names.index(comp)
            wght_list[corps.index(comp)] = round(vesa[n], 4)
        else:
            m = corps.index(comp)
            wght_list[m] = 0.01

    return wght_list


def dates_list(data):
    """
    Get the list of dates in the data set
    """

    original_data = pd.read_csv(data)
    dates = list(original_data['Date'])

    return dates

def plot_clouds(dates, returns, roughness, weights, min_window,
max_window, start_day, finish_day):
    """
    Plot the clouds for each trading day
```

```
"""
sizes = [i * 300 for i in weights]

for i in range(start_day, finish_day):
    x = returns[i, 0: ]
    y = roughness[i, 0: ]
    plt.figure(figsize=(35,35))

    plt.subplot(1, 1, 1)
    plt.title('The phase portrait of \nS&P500 on {}'.format
     (dates[i]), y=1.08, fontsize = 120)
    plt.scatter(x, y, c=weights, s=sizes, cmap = 'rainbow',
     alpha = 0.65)
    plt.xlim(min_window, max_window)
    plt.ylim(min_window, max_window)
    plt.tick_params(axis='both', which='major', labelsize=70)
    plt.xticks(rotation=70)
    plt.yticks(rotation=20)
    #gradient = np.linspace(0, 1, 100).reshape(1, -1)
    #plt.imshow(gradient , extent=[-0.07, 0.06, -1, n],alpha = 0.4,
     aspect='auto', cmap='RdYlGn')
    plt.xlabel('Return', fontsize = 140)
    plt.ylabel('Roughness', fontsize = 140)
    plt.tight_layout()


    filename = 'daily_plot_'+str(i)+'.png'
    plt.savefig(filename)
    plt.gcf().clear()
    plt.close("all")
```

Extreme events and emergency scales

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created on Fri Jul  5 22:14:11 2019

@author: benjamin
"""

import numpy as np
import seaborn as sns
sns.set(rc={'figure.figsize':(8,8)})
import matplotlib.pyplot as plt
import time
import pandas as pd
from scipy.stats import gaussian_kde
from datetime import datetime
from scipy.optimize import curve_fit
from scipy.stats import norm
from itertools import cycle




def days_between(d1, d2):
    d1 = datetime.strptime(d1, "%m/%d/%Y")
    d2 = datetime.strptime(d2, "%m/%d/%Y")

    return abs((d2 - d1).days)

def prob_time(data_path, threshold_pos, threshold_neg):
    """Returns distribution of excesses over threshold versus time periods
    """

    df = pd.read_csv(data_path, encoding = "ISO-8859-1")
    df_over = df.loc[(df['Return'] > threshold_pos) | (df['Return']  \
     < threshold_neg)]
```

```
    l_dates = np.array(df_over['Date'])
    n = len(l_dates)
    days = []

    for i in range(1, n):
        days.append(days_between(l_dates[i-1], l_dates[i]))

    print(max(days))
    np.savetxt('days_intervals_threshhold_{}.txt'.format(threshold_pos),
     np.array(days).astype(int), delimiter = ',', fmt='%i')
    g = sns.distplot(days, bins=60, kde=False, rug=False)
    g.axes.set_yscale('log', basey = 2)
    g.axes.set_xscale('log', basex = 2)
    return days



def excess_points(df, col, threshold, bins):
    """Retuns points of excess (arrays) by a given column name, threshold,
    and a number of bins to use. X is normalized by the threshold.
    """

    df1 = df.loc[(df[col] < -threshold) | (df[col] > threshold)]

    dfex_norm = np.multiply(np.array(df1[col]), 1/threshold)

    high, edge = np.histogram(dfex_norm, bins = bins)
    x = []
    y = []

    for k in range(len(high)):
        if high[k] == 0:
            pass
        else:

            x.append((edge[k] + edge[k+1]) / 2)
            y.append(high[k])
```

```
    return np.array(x), np.array(y)


def weighted_avg_and_std(values, weights):
    """
    Return the weighted average and standard deviation.

    values, weights -- Numpy ndarrays with the same shape.
    """
    average = np.average(values, weights=weights)
    # Fast and numerically precise:
    variance = np.average((values-average)**2, weights=weights)
    return (average, np.sqrt(variance))



def gau_points(x, mu, std):
    """y coordinates associated with the given x points, mean
    and standard deviation.
    """

    y = (1/(np.sqrt(2*np.pi)*std))*np.exp(-(x-mu)**2/(2*(std**2)))

    return y


def calc_parabola_vertex(x1, y1, x2, y2, x3, y3):
"""
Adapted and modifed to get the unknowns for defining a parabola:
http://stackoverflow.com/questions/717762/how-to-calculate-
the-vertex-of-a-parabola-given-three-points
"""

denom = (x1-x2) * (x1-x3) * (x2-x3);
A     = (x3 * (y2-y1) + x2 * (y1-y3) + x1 * (y3-y2)) / denom;
B     = (x3*x3 * (y1-y2) + x2*x2 * (y3-y1) + x1*x1 * (y2-y3)) / denom;
C     = (x2*x3*(x2-x3)*y1+x3*x1*(x3-x1)*y2+x1*x2*(x1-x2)*y3)/denom;

return A,B,C
```

```
def parabola_coord(x, a, b, c):
    """Calculate y coordinates for f(x) = ax^2+bx+c
    """
    y = np.multiply(a, np.square(x)) + np.multiply(x, b) + c

    return y



def plot_prb_lin(x_coord, y_coord):
    """ Plot scater plot with points, and parabola plus lines
    """
    # plot a parabola
    x_par = np.linspace(-0.6, 0.6, 100)
    a, b, c = calc_parabola_vertex(-0.6,0,0,5,0.6,0)
    y_par = parabola_coord(x_par, a, b, c)




    plt.figure(figsize=(12,12))
    plt.scatter(x_coord, y_coord, c = 'blue', s = 90)
    plt.plot(x_par, y_par, c = 'orchid', linewidth = 7.0,
     linestyle = '-.')
    plt.plot([-2.56, 0], [0, 2.56], c = 'limegreen', linewidth = 7.0,
     linestyle = ':')
    plt.plot([2.56, 0], [0, 2.56], c = 'limegreen', linewidth = 7.0,
     linestyle = ':')
    plt.plot([-0.1, -1.5], [5, 0], c = 'orangered', linewidth = 7.0,
     linestyle = '--')
    plt.plot([0.1, 1.5],[5, 0],  c = 'orangered', linewidth = 7.0,
     linestyle = '--')
    plt.xticks(fontsize=36, rotation=0)
    plt.yticks(fontsize=36, rotation=0)
    plt.ylabel(r'$\ln(number \; of \; samples)$', fontsize = 30)
    plt.xlabel(r'$\ln \left( \frac{return \; excess}{threshold}
     \right)$', fontsize = 30)
    plt.show()
```

```
def plot_prob_days(days, bins):
    """Plot periods between excesses as a histogram with a straight line
    """

    x_cr = np.linspace(4, 130, 100)
    y_cr = [11500/(p ** 2) for p in x_cr]
    x_zr = np.linspace(2, 300, 100)
    y_zr = [285/p for p in x_zr]

    plt.figure(figsize = (16, 16))
    plt.hist(days, bins, color = 'lightskyblue', edgecolor = 'k')
    plt.yscale('log')
    plt.xscale('log')
    plt.plot(x_cr, y_cr, c= 'orangered', linewidth = 17.0, alpha = 0.9)
    plt.plot(x_zr, y_zr, c= 'orangered', linestyle = ':',
     linewidth = 17.0, alpha = 0.9)
#    plt.plot([260, 1.2], [1, 188], c= 'orangered', linestyle = ':',
linewidth = 17.0, alpha = 0.9)
    plt.xticks(fontsize=60, rotation=0)
    plt.yticks(fontsize=60, rotation=0)
    axes = plt.gca()
#    axes.set_xlim([xmin,xmax])
    axes.set_ylim([0.5,1000])
    plt.ylabel('Number of data samples', fontsize = 50)
    plt.xlabel('Days between sequential extreme events', fontsize = 50)
    plt.show()



def returns_log_lin(data, bins):
    """ Plot the histogram in log linear scale with lines and etc
    """
    x_lp = np.linspace(-0.228997, -0.0001, 60)
#    a, b, c = calc_parabola_vertex(-0.228997,1,-0.0935696,2,-0.056327,4)
    y_lp = np.divide(0.2, np.power(np.abs(x_lp), 1))

    x_lp2 = np.linspace(-0.10, 0.001, 60)
    y_lp2 = np.divide(0.00015, np.power(np.abs(x_lp2), 3.5))
```

```
    x_lr2 = np.linspace(0.001, 0.10, 60)
    y_lr2 = np.divide(0.00015, np.power(np.abs(x_lr2), 3.5))

    x_lr = np.linspace(0.0001, 0.12, 60)
    y_lr = np.divide(0.2, np.power(np.abs(x_lr), 1))

    plt.figure(figsize = (16, 16))
    plt.hist(data[1:], bins, color = 'lightskyblue', edgecolor = 'k',
     alpha = 0.65)
    plt.plot([-0.05, 0], [0.5, 1897], c= 'orangered', linewidth = 11.0,
     linestyle = '--')
    plt.plot([0.05, 0], [0.5, 1897], c= 'orangered', linewidth = 11.0,
     linestyle = '--')
    plt.plot(x_lp, y_lp, c= 'orangered', linewidth = 11.0)
    plt.plot(x_lr, y_lr, c= 'orangered', linewidth = 11.0)
    plt.plot(x_lp2, y_lp2, c= 'orangered', linewidth = 11.0,
     linestyle = ':')
    plt.plot(x_lr2, y_lr2, c= 'orangered', linewidth = 11.0,
     linestyle = ':')
    axes = plt.gca()
    axes.set_ylim([0.5,2100])
    plt.yscale('log', basey = 2)
    plt.xticks(fontsize=40, rotation=90)
    plt.yticks(fontsize=40, rotation=0)
    plt.ylabel('Number of accurances', fontsize = 50)
    plt.xlabel('log return', fontsize = 50)
    plt.show()
```

```
def rev_product(k, l):
    """product of (k-1)(k-2)...(k-l)
    """
    prod = 1
    for step in range(l):
```

```
        prod *= k - (step + 1)

    return prod



def probability_weight(data, r):
    """Compute a probability weight for a given data set
    """
    #Sort the given data set
    b = 0
    data_s = np.sort(np.array(data))
    if r == 0:
        b = (1/len(data_s)) * sum(data_s)
        return b
    elif r < 0:
        print('r cannot be negative!')
    else:
        n = len(data_s)
        for i in range(r, n):
            b += (rev_product(i+1, r) / (rev_product(n, r))) * data_s[i]


        return b * (1 / n)



def lmoments4(data):
    """Compute the first 4 lmoments
    """
    mom = []
    l0 = probability_weight(data, 0)
    l1 = probability_weight(data, 1) * 2 - probability_weight(data, 0)
    l2 = (probability_weight(data, 2) *6-probability_weight(data, 1)*6+
          probability_weight(data, 0))
    l3 = (probability_weight(data, 3) *20-30*probability_weight(data, 2)+
          12 * probability_weight(data, 1) - probability_weight(data, 0))
    mom.append(l0)
    mom.append(l1)
    mom.append(l2 / l1)
```

```
    mom.append(l3 / l1)


    return mom



def lmom_threshold(data):
    """Determines a threshold based on L-Moments as prescrived by
    L-Moments for automatic threshold selection in extreme value analysis by
    Jessica Silva Lombda and Maria Isabel Fraga Alves
    """

    tau = lmoments4(data)
    minimum = min(data)
    maximum = max(data)

    distance = []
    threshold = []

    for i in range(20):
        #20 sample quantiles, starting at 25% by steps of 3.7%
        cut_off = (maximum - minimum) * (0.25 + 0.037 * i) + minimum
        temp_data = data[data >= cut_off]
        t = lmoments4(temp_data)
        gtau3 = tau[2] * ((1 + 5 * tau[2]) / (5 + tau[2]))
        distance.append(np.sqrt((t[2] - tau[2])**2 + (t[3] - gtau3)**2))
        threshold.append(cut_off)

        return threshold[np.argmin(distance)]



def threshold_change(data_frame, window):
    """Change of the thresholds depending on the window with Pos_Th
    and Neg_Th columns to be filled with new values
    """

    data_frame['Pos_Th'] = pd.Series()
```

```
    data_frame['Neg_Th'] = pd.Series()

    r,c = data_frame.shape

    for i in range(r - window):
        #Slice a window of return
        data_sliced = np.array(data_frame[i: i + window]['Return'])

        data_frame.iat[i + window, data_frame.columns.get_loc('Pos_Th')] =
         lmom_threshold(data_sliced[data_sliced >= 0])
        data_frame.iat[i + window, data_frame.columns.get_loc('Neg_Th')] =
         lmom_threshold(data_sliced[data_sliced <= 0])



    return data_frame


#Compute the range for threshold values based on
#Automated threshold selection methods for extreme wave analysis

def width_test(df, window_width):
    """Compute the window for the threshold existence
    """

    r, c = df.shape
    med_pos = []
    upper_bound_pos = []
    med_neg = []
    lower_bound_neg = []
    for i in range(window_width, r):
        df_chunk = df.loc[i - window_width: i][:]
        pos_values = np.array(df_chunk.loc[df_chunk['Return'] > 0]['Return'])
        p = len(pos_values) // 10
        neg_values = np.array(df_chunk.loc[df_chunk['Return'] < 0]['Return'])
        n = len(neg_values) // 10
        med_pos.append(np.median(pos_values))
        upper_bound_pos.append(np.sort(pos_values)[-p])
        med_neg.append(np.median(neg_values))
        lower_bound_neg.append(np.sort(neg_values)[n])
```

```
def width_test2(df, window_width):
    """Compute the window for the threshold existence WITHOUT the assumption
    of 10% inherited from the rule of thumb.
    """

    r, c = df.shape
    med_pos = []
    upper_bound_pos = []
    med_neg = []
    lower_bound_neg = []
    for i in range(window_width, r):
        df_chunk = df.loc[i - window_width: i][:]
        pos_values = np.array(df_chunk.loc[df_chunk['Return'] > 0]['Return'])
        p = len(pos_values) // 20
        neg_values = np.array(df_chunk.loc[df_chunk['Return'] < 0]['Return'])
        n = len(neg_values) // 20
        med_pos.append(np.median(pos_values))
        upper_bound_pos.append(np.sort(pos_values)[-p])
        med_neg.append(np.median(neg_values))
        lower_bound_neg.append(np.sort(neg_values)[n])



#    plt.figure(figsize = (64, 32))
#    x = [p for p in range(window_width, r)]
#    plt.plot(x, med_pos, c = 'orangered', linewidth = 2)
#    plt.plot(x, upper_bound_pos, c = 'blue', linewidth = 2)
#    plt.plot(x, med_neg, c = 'black', linewidth = 2)
#    plt.plot(x, lower_bound_neg, c = 'blue', linewidth = 2)
#    plt.fill_between(x, med_pos, upper_bound_pos, facecolor='green',
# alpha=0.5)
#    plt.fill_between(x, med_neg, lower_bound_neg, facecolor='orange',
# alpha=0.5)
#    plt.xticks(fontsize=80, rotation=0)
#    plt.yticks(fontsize=80, rotation=0)
#    axes = plt.gca()
#    axes.set_xlim([window_width,9840])
```

```
#    plt.show()

    return med_pos, upper_bound_pos, med_neg, lower_bound_neg



def change_thresh(m_pos, up_pos, m_neg, low_neg, step):
    """Compute how long on average one value of a threshold can survive
    """

    min_pos = min(m_pos)
    max_pos = max(up_pos)
    min_neg = min(low_neg)
    max_neg = max(m_neg)

    x_pos = np.linspace(min_pos + ((max_pos - min_pos) / step), max_pos -
                        ((max_pos - min_pos) / step), step)
    x_neg = np.linspace(min_neg + ((max_neg - min_neg) / step), max_neg -
                        ((max_neg - min_neg) / step), step)

    y_pos = []
    y_neg = []

    m = len(m_pos)

    for i in range(step):
        temp_pos = 0
        temp_neg = 0
        for j in range(m):
            if (x_pos[i] > m_pos[j]) and (x_pos[i] < up_pos[j]):
                temp_pos += 1
            elif (x_neg[i] > low_neg[j]) and (x_neg[i] < m_neg[j]):
                temp_neg += 1

        y_pos.append(temp_pos / m)
        y_neg.append(temp_neg / m)


    return x_pos, y_pos, x_neg, y_neg
```

```
def plot_thresh_belt(df, step, windows):
    """Plot saveral prbabilities of the chosen threshold. Step is
    for threshold increments, windows is a list of window values.
    For POSITIVE returns only!!!!
    """

    plt.figure(figsize = (20, 20))
    lines = ["-","--","-.",":"]
    linecycler = cycle(lines)

    for w in windows:
        m_pos, up_pos, m_neg, low_neg = width_test(df, w)
        xp, yp, _, _ = change_thresh(m_pos, up_pos, m_neg, low_neg, step)
        yp_vers = 1 - np.array(yp)
        plt.plot(xp, yp_vers, c = np.random.rand(3,), linestyle =
         next(linecycler), linewidth = 5.0, label = str(w) + ' days')

    plt.legend(fontsize = 40)
    plt.xticks(fontsize=40, rotation=0)
    plt.yticks(fontsize=40, rotation=0)
    plt.show()

def entropy(h):
    """The input is a real number, the output is a value of an
    entropy computed as -h * log(h) - (1-h)*log(1-h).
    There are several cases to avoid log(0)
    """
    if h == 0:
        return 0
    else:
        return - h * np.log(h) - (1 - h) * np.log(1 - h)


def entropy_check_list(x, y):
    """Extends the domain to make sure all the entropy graphs are
    complete and extended on the entire domain
    """
#add lower values
```

```
    if x[0] > 0.002:
        x_add = np.array([0, x[0]-0.001])
        y_add = np.array([0,  0])
        x = np.concatenate((x_add, x))
        y = np.concatenate((y_add, y))
    else:
        pass
#add upper values
    if x[-1] < 0.08:
        x_add = np.array([x[-1]+0.001, 0.08])
        y_add = np.array([0, 0])
        x = np.concatenate((x, x_add))
        y = np.concatenate((y, y_add))
    else:
        pass


    return x, y



def plot_thresh_entropy(df, step, windows):
    """Plot saveral uncertainty rates of the chosen threshold.
    Step is for threshold increments, windows is a list of
    window values. For POSITIVE returns only!!!!
    USE width_test2 in the cycle to drop the assumption of
    10% inherited from rules of thumb.
    """

    plt.figure(figsize = (20, 20))
    lines = ["-","--","-.",":"]
    clrs = cycle(['black', 'blue', 'red'])
    linecycler = cycle(lines)
    for w in windows:
        m_pos, up_pos, m_neg, low_neg = width_test2(df, w)
        xp, yp, _, _ = change_thresh(m_pos, up_pos, m_neg, low_neg, step)
        yp_vers = 1 - np.array(yp)
        yp_h = np.array([entropy(h) for h in yp_vers])
        xp, yp_h = entropy_check_list(xp, yp_h)
        table = np.stack((xp, yp_h), axis=-1)
```

```
        np.savetxt('entropy_curve_{}_days.csv'.format(w), table,
         delimiter = ',')
        #plt.plot(xp, yp_h, c = np.random.rand(3,), linestyle =
         next(linecycler), linewidth = 5.0, label = str(w) + ' days')
        plt.plot(xp, yp_h, c = next(clrs), linestyle = next(linecycler),
         linewidth = 5.0, label = str(w) + ' days')
    plt.axvspan(0, 0.0053, facecolor='g', alpha=0.4)
    plt.axvspan(0.00531, 0.01606, facecolor='y', alpha=0.4)
    plt.axvspan(0.01607, 0.075, facecolor='r', alpha=0.3)
    plt.legend(fontsize = 40)
    plt.xticks(fontsize=40, rotation=0)
    plt.yticks(fontsize=40, rotation=0)
    plt.ylabel('Uncertainty of threshold value', fontsize = 60)
    plt.xlabel('Threshold', fontsize = 60)
    plt.text(0.002, -0.03, 'I', fontsize=50, color = 'blue')
    plt.text(0.009, -0.03, 'II', fontsize=50, color = 'blue')
    plt.text(0.0455, -0.03, 'III', fontsize=50, color = 'blue')
    plt.show()



def entr_vs_prob():
    """A function that plots uncertainty versus various probability values
    """

    eta = np.linspace(0.001, 0.999, 150)
    h = -eta * np.log(eta) - (1 - eta) * np.log(1 - eta)
    plt.figure(figsize = (20, 20))
    plt.plot(eta, h, c = 'blue', linestyle = '-', linewidth = 6.0)
    plt.xticks(fontsize=40, rotation=0)
    plt.yticks(fontsize=40, rotation=0)
    plt.ylabel('Uncertainty', fontsize = 60)
    plt.xlabel('Probability', fontsize = 60)
    plt.show()
```