Application of Data Analytics to Cyber Forensic Data

A Major Qualifying Project Report

submitted to the Faculty

of the

Worcester Polytechnic Institute

in partial fulfillment of the requirements for the
Degree of Bachelor of Science

By

Yao Yuan Chow

Date: October 13, 2016

Approved:

Professor: George T. Heineman, Computer Science Advisor

Professor: Randy C. Paffenroth, Mathematical Sciences Advisor

Sponsored by:

MITRE Corporation

Table of Contents

Table of Figures

Table of Tables

Abstract

The goal of this project is to develop indicators of a cyber-attack by applying a new data analysis technique known as *Topological Data Analysis* on synthetic cyber data. In particular, the research and work completed throughout this paper centers around an open-source implementation, called Python Mapper, that helps users globally view forensic data and gain qualitative insights from the shape of the data. To assist in these efforts, we developed tools to directly import data into Mapper from existing data sources. Due to the exploratory nature of the project and the method, the outcome of the work completed was not developing new cyber analytics of malicious activity but creating a stopping point in the research where future collaborators can build off of our work.

Acknowledgements

Without the assistance of various individuals, this Major Qualifying Project would not have been possible. The amount of support and available resources I received throughout the project was invaluable. I would like to thank Dr. Alan Evans, my MITRE mentor, for his guidance throughout the project, his experience in data analysis, and knowledge of algebraic geometry. His assistance helped me meet necessary points of contact for the forensic data as well as guided me in the process of creating meaningful data visualizations. Also at the MITRE Corporation, I would like to thank Phillip Mothon who was the Project Lead responsible for ensuring that my time at MITRE was as productive and enriching as possible. I would also like to acknowledge Craig Wampler, Michael Kemmerer, and Ross Wolf for their involvement in accessing and pulling the cyber data and answering questions about BRAWL. Concerning the graphical user interface for the tool, I want to thank Shawn Chin for providing his feedback on the UI design. Last but not least, I would like to acknowledge Dr. Stanley Barr and Reid Gilman for their input on conditioning the data. I greatly appreciate their willingness to take time away from their projects to help me.

Throughout the mathematical component of this project, Professor Padraig Cathain's assistance was much appreciated. He answered any questions I had involving group theory and homology.

Finally, I would like to thank my project advisers for directing my efforts and keeping me on track. Professor George Heineman offered many suggestions regarding software design and customization of the tool. His help in reviewing this report and organizing the structure of the paper were also appreciated. In terms of data science, I would like to thank Professor Randy Paffenroth because his expertise in the field guided the efforts in the mathematical component of the project.

Executive Summary

In this age of technological advances in computing and anonymous access to overwhelming amounts of information, cyber threats are prevalent on a national scale. In 2010, cyber security experts witnessed the first modern cyber warfare weapon called Stuxnet, a sophisticated cyber weapon that targeted a uranium enrichment plant in Natanz, Iran. With the upcoming release of the motion picture *The Interview* in 2015, there was a data breach within Sony Pictures that led to the release of employee confidential information and files for a number of Sony movies onto the Internet. There have been other cyber incidences such as the Office of Personnel Management (OPM) data breach and the cyber-attacks on Twitter and Netflix. With advanced persistent threats, malware, and phishing plaguing cyberspace, there is no all-encompassing method to discern normal users from malicious actors. However, utilizing data analytics is an attempt to generalize such behaviors of suspicious activity in order to promote informed and efficient decision-making, otherwise known as cyber threat intelligence.

Residing within the realm of Big Data, cyber data or event log data can be complex and highly dimensional consisting of host names, ID numbers, command-line commands, and various other fields. This complexity and high-dimensionality introduces many problems in addition to traditional roadblocks with large datasets. Topological Data Analysis (TDA) gives rise to methods that extract and highlight topological features in order to summarize the data. The idea is that data possesses shape that may convey meaningful information. With its origins from combinatorics, discrete geometry, linear algebra, and topology, TDA methods use three invariants studied in topology to capture the shape of the data [Chintakunta, 2016]. First introduced by Gurjeet Singh, Gunnar Carlsson, and Facundo Memoli, the Mapper algorithm is a method for Topological Data Analysis, emphasizing the goal of giving shape to data and interpreting this shape [Singh, 2007]. This method looks to build a compressed representation of

data to highlight the significant features. The Mapper algorithm achieves this result by using topological ideas that preserve closeness or similarities but alter distances of large magnitude. The Python Mapper is an implementation of the Mapper algorithm written with a graphical user interface that enables beginners to quickly acquaint themselves with its many features such as filter selection and node coloring options.

The cyber data discussed throughout this paper is a product of the BRAWL project; BRAWL is an abbreviation for Blue versus Red Agent War-game Evaluation. As the name implies, there are two competitors in the cyber game; the red team agents simulate cyber intrusions while blue team agents have the task of detecting this malicious activity and developing analytics for that behavior. These games occur on a hyper-sensing network that generates data and records logs of network activity in real-time. After each BRAWL cyber game, the output is a multitude of indicators that highlight malicious behavior or by-products of red team activity. These indicators are stored in the Cyber Analytic Repository (CAR) for further study and testing, no matter how trivial or groundbreaking. The overall goal of applying the Mapper to this data was to gain qualitative understanding of the data from a different perspective.

Methodology

The primary goal of this project is the design and implementation of a system that combined the capabilities of the Python Mapper with the ability to pull data directly from a cloud repository for forensic data. Once completed, carrying out case studies on the cyber data, testing different filters and observing the outcomes formed the general procedure in evaluating the software system. The final step, afterwards, was to explore the visualizations based on criteria tailored to this project.

Results and Conclusions

This work has been largely exploratory so far and the visualizations included in this report need careful study (based on TDA) to understand the results. There are four primary findings resulting from research and experimenting with the case studies.

1. The dimensionality of the data can be reduced by removing all fields except the Data Model fields. Doing so does not result in the loss of information.
2. Loops in the visualizations of each case study confirm some cyclic behavior prevalent in the cyber data.
3. Often, there were too many noisy or non-malicious data points to conclude whether or not these loop features indicated malicious behavior.
4. Further study of the TDA approach using the cyber data requires instances of malicious behavior in the data set.

Future work should build upon these findings to progress towards developing cyber analytics for cyber threat intelligence. One direction for further study would be to incorporate machine learning into the Mapper workflow. In terms of filtering, a model trained using a machine-learning algorithm and labeled cyber data can act as a filter function for assigning numerical scores to each data point based on likelihood of malicious intent. Of course, data preparation should entail labeling each data point on whether or not it was a blue team or red team member. Once new indicators of attack have been created from analysis, a new model trained using these indicators would act as part of a cyber detection system. Another idea for future work is developing a better data transformation that would highlight the purpose of each data point and its impact on the network; the user should be able to each data point in relation to type and obtain a quick overview of network activity. Although transforming the data into

continuous vectors using available cyber analytics, this transformation was only one way of incorporating meaning into each data point

1 Introduction

In this age of technological advances in computing and anonymous access to overwhelming amounts of information, cyber threats are prevalent on a national scale. In 2010, cyber security experts witnessed the first modern cyber warfare weapon called Stuxnet, a sophisticated cyber weapon that targeted a uranium enrichment plant in Natanz, Iran. Its mission was to search for specific software that monitored and controlled the programmable logic controllers of the nuclear centrifuges, intercept communication, and sabotage the centrifuges [Langner, 2011]. This program was the first of its kind to physically damage hardware while remaining undetected. In 2015, with the upcoming release of the motion picture *The Interview*, there was a cyber-attack on Sony Pictures that led to the release of employee confidential information and files for a number of Sony movies onto the Internet. After several days and various warnings/threats from a cyber terrorist group, the attacks took advantage of file-sharing hubs to distribute Sony files, film contracts, personal information on celebrities, and employee health records [Haggard, 2015]. There have been other incidents in which adversaries have stolen data without detection; these adversaries, however, hide within the network for long periods of time and steal much larger volumes of data. These attacks showcase the need to strengthen defenses against cyber threats.

The European Union Agency of Network and Information Security (ENISA), a center of network and information security expertise, released a report titled *ENISA Threat Landscape 2015* to summarize the trends and prominence of current cyber threats. In ascending order, the top ten cyber threats of 2015 were malware, web-based attacks, web application attacks, botnets, denial of service, physical damage/theft, insider threat, phishing, spam, and exploit kits. As noted in the report, major actors for these threats consisted of cyber criminals and terrorists, insiders, online social hackers, nations/states, corporations, and hacktivists [Marinos, 2016]. Trends for

the top three threats were increasing in 2015. Aside from this report, other companies/corporations specializing in cyber security also released similar annual summaries that describe their findings of these attacks and provide a brief list of mitigation techniques for each one.

The purpose of data analytics is to gain insight from information represented in the form of records, graphs, or text by observing and extracting important features. For example, a U.S retailer used analytic methods to observe consumer-purchasing behavior. Through this process, the retailer learned that among the major life events, the most profitable to its bottom line was pregnancy. Therefore, the retailer offered specific coupons and incentives for its pregnant shoppers, keeping in mind that some items would be in higher demand depending upon the individuals [Dietrich, 2015]. Most companies and corporations rely on data analytics to compete with each other. In addition to major contributions in industry and medicine, the application of data analytics can benefit many other facets of science and engineering, including cyber security. With attacks such as intrusion, advanced persistent threats, malware, and phishing plaguing cyberspace, there is no clear method to discern normal users from malicious actors. However, these attacks follow certain procedures that can be categorized. Utilizing data analytics is an attempt to generalize such behaviors of suspicious activity in order to promote informed and efficient decision-making, otherwise known as cyber threat intelligence.

In many papers pertaining to Topological Data Analysis (TDA), one of the common motivating factors for pursuing this method of analysis has been the increasing size and complexity of data. Improvements and optimizations in modern computing have allowed for the collection of massive amounts of data and its storage; such data could possibly reside in high (or infinite) dimensional spaces. Some examples include medical information from patients, gene

expressions from DNA, preferences/keywords from a search engine, and even data collected from a computer network. Additionally, several industries have taken the lead to exploit the data. Credit card companies collect transaction information to identify fraudulent purchases, and social media sites have become major data sources for both industry and academia [Dietrich, 2015]. The terminology used to describe data of this magnitude is unsurprisingly "Big Data". Traditionally, analysts interpret data visually by charting or graphing data in two or even three dimensions to find trends or patterns; the shape of the data is significant to observe during data analysis. The problem that topological data analysis strives to solve is to provide generalizations and conclusions based on the shape of the data.

This paper begins with an overview of cyber threat intelligence and existing cyber threats. Next, the Problem chapter presents the obstacles in developing predictive analytics for cyber intrusions, while also posing Topological Data Analysis (TDA) as a possible solution to these challenges. Mathematics behind TDA and Overview of TDA chapters focus on introducing TDA, the Mapper method, and the mathematical concepts that support them. Additionally, a section in the Overview of TDA chapter also described the provided cyber data for analysis. Finally, the Methodology chapter outlines developing an application for automating an open-source implementation of Mapper and applying the Mapper method to the cyber data. Results after applying Mapper can be found in the Results chapter.

2 Background

2.1 Tactical Cyber Threat Intelligence

The goal of cyber threat intelligence is to understand existing cyber threats, to share information with the responsible parties, and to enable organizations in proactively defending against future attacks. There exist three levels of cyber intelligence including Strategic, Operational, and Tactical Cyber Intelligence. Since the research and effort of this paper pertains to developing indicators of adversarial behavior, this paper focuses on cyber threat intelligence at the tactical level. Tactical Cyber Intelligence (TCI) has the purpose of monitoring network activity, examining and assessing the strengths and vulnerabilities of an organization's network defenses, and ascertaining patterns and techniques employed by adversaries. At this level, cyber threat intelligence positions the organization to progress towards a proactive cyber security posture that is both reactive and predictive in nature [INSA, 2015]. To reach this point, TCI provides context and relevance to terabytes of data and transform the data into actionable intelligence. Action intelligence, to be precise, "refers to information that has been integrated, processed, and analyzed from various sources so that customers can use it to meet their threat information needs" [INSA, 2015]. In doing so allows an organization to mitigate its enterprise risk, to gather threat information following a cyber intrusion in the form of indicators of compromise (IOC), and to better equip its cyber defenses against all adversaries.

To defend against the increase in cyber-attacks and threats, several governments and private companies have developed cyber threat intelligence centers and updated cyber security measures. Each center is responsible for the research and mitigation of cyber threats, and the sharing of intelligence and collaboration between government-specific organizations [Dalziel, 2015]. A few examples of government-led cyber threat intelligence centers include the Cyber Threat Intelligence Integration Center (CTIIC) for the U.S., Australian Cyber Security Centre (ACSC),

and the European Union Agency for Network and Information Security (ENISA). Other organizations involved in cyber threat intelligence collection and haring include the National Security Agency, the Department of Defense Cyber Crime Center, the NSA/CSS Threat Operations Center and a wide array of other agencies. In keeping with the theme of cyber threat intelligence at the tactical level, the Cyber Intelligence Lifecycle section introduces the general process of developing actionable intelligence from the collection and analysis of raw data. Moving onto existing cyber threats, the next section introduces the cyber threat environment, a classification of the different, recorded cyber threats. Finally, to complete the discussion on cyber threat intelligence, the focus changes to Advanced Persistent Threats, a fast-growing cyber security concern.

2.2 Cyber Intelligence Lifecycle

The Cyber Intelligence Lifecycle outlines the necessary steps for the collection and collaboration of cyber threat intelligence. As a component of TCI, there are seven steps in the lifecycle that include specifying requirements, collection, process and exploitation, analysis and production, dissemination, consumption, and feedback [INSA, 2015]. The first step in the life cycle is identifying the information and data to collect based on designated requirements and formulate a plan to acquire this information. For instance, an organization seeks to ensure that its supply chain is secure, so its cyber intelligence team must gather information about its suppliers such as history of transactions or past cyber threat incidences. In collection, this step entails selecting relevant sources for information retrieval such as news feeds, whitepapers, technical reports, or cyber threat intelligence sharing venues; in the supply chain example, a possible data source could be customer reviews or consumer ratings. Next, the processing and exploitation stage suggests extracting insights and trends from the data to create a broader intelligence product for cyber threat intelligence and decision-making. Doing so requires “an analytic process

that is repeatable, easy to implement, and produce valuable information" [INSA, 2015]. The next step in the cyber intelligence lifecycle involves packaging and delivering any actionable intelligence or assessments produced in the previous stage to sponsors, management, or other relevant parties during analysis and production step. Examples of cyber intelligence products could include newly discovered vulnerabilities, cyber threat indicators, or general technical insights. During the dissemination step, the recipients of the information have analysts and senior executives review the actionable intelligence; understanding the findings from the analysis step would be essential for determining what actions to be taken afterwards. If the sponsor receiving this intelligence discovers that one of its suppliers does promote a good security posture, the sponsor may halt all transactions with this supplier. Consumption of information involves assessing how it affects the business and mission objects. Finally, the feedback phase of this lifecycle is very important for determining if the intelligence products met the customer's needs/requirements and allowing cyber intelligence operators to improve methodologies for future intelligence collection.

2.3 Cyber Threat Environment

Before describing the taxonomy of cyber threats, consider all the terms used to discuss cyber threats such as cyber adversaries, cyber-attacks, disruptions, and cyber security incidents. Formally, a cyber adversary is an individual or organization that carries out cyber-attacks, cyber-crime, or cyber espionage. Examples of cyber adversaries can vary from foreign state-sponsored adversaries, organized criminals, or issue-motivated groups seeking attention to their causes. Cyber-attacks, building off of cyber adversaries, are "deliberate acts through cyber space to manipulate, destruct, deny, degrade, or destroy computers or networks in order to compromise national security, stability or prosperity" [ACSC, 2016]. The Stuxnet program and the incidents with the film *The Interview* would be examples of cyber-attacks; although cyber-attacks can be

impactful and damaging economically, cyber adversaries also conduct malicious activity that act as simply nuisances, hindering a system or organization but with minimal damage. Such activity, known as disruption of services or networks, includes denial of service, web defacement, and electronic graffiti. Hacktivists and issue-motivated groups utilize these methods to attract attention to themselves and their cause by disrupting social events, business dealings, and government activities; these adversaries usually take advantage of the vulnerabilities or poor security within a system in order to carry out their aims. Staying with this idea of compromises to a security system, a cyber security incident is "any activity that may threaten the security of a system or its information whereas a compromise is an incident where a system or its information was successfully harmed"; an example would be the extraction of information from a computer network [ACSC, 2016]. After understanding these terms, the next topic to discuss would be the taxonomy of cyber threats executed by these cyber adversaries.

At the top of the threat taxonomy, according to an ENISA threat report for 2015, the first cyber threat to discuss is malicious software or malware. Malware provides unauthorized access to a system, allowing adversaries to harm or disrupt the system depending on their goals [ACSC, 2016]. The results are usually heavy costs and damages, affecting a single user in the best case or an entire country in the worst case. Cyber threats involving malware have been the most predominant cyber threat for the past two years according to both ACSC and ENISA. There is a program based in Australia called the Australian Internet Security Initiative (AISI) that provides daily notifications to its participants that alert them of IP addresses on their networks that are potentially vulnerable to or infected by malware [ACSC, 2016]. Between October 2014 and January 2015, the program has reported over 15,000 malware compromises daily to Australian Internet Service Providers. Aside from Australia, other malware incidents occur around the

world. The top five countries where malware has infected IT resources include Bangladesh, Vietnam, Pakistan, Mongolia, and Georgia whereas the United States, Russia, the Netherlands, Germany, and France are the top five countries hosting online malware resources [Marinos, 2016]. Although malware has become a major cyber threat, the amount of malware appearing around the world is still increasing. According the ENISA 2015 Threat Report, malware continues to "increase by approximately one million samples every day and, by the publication of this report, this overall amount of malware would reach the 2 billion threshold". Techniques for infecting through malware are only becoming more sophisticated; the use of malware is even migrating to mobile devices such as Android and iOS. One such delivery mechanism is through web-based attacks.

Web based attacks utilize the web as a medium to first detect exploits and then install its payload. Some examples of web-based attacks range from malicious URLs or compromised webpages to browser exploits. Malicious URLs refer to URLs that have been compromised, allowing cyber adversaries to leave malware on the site or redirect end-users to download malware onto their devices. This tactic in combination with social media and phishing tactics has become an alternative to sending malicious programs by e-mail. In fact, Kaspersky Lab's web antivirus identified malicious URLs as the most active malicious object involved in online attacks, composing 37% of all attacks [Marinos, 2016]. A compromised webpage attack, also known as watering-hole attacks, is baited traps that target frequent visitors of the webpage. In 2014, the Computer Emergency Response Team in Australia handled 8,100 cases involving compromised websites [ACSC, 2016]. Although there are many other cyber threats in cyber space, the final types of cyber threats that this paper will focus upon involve an adversary gaining access to confidential information.

Continuing with the cyber threat taxonomy, cyber intrusions, cyber espionage, and data breaches all consist of similar end goals involving intelligence collection. However, there are slight distinctions between the terms. A *cyber intrusion* is an unauthorized access to a computer or system, leading to installation of malware, physical damage, or a data breach. *Data breaches/data losses*, a result of cyber intrusions, is the compromise of significant information such as trade secrets, business dealings, intellectual property, or any other data under the protection of an organization. Identity information, financial credentials, confidential data or intellectual property, and user credentials compose 90% of the types of stolen information; the loss of said information mainly affecting government, health, and technology sectors [Marinos, 2016]. The U.S. Office of Personal management data breach provides a fitting example of how cyber threat actors affected government; in this incident, cyber adversaries gained access to the personal information of several personnel applying for security clearances. *Cyber espionage* entails collecting information/data from a victim's computer network; however, the distinction from data breaches and cyber intrusions is that cyber espionage can be state-sponsored or motivated by industrial competition. There exists a broad range of other threat actors that may carry out cyber intrusions and data breaches such as lone cyber criminals, criminal organizations, and hacktivism groups. In contrast, state-sponsored threat agents have composed 87% of the cyber espionage incidents according the ENISA 2015 Threat Report. Some reported cases include the Sony and TV5Monde attacks, and the Bundestag hack. For most of these incidents, especially in the case of the Sony attack, adversaries were able to maintain a stable presence within the victim's system for a period of time and remain undetected while stealing confidential data. This pattern is characteristic of a growing security concern often referred to as an Advanced Persistent Threat.

2.4 Advanced Persistent Threats

Advanced Persistent Threats (APTs) are a growing security concern for many agencies/ organizations; they are sophisticated, multistage cyber-attacks with the goal of stealing confidential information. In general, threat actors typically target government or military agencies as well as businesses. As a result of the intended targets and the level of sophistication supporting the attacks, many cyber security researchers attribute them to nation-states. Regardless of adversary, the sophistication of the attack lies in an APT's attack stages in its lifecycle. These stages include choosing a victim, reconnaissance, delivery, exploitation, operation, data collection and exfiltration [Bere, 2015]. The first and second stages involve selecting a target and searching for an entry point into a system (usually an unsuspecting employee). Delivery is the stage in the attack where an APT can utilize several methods of penetration to gain access into the system; examples of methods include spear-phishing e-mails, memory-based attacks, and web-based click jacking. After entering the target's network, exploitation is the stage where an APT executes its payload, exploiting vulnerabilities identified in reconnaissance and establishing a secure connection with the system. Once established, an APT reaches the operation stage where it can persistently remain within the system possibly over a long period of time. The threat searches for sensitive information, traversing through the system and collecting data along the way during this stage. Finally, in the data exfiltration stage, an APT gathers the acquired data, encrypts it, and sends the packaged data over an encrypted channel to multiple servers controlled by the adversary [Bere, 2015]. An APT can repeat any part of the lifecycle over again and can reside within the system undetected, leaving the victim system at its mercy. There are two real-life APTs known as APT1 and APT29 that illustrate how effectiveness of an APT in cyber espionage.

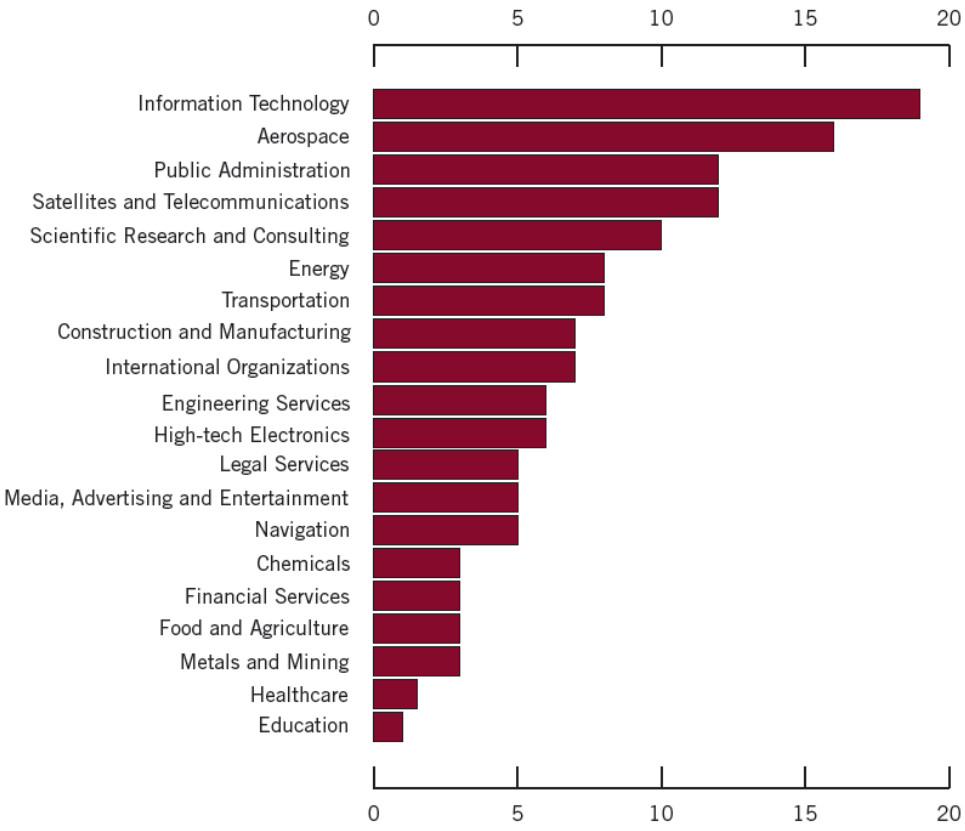


Figure 1: Industries Compromised by APT1 [Fireeye, 2013]

The first APT to discuss, codenamed APT1, has been one of the most persistent of China's cyber threat actors, carrying out an extensive cyber espionage campaign possibly with government-support. Officials at Mandiant, a FireEye company that helps organizations bolster their cyber defenses or detect cyber intrusions, wrote an intelligence report outlining the details of APT1. Although known for its long-running campaign, according to Mandiant officials, APT1 was a single organization of operators whose engagement in cyber espionage also led to the acquisition of large volumes of information. Since 2006, Mandiant has observed APT1 compromise "141 companies spanning 20 major industries; of these victims, 87% of them were headquartered in countries where English is the native language" [Fireeye, 2013]. Observing the industries targeted by APT1, the five highly-valued targets included information technology, aerospace, public administration, satellite and telecommunications, and scientific research and

consulting. The choice in targets was not the distinguishing aspect of APT1, but the amount of time APT1 had access to those systems. Generally, APT1 maintained its connection with the victim's network for an average of 356 days, whereas the longest time APT1 maintained access was four years and ten months [Fireeye, 2013]. Once APT1 infiltrated a network, the idea was to periodically revisit the victim's system over a period of weeks or years to avoid attracting attention when stealing confidential data; this stolen intellectual property ranged from blueprints and source code to business plans and contact information. Mandiant officials have observed APT1 stealing "6.5 terabytes of information from a single organization over a time period of ten months" [Fireeye, 2013]. Given that APT1 was able to steal this volume of data from a single organization and APT1 has compromised at least 141 companies, there is a likely possibility that this threat group has stolen hundreds of terabytes of information.

The next group is known in cyber threat literature as APT29, a Russian Cyber Threat group. Detected in early 2015, stealthy malware titled HAMMERTOSS was responsible for relaying commands and extracting data from compromised networks for APT29, using cloud storage services and Twitter as places to hide and store the data. This tool had the ability to retrieve commands via Twitter and GitHub, to automatically visit different Twitter handles on a daily basis, to communicate within specific time periods, to obtain commands via images containing encrypted data. This tool could also extract information and upload the data to a cloud [Fireeye, 2015]. To summarize the procedure, there were five stages of operation beginning from finding a Twitter handle to uploading stolen data. Using an algorithm, HAMMERTOSS generates and searches for a Twitter handle; an example handle for the day could be "123Bob123". If the handle is registered, the malware visits the Twitter page with that handle; otherwise, it waits another day and checks a different Twitter handle. The next stage involves

searching for a tweet with a URL and hashtag on the Twitter page that could possibly represent a location and file size. After this stage, the malware follows the URL to an image. This image uses *steganography*, the art of concealing a message in an image or document, to hide encrypted data. In the final stage, HAMMERTOSS decrypts the instructions from the image and executes them. Since the malware abilities range from reconnaissance to the transfer of data to a cloud storage service, these instructions may include any combination of the capabilities described previously [Fireeye, 2015]. APT29 was a capable cyber threat group and its effective tool only illustrates their ability to carry out cyber espionage campaigns while covering up any evidence of their presence.

3 The Problem

With the ever-present threat of adversaries compromising a U.S. government cyber system or network, there exist many countermeasures and means of protection such as firewalls, anti-virus software, and intrusion detection systems. However, such methods rely upon various indicators to assist in distinguishing malware and intrusions from normal user activity. To use an analogy, security personnel at a research facility can only allow trusted personnel to enter the building, but doing so without a list of employees and ID cards is nearly impossible; consider the identification and list of employees as proof of trusted personnel whereas individuals without these items to be intruders. Of course, creating finitely many rules that would always identify malicious activity is challenging, so modern systems continuously update their security based on known cyber threat trends. Thus, there is always a need to collect decision quality cyber threat intelligence in the form of cyber threat indicators.

As described in the section about Cyber Threat Intelligence, there are three levels of cyber intelligence. An indicator, precisely, is an inference mechanism based on historical data

and past occurrences that can be used to predict future occurrences within a similar setting. These indicators in the cyberspace context serve varying purposes at each level. At the Strategic level of cyber intelligence, cyber threat indicators include nation-state interest in innovative technology to obtain military advantage on a global scale. Operational cyber threat indicators, provide the understanding of adversary intent, capability, and likelihood of attack against fielded forces. Finally, Tactical cyber threat indicators find trends and patterns in the techniques, tactics, and procedures of employed by adversaries in order to predict malicious activity and help mission capabilities. Focusing on the tactical level, this problem posed in this paper revolves around predictive analytics and producing them through data analysis.

Residing within the realm of Big Data, cyber data or event log data can be complex and highly dimensional consisting of host names, ID numbers, command-line commands, and various other fields. This complexity and high-dimensionality introduces a slew of problems in addition to traditional roadblocks with large datasets. To identify a few, there are missing values within the data, fields that repeat information from other fields, and an inability to visually observe the data in a graph or chart. The cyber data discussed in Section 4.5 contains 231 fields/variables in its original form, but much of the data points were missing information and fields like host name and full domain query name represented similar information. Graphing or plotting 231-dimensional data in a way that humans can understand is still not possible.

Aside from complexity, there is also the problem of identifying relationships within the data. Principle Component Analysis (PCA), multidimensional scaling (MDS), clustering methods are well-known for their applications in data science and machine learning. PCA and MDS “attempt to find non-linear maps to Euclidean space which preserve the distance functions on the data set”, producing useful two or three dimensional versions of the data if successful

[Singh, 2007]. As a result, these methods do handle the complexity problem of the data. Concerning clustering methods, they produce distinct, unrelated groupings of the data; alternatively, clustering methods creates groups of points that are closer to each other in terms of some distance function. The drawback with these methods is these methodologies sometimes obscure geometric features. For example, a breast cancer study exploring a new data analysis method was able to identify a subgroup of breast cancer tumors that was characterized by high survivability in their patients. Researchers in the study tried validating this new subgroup using cluster analysis and discovered that clustering overlooked this subgroup of tumors because it had scattered points from the subgroups across many clusters [Nicolau, 2011]. One method that strives to solve these data complexity problem and sensitivity to large and small scale patterns is known as Topological Data Analysis.

Topological Data Analysis gives rise to methods that extract and highlight topological features in order to summarize the data. The idea is that data possesses shape that may convey meaningful information. Data analysts or even curious individuals naturally attempt to look for this shape in order to gain insight, but the complexity of the data prevents them from using basic methods like graphing in two or three dimensions. With its origins from combinatorics, discrete geometry, linear algebra, and topology, TDA methods utilize three invariants studied in topology in order to capture the shape of the data [Chintakunta, 2016]. The following section discusses these three invariants in detail. In addition, TDA methods derive relationships within the data by preserving a notion of nearness but distorting large scale distances; small distances can signify similarity whereas large distances possess little meaning [Singh, 2007]. One TDA method highlighted within this paper is called Mapper; Mapper essentially reduces high dimensional data sets into simplicial complexes or discrete combinatorial objects that globally represents the data.

The Python Mapper is an open-source implementation of the Mapper method that is limited in its capabilities but provides enough features as described in the method. Although a standalone Python application, the Python Mapper does not provide the capability to pull data or to analyze the data beyond visualization. This paper proposes designing and implementing a system to integrate these features into the Python Mapper as well as applying the Python Mapper to cyber data.

Along with the completion of this report, there are many criteria for the success of this project. One criterion is the completion of a TDA system that pulls information from a cloud storage service, cleans and parses the data, applies the Mapper method to the data, and allows for further analysis of the Mapper output. Another goal for the project is to develop cyber analytics or rules that help existing cyber detection systems to identify malicious activity. Finally, the last criterion for success is the continuation of the project with an emphasis of TDA being applied to machine learning and the additional research of persistent homology.

4 Mathematics behind TDA

Topology is a branch of mathematics whose origin dates back to the time of Leonhard Euler, an 18^{th} century Swiss mathematician well-known for many influential discoveries and contributions to mathematics. Topology is the mathematical study of properties unaltered by continuous deformations such as stretching or twisting on objects in space. To elaborate further, this notion of preservation after some deformation translates to a notion of equivalence. An example is the deformation of a donut to a coffee cup; both are topologically equivalent to each other. Until the development of topological data analysis, concepts and results produced in topology have resided solely in theory and pure mathematics. Thus, before introducing this new method of data analysis, the following sections focus on the supporting mathematics. Many of

these concepts and definitions can be found in references such as Morris's *Topology without Tears*, Zomorodin's *Topology for Computing*, and from lectures taught by Norman Wildberger and T. Dey.

4.1 Topological Spaces

Beginning with some basic definitions, a set is a collection of objects that can be numbers or other sets; in this case, the objects are points. A metric computes the distance between two points in the set. A space is a set of points and any space with a metric is known as a metric space; the most familiar metric space is Euclidean space.

Definition 4.1.1 A set X with a metric function d is called *metric space*. The metric or distance function $d: X \times X \rightarrow R$ is a function satisfying the following axioms:

i. For all x, y in X, $d(x, y) \geq 0$
ii. If $d(x, y) = 0$, then $x = y$
iii. For all x, y in X, $d(x, y) = d(y, x)$
iv. For all x, y, z in X, $d(x, y) + d(y, z) \geq d(x, z)$

In a more general manner, metrics for any topological space reflect the notion of similarity or closeness. Since the points in a topological space are distinct and abstract entities, small or large distances between two points may signify similarity or dissimilarity, respectively. Coordinate invariance, a term defined later on, allows one to study the connectedness of a point set in a more general manner than with metric spaces that assigns distances based on coordinates. A topological space consists of the point set and a set of all subsets from this point set, but the formal definition is as follows:

Definition 4.1.2 Let X be a non-empty set. A set τ of subsets of X is said to be a *topology* on X if:

i. X and the empty set belong to τ.
ii. The union of any (finite or infinite) number of sets in τ belongs to τ.
iii. The intersection of any two sets in τ belongs to τ.

Definition 4.1.3 The pair (X, τ) is a *topological space*, where X and τ are defined as before.

A fundamental question in topology is whether two topological spaces are equivalent. To do so requires that any topological properties in one topological space map to the other space. Elaborating on mapping, recall first that a function $f: R \to R$ is said to be continuous if for $a \in R$ and each positive real number ε, there exists a positive real number δ such that $|x - a| < \delta$ implies $|f(x) - f(a)| < \varepsilon$. Alternatively, let X and Y be two topological spaces, a function that maps X to Y is continuous if the inverse image of an arbitrary open set in Y is open in X [Kraft, 2016]. These continuous maps between topological spaces preserve key properties of space and help exhibit the notion of topological equivalence. One fundamental notion of equivalence in topology is a *homeomorphism*.

> **Definition 4.1.4** Let X and Y be two topological spaces. A *homeomorphism* f: $X \to Y$ is a bijective function, such that both f and the inverse of f are continuous. We say that X is *homeomorphic* to Y, $X \approx Y$, and that X and Y are topologically equivalent.

Referring back to the example of the donut (torus) and coffee cup, both objects are topologically equivalent, or more formally homeomorphic. One can apply continuous transformations on the donut to produce the cup and vice versa. Another more abstract example concerning homeomorphism pertains to the interval [0, 1] and the unit circle [Wildberger, 2012]. To intuitively show that these spaces are not homeomorphic to each other, imagine removing a point from the circle and removing a point from the interval. While the circle remains a singly connected component, the interval splits into two components. If the circle and closed interval were homeomorphic, they would share the same genus; the genus is the maximum number of cuttings on the surface without disconnecting it. Another notion of equivalence studied in topology is known as homotopy equivalence. Homotopy preserves some aspects of connectedness such as the number of connected components and the number of holes in space. Its definition and its requirements for equivalences are as follows:

Definition 4.1.5 Let g: $X \to U$ and h: $X \to U$ be maps. A *homotopy* is a map H: $X \times [0,1] \to U$ such that H (*, 0) = g and H (*, 1) = h. Two maps are *homotopic* if there is a homotopy connecting them.

Definition 4.1.6 Two topological space T and U are *homotopy equivalent* if there exists maps g: $T \to U$ and h: $U \to T$ such that $h \circ g$ is homotopic to the identity map i_T: $T \to T$ and $g \circ h$ is homotopic to the identity map i_U: $U \to U$

Alternatively, two spaces are considered homotopy equivalent if one space can be continuously deformed into the other space. Consider a loop in some space; shrinking the loop eventually turns the loop into a single point, so this loop would be homotopy equivalent to a point. These concepts of homeomorphisms and homotopic equivalence become involved in the discrete constructs and mathematical definitions supporting the Mapper output. In fact, Gurjeet Singh noted in his dissertation that these Mapper constructions were similar to Reeb graphs [Singh, 2007]. The following definitions help describe these constructs.

Definition 4.1.7 Let X and Z be topological spaces and let f: $X \to Z$ be a continuous map. Let $U = \{U_\alpha\}_{\alpha \in A}$ be a finite open covering of Z. Consider the cover $f^*(U) = \{f^{-1}(U_\alpha)\}_{\alpha \in A}$ of X, typically called the pull-back of U along f. The mapper construction resulting from the given dataset (X) is known as the nerve complex of the pull-back cover $M(U, f) := N(f^*(U))$ [Kim, 2016].

Definition 4.1.8 Define a covering (projection) as a surjective open map from two topological spaces that is locally a homeomorphism.

Definition 4.1.9 Given a finite covering $U = \{U_\alpha\}_{\alpha \in A}$ of a space X, define the *nerve* of the covering U to be the simplicial complex N(U) whose vertex set is the indexing set A, and where a family {a0,a1, . . . ,ak} spans a k-simplex in N(U) if and only if $U_{\alpha 0} \cap U_{\alpha 1} \cap \ldots \cap U_{\alpha k} = \emptyset$ [Singh, 2007].

To utilize Mapper, the algorithm requires a metric space X (data), a filter function mapping X to Z, and a covering for Z as input. The following sections should give an overview of how the Mapper is able to summarize the shape of the data with a high degree of accuracy given these inputs.

4.2 Simplicial Complexes

To begin this section, suppose there is a set X composed of n points $x_1, \ldots, x_n$ that belong to R^d where d is the dimension for *R*. An affine combination can be any summation of these points in X, written as $\sum_{i=1}^{n} \mu_i x_i$, such that the summation of all coefficients μ_i totals to 1. If all the coefficients μ_i sum to 1 and each μ_i is nonnegative, then x = $\sum_{i=1}^{n} \mu_i x_i$ is referred to as a convex combination of the μ_i. Consequently, the convex hull is the set of all possible convex combination of these coefficients. Keeping these definitions in mind, simplexes are as follows:

> **Definition 4.2.1** A *k-simplex* Ω is the convex hull of a set p of k+1 affinely independent points. A k-simplex has dimension k.

Since the points belong to R^n, the k-simplex can be defined in terms of k+1 linearly independent points. Recall that a set S is linearly or *affinely independent* if no point in S is a linear (affine) combination of any other point in S. As a result, the two terms for independence are interchangeable. Now, some familiar simplexes include a point (0-simplex), a line segment (1-simplex), a triangle (2-simplex), and a tetrahedron (3-simplex). Before moving onto simplicial complexes, given a k-simplex defined by a point set S, know that a face is a simplex defined by a nonempty subset of S.

> **Definition 4.2.2** A *simplicial complex* K is a set containing finitely many simplexes that satisfy the following two restrictions:
>
> i. K contains every face of every simplex in K
> ii. For any two simplices σ, τ belonging to K, their intersection is either empty or a face of both σ and τ

Simplicial complexes are combinatorial objects that represent spaces, encoding locations of points relative to nearby points and representing the connectedness of the space. These aspects are helpful in representing high-dimensional objects or highlighting specific topological properties in surfaces. To do so requires creating triangulations of the surface or high-dimensional object.

> **Definition 4.2.3** A *triangulation* of topological space X is a simplicial complex K together with a homeomorphism between X and |K|, where |K| is the *underlying space* of K and $|K| = \bigcup_{\sigma \epsilon K} \sigma$. Note that |K| is itself a topological space.

Take, for example, the surface of a sphere; the surface of a tetrahedron or 3-simplex is a triangulation of a sphere because they share similar properties like number of connected components and number of voids. Representing a sphere with a tetrahedron requires less resources and memory for a computer to process and display. This idea of constructing triangulations is evident in the Mapper method discussed in Section 5.2. With Mapper, triangulations create topological summaries of the complex data.

4.3 Simplicial Homology and Betti Numbers

Simplicial complexes can approximate all topological spaces. Building upon this concept, simplicial homology utilizes finitely generated Abelian groups to measure the number of n-dimensional holes in a topological space. Algebraic topology is a branch of topology that provides simplicial homology the ability to construct algebraic tools called homology groups to quantify topological features in simplicial complexes [Dey, 2013]. Before formally defining homology groups for simplicial complexes, the following definitions introduce chains and boundaries. Consider a set X and a simplicial complex K on X. A p-chain c for K is a formal sum of p-simplices added with coefficients, that is, $c = \sum_{i=1}^{k} \alpha_i \sigma_i$ where σ_i are the p-simplices and α_i are the coefficients. P-chains can form a group under addition.

> **Definition 4.3.1** The k^{th} *chain group* of a simplicial complex K is $\langle C_k(K), + \rangle$, the free Abelian group on the oriented k-simplices, where the identity is the chain $0 = \sum_{i=1}^{k} 0\sigma_i$ and the inverse of a chain c is c itself since c + c = 0.

Figure 3 will help illustrate concepts for previous and consequent definitions. In this oriented graph, the 0-simplices (vertices) are X, Y, and Z; the 1-simplices (edges) are a, b, c, and d. The closed walk [abc] starting from X can be written as an addition of edges such as a + b + c. Since commutativity applies to this naming scheme, a + b + c is the same as b + c + a and c + a + b. In

this case, note that the closed walk a + b + c = (a + b + d) + (c - d) where (a + b + d) and (c - d) are the two smaller closed walks in the graph. Chain groups for this graph include C_0 (the free abelian group on vertices in G) and C_1 (free abelian group on directed edges). C_0 contain 0-dimensional chains that are integral linear combinations of the vertices (e.g. x – 6y + 8z). C_1 consists of 1-dimensional chains that are integral linear combinations of directed edges such as 4a – 5b + 3c + 4d.

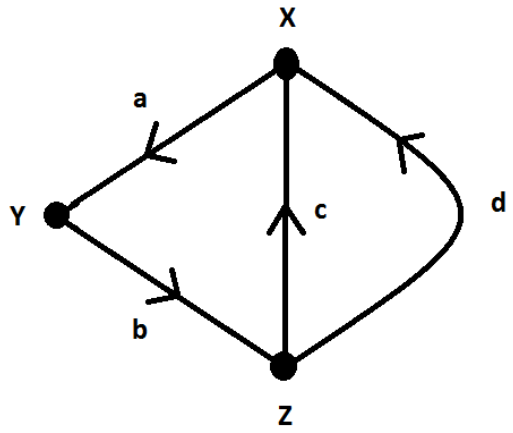


Figure 2: Graph G Illustrating Chain groups [Wildberger, 2012]

With the chain group defined as $C_k(K)$, choose an ordering on X and define the following sequence of homomorphism:

$$\cdots \xrightarrow{\partial_{n+1}} C_n(K) \xrightarrow{\partial_n} C_{n-1}(K) \to \cdots \to C_1(K) \xrightarrow{\partial_1} C_0(K) \to 0$$

Figure 3: Sequence of homomorphisms [Dey, 2013]

Definition 4.3.2 If $\sigma = [v_0, \ldots, v_p]$ is a p-simplex, its *boundary* is $\delta_p\sigma = \sum_{i=0}^{p}\{v_0, \ldots, \widehat{v_i}, \ldots, v_p\}$ where $\widehat{v_i}$ indicates that the vertex v_i omitted.

The boundary operator, denoted δ_p, defines a mapping between chain groups found in different dimensions. Specifically, given a p-simplex, the boundary operator returns the (p-1)-chain of its boundary (p-1)-simplices. An example would be that, given a tetrahedron, the boundary operator

would return the four triangle faces of the tetrahedron. For the figure above, the boundary operator describes the relationship between edges and vertices; the boundary of a directed edge in Graph G is written destination vertex – starting vertex. The boundary of directed edge a is y – x and the boundary of edge b is z - y, or alternatively $\delta(a) = y - x$ and $\delta(b) = z - y$. In addition to boundaries and chains, a p-cycle c is a p-chain such that $\delta c = 0$. To see that the boundary operator returns 0 for cycles/loops, applying the operator boundary to a + b + c for the graph in Figure 2 results in $\delta(a + b + c) = \delta(a) + \delta(b) + \delta(c) = 0$. Similar to chains and chain groups, p-cycles can form a p-cycle group.

> **Definition 4.3.3** The *k^{th} cycle group* is Z_k = Kernel (δ_k). A chain that is an element of Z_k is a k-cycle. The *k^{th} boundary group* is B_k = Image (δ_{k+1}). A chain that is an element of B_k is a k-boundary.

Define the Kernel of the boundary operator to be the elements of the k^{th} cycle group that map to the identity element in the $(k-1)^{th}$ cycle group. Additionally, define the Image (δ_{k+1}) to be set of all elements in the $(k-1)^{th}$ cycle group that elements in k^{th} cycle group can map to under the boundary operation. From here, a formal definition of a homology group follows from the previous definitions.

> **Definition 4.3.4** The *k^{th} homology group* is the quotient Kernel (δ_k) / Image (δ_{k+1}), denoted $H_k = Z_k / B_k$.

Related to homology groups, Betti numbers measure the number of connected components, cycle, voids, and their higher dimensional analogues [Kraft, 2016]. They essentially summarize homology groups by encapsulating properties of topological spaces as numerical values; in particular, Betti numbers refer to the number of n-dimensional holes on a topological surface or simplicial complex. More formally, the definition of a Betti number is as follows:

Definition 4.3.5 The *k^{th} Betti number*, denoted β_k, of a simplicial complex K is $\beta(H_k)$, the rank of the free part of H_k. β_k = rank (H_k) = rank (Z_k) – rank (B_k) where Z_k and B_k have been defined previously.

The next section builds upon all the definitions discussed thus far and introduces a method for finding features in a topological space called persistent homology.

4.4 Persistent Homology

Creating triangulations of point cloud can summarize any topological features showcased in the data, but another approach known as Persistent homology goes a step further. The key idea in Persistent Homology is discovering the extent in which persistent (actual) topological features remain after introducing noise to the data. Observe in Figure 4 that the data set is a curve in R^2 with two loops. First, consider a noisy point cloud sampled from the curve; sampling points means selecting points with a probability distribution concentrated near the shape of the curve [Carlsson, 2009]. Let P denote the set of sampled data points. As illustrated in the figure below, the approach entails creating a complex from the union of balls of size $\varepsilon > 0$ around each data point in P, calculating the homology of the resulting construct, and observing which features persist while modifying ε. Figure 4 shows that increasing the radii (ε) of the balls centered on each data point eventually creates the two loops. When the size of ε is large enough, the two loops disappear in Figure 4D. These two loops must be significant topological features because they were able to persist longer than any other feature that might have existed. In summary, this example exhibits how persistent homology gives an overview of the homological information for all the different values of ε.

From point cloud to complex, the process is essentially using the data points from the point cloud as vertices to output a combinatorial graph whose edges are based on proximity of data points. Modeling a point cloud data *X* by a simplicial complex serves to assist in computing its

topological invariants. In computational topology, there are various ways to convert X into a complex; the following definitions describe two such methods.

> **Definition 4.3.6** Let ε > 0 be a real number. Define the *Vietoris-Rips Complex*, denoted $V(X, \varepsilon)$, to be a simplicial complex on the set X consisting of these subsets $\sigma \subset X$ where d (x, y) < ε for any x and y in σ. Note that for $\varepsilon \leq \varepsilon'$ for some nonnegative real number ε', $V(X, \varepsilon) \subset V(X, \varepsilon')$.
> **Definition 4.3.7** Let ε > 0 be a real number. Define the *Cech Complex*, denoted $C(X, \varepsilon)$, to be a simplicial complex on the set X consisting of these subsets $\sigma \subset X$ for which there is a $y \in X$ such that $d(x, y) < \varepsilon$ for any $x \in \sigma$. Note that for $\varepsilon \leq \varepsilon'$ for some nonnegative real number ε', $C(X, \varepsilon) \subset C(X, \varepsilon')$.

Using either method to construct complexes from a point cloud, the result is a nested sequence of complexes that record the changes in homology as ε increases. This nested sequence is known as a *filtration.*

> **Definition 4.3.8** Given a simplicial complex K, a *filtration* is a nested subsequence of complexes K^i of K, indexed by the nonnegative integers, such that if i ≤ j then $K^i \subseteq K^j$. The total ordering itself is called a *filter.*

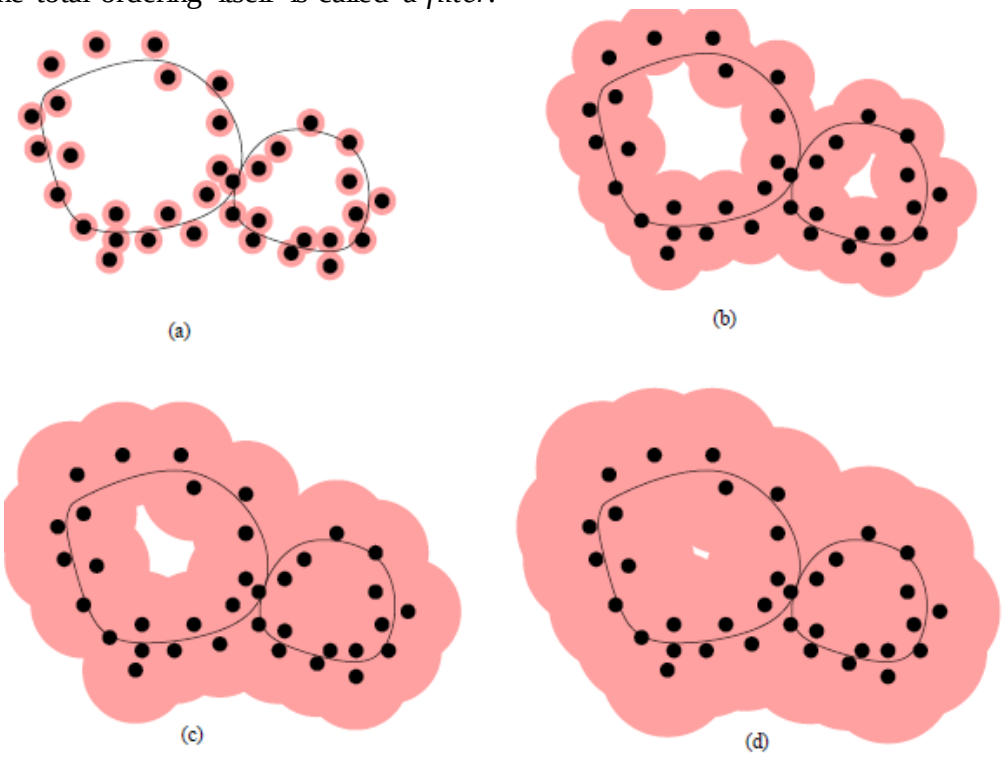


Figure 4: Illustration of Persistent Homology [Dey, 2013]

> **Definition 4.3** In relation to persistence, the *p-th persistent homology groups* are the Images of homomorphism, $H_p^{ij} = im\, f_p^{ij}$ for $1 \leq i \leq j \leq n$.

Given a filtration, the i-th complex K^i has associated boundary operators δ_k^i and groups such as $C_k^i,\ B_k^i, Z_k^i, and\ H_k^i$ for all $i, k \geq 0$. Note that the p-th persistent homology groups describe when a class is born and when it dies.

> **Definition 4.3** (Birth and Death) Consider a simplicial complex K and a function $f: K \to R$ on it (note f^{-1}on some interval between $-\infty$ and $a \in R$ gives a filtration on K). A p-th homology class [c] is *born* at a simplicial complex K_i if $[c] \in H_p(K_i)$, but $[c] \notin H_p(K_{i-1})$. The p-th homology class *dies* entering K_j if it merges with a class that is born earlier. Formally, $f^{i,j-1}([c]) \notin H_p^{i-1,j-1}$, but $f^{i,j}([c]) \in H_p^{i-1,j}$.

Persistent diagrams are visual representations of the persistent homology that exhibit the evolution of topological features within the filtration. They are defined as "extended planes $(\mathbb{R} \cup \{\pm\infty\})^2$ on which a point with two coordinates represents a birth paired with a death" [Dey, 2013]. In other words, the first coordinate corresponds to the starting index when the feature was created and the second point represents when it dies (persistence). Key invariants in persistence diagrams called barcodes "measure the significance of a feature based on its lifespan through a filtration" [Kraft, 2016]. The bars in a barcode record when a feature lives or dies at some particular value for ε; longer bars mean that a feature within a complex is persistent and gives meaning to statistical significance for a component of the complex such as a loop or void. On the other hand, smaller bars correspond to either noise within the data or noise introduced through sampling.

Gunnar Carlsson's *Topology and Data, The Basic Theory of Persistent Homology* by K. Wang, and lecture notes written by Tamal Dey provide a rigorous explanation of persistent homology using an algebraic construction. In terms of application of persistent homology, Fintan Hegarty's doctoral dissertation, *Computation Homology of Cubical and Permutahedral*

Complexes, outlines various applications in topology estimation (finding topological features from a sampling of the original data set), image segmentation (changing an image's representation for feature recognition), and in medicine [Hegarty, 2012]. The next chapter discusses research and advances in topological data analysis, focusing especially on Mapper and a company called Ayasdi that combined machine learning with an improved version of Mapper.

5. Overview of TDA

In terms of major difficulties with data, there are many challenges involved with Big Data. Advances in data management have allowed professionals to tackle the sheer volume of information associated with Big Data. However, the other major problem that this paper focuses upon is the high dimensionality and complexity of data. With the introduction of topological data analysis (TDA), research has leaned towards capturing the underlying shape of the data to uncover any meaningful phenomena hidden within the data. To be more precise, a given dataset creates a point cloud of data in possibly high dimension; within topology, the resulting shape refers to the features within the data that do not change under continuous deformations like shrinking or stretching [Chintakunta, 2016].

Before discussing topological data analysis in general, the first section describes three invariants in topology that enable methods in topological data analysis to accurately and robustly obtain shapes of data. Software packages, such as the Python Mapper or JavaPlex, have taken advantage of these properties to create visualizations of high dimensional data. Utilizing the available TDA software, there have been several applications within medical research as well as in the social sciences; concerning medical research, TDA has assisted in identifying subgroups within breast cancer patients and diabetes patients. This application along with other examples appear in Section 5.4. Building upon these foundations, persistent homology and barcodes of

data have led to further advances in TDA as well as more robust algorithms for pattern recognition. Many publications like Robert Ghrist's *Barcodes* and Gunnar Carlsson's *Topology and Data* have a focus on persistent homology and its applications.

5.1 Topological Invariants

Stemming from ideas and concepts since the time of Leonhard Euler, topology has been an abstract field of mathematics that has solely been used to study abstractly defined shapes and surfaces [Lum, 2013]. With the introduction of topological data analysis, data scientists and researchers saw the potential of adapting topological concepts to large and complex data sets. The reason for its success with high dimensional data sets is due to three topological properties known as *coordinate invariance, deformation invariance,* and *compressed representation*. These properties give topology its ability to study shape in a robust manner. Coordinate invariance refers to how topological objects are independent of any specific representation in a coordinate system. Instead, the pairwise distances between points of the object are significant [Carlsson, 2014]. In other words, these properties of shape remain unaltered after rotating the object or changing the coordinate system of the object, allowing for comparisons between objects residing in different dimensions. Deformation invariance refers to how topological properties of shape also remain unchanged if one stretches or shrinks the object in question without any tearing or gluing. For example, consider any letter of the English alphabet; most people can recognize this letter even in a different font or size. In topology, an ellipse and a circle in the 2-dimensional plane are topologically equivalent because they share similar characteristics like number of holes/loops. Finally, compressed representation refers to building simpler representations of shapes, capturing the topological features of the shape and losing the finer details [Carlsson, 2014]. Considering a circle as an example, one topological feature defining the shape of a circle is its loop. Thus, a hexagon with the similar shape and loop can represent a circle; this

representation is more preferable compared to the infinite number of points in a circle because a hexagon consists of only six nodes and six edges [Lum, 2013]. The curvature of the circle is lost in the compression, but the representation preserves the major topological feature of a loop. Applications of topological data analysis have been wide-reaching, ranging from applications in medicine to business data analytics. However, in this paper, the case studies utilizing topological data analysis are going to focus on one method known as Mapper and its implementation called Python Mapper.

5.2 Mapper

First introduced by Gurjeet Singh, Gunnar Carlsson, and Facundo Memoli, the Mapper algorithm is a method for topological data analysis, emphasizing the goal of giving shape to data and interpreting this shape [Singh, 2007]. Complex data can incorporate several attributes or features; for example, patient medical data consists of age, gender, blood pressure, etc. Singh states in a paper dedicated to introducing this method, "this new method for qualitative analysis, simplification, and visualization of high dimensional sets possesses the ability to reduce high dimensional data into simplicial complexes in order to capture topological and geometric features at a specified resolution" [Singh, 2007]. This method builds a compressed representation of data to highlight its significant features. The Mapper algorithm achieves this result by using topological ideas that preserve closeness or similarity but alter distances of large magnitude. This property fits into data analysis of high throughput biological data especially well, as noted in the application of Mapper sections.

Moving to the algorithm itself, it consists of multiple steps involving a distance/similarity metric, a filter function, a clustering algorithm, and the output visualization. Section 5.2.2 provides an explanation of metrics and the criteria used for selecting one. Three assumptions of the algorithm are: only numerical values compose the given data set X, the user has chosen an

appropriate distance or similarity metric, and "computation of inter-point distance between points or even sets of points from the data is possible" [Singh, 2007].

If the data set contains non-numeric strings, the user must transform these attributes in the data to numeric data by either encoding the information or dropping them from the data altogether. A filter function $f: X \rightarrow R$ refers to a real-valued function that maps data points in X to the real number line. Selecting this filter function is as significant as choosing a distance metric because this function determines the topological features to highlight; the user may select or integrate functions that reflect certain properties of the data. Keeping these instructions and advice in mind, the first step in the algorithm is finding the range of function restricted to the data points. The accompanying step is partitioning this range into smaller, overlapping intervals in order to create a covering space of the data points. Percentage of overlap and number of interval are both parameters that the user can tune or change. To illustrate this step, consider a closed interval [0, 2] partitioned into 3 smaller intervals with 50% overlap; the resulting intervals are [0, 1], [0.5, 1.5], [1, 2]. Now, within each interval, the algorithm forms sets of data points such that from the data set such that the function value of the data point lies within that interval and clusters the points based on distance [Singh, 2007]. More formally, the algorithm forms sets $X_j = \{ x \in X \mid f_j(x) \in I_j \}$ where j is an index into the set of intervals I. Although good clustering is also important to consider when visualizing the data, this algorithm does not place any restrictions on the clustering algorithm involved, so it promotes the use of user-defined and domain specific clustering algorithms. The authors introducing the Mapper algorithm have based their clustering algorithm on single-linkage clustering; this algorithm "returns a vector in R^{N-1} that holds the length of an added edge to reduce the number of clusters by one at each step of the algorithm" [Singh, 2007]. Once clustering is complete, the algorithm outputs a representative

simplicial complex of the data; the Python Mapper goes further to display this representation as a 2-dimensional network of nodes. All edges between nodes/clusters in the graph indicate that their intersection is non-empty. This visualization additional has a coloring and labeling of each node; colors represent the average function value of the node and labels indicate the number of data points residing within a node.

5.2.1 Python Mapper

The Python Mapper is an implementation of the Mapper algorithm written with Python and C++ bindings [Mullner, 2013]. In addition to implementing the algorithm, the software also consists of a graphical user interface that enables beginners to quickly acquaint themselves with its many features such as filter selection and node coloring options. Developers of the toolchain, David Mullner and Aravindakshan Babu, have provided this complete and extensible toolchain to the scientific community as open-source software in hopes of other researchers "extending, modifying, and applying the software for new ideas" [Mullner, 2013]. As of 2016, there have been numerous uses of Python Mapper on breast cancer patient data as well as employee information. Before discussing these applications in more detail, this section begins by elaborating on the Mapper algorithm further.

5.2.2 Metrics

Since clustering contributes to the algorithm, there must be a measure of distance and/or similarity to determine the closeness of any two data points. The formal definition of similarity is, "the measure of how close to each other two instance are; the closer two instances are in comparison, the larger the similarity value". In contrast, the dissimilarity measure determines how different two instances are. A distance metric is a measure of dissimilarity that obeys the conditions outlined in Definition above.

Utilizing one of these metrics to determine similarity/dissimilarity requires first that the fields within the data align with the criteria of that metric; matching the criteria allows for an accurate measure of closeness between two data points. To elaborate on this point, consider that attributes within the dataset can vary in type; attributes of the data can be nominal, cardinal, or ordinal. Cardinal or numerical data have a continuous scale of values like sale profits, time, or height, whereas ordinal data infers an order on the values and nominal or categorical data are discrete values. Examples of nominal data include gender, favorite color, and names. Comparing two data points composed of only nominal attributes with some real-valued metric like Euclidean distance would not provide any information on closeness; one cannot compare two names using Euclidean distance. However, the alternative is to use Hamming distance to determine the number of different/similar characters present in the two names. The Python Mapper implementation provides three distance metrics such as Euclidean, Minkowski, and Chebyshev.

Beginning with the Euclidean metric, Euclidean distance is the most well-known measurement for distance. Given any two points U, V in a two-dimensional space, the square root of summing $(U_x - V_x)^2$ and $(U_y - V_y)^2$ gives the distance between those two points. A generalization of this formula to an N-dimensional space is a summation of the squared differences between corresponding components of any two N-dimensional vectors. The Minkowski distance, on the other hand, is a generalized metric to measure the differences between absolute magnitude of differences between a pair of objects. Chebyshev Distance, also known as maximum value distance, "measures the absolute magnitude of difference between coordinates of a pair of objects" [Tabish, 2009]. The formulas for these metrics can be seen in Table 1.

Table 1: Distance functions

Distance function	$d(x, y)$
Manhattan	$\sum_{w \in L} \lvert\phi_w(x) - \phi_w(y)\rvert$
Canberra	$\sum_{w \in L} \frac{\lvert\phi_w(x) - \phi_w(y)\rvert}{\phi_w(x) + \phi_w(y)}$
Minkowski	$\sqrt[k]{\sum_{w \in L} \lvert\phi_w(x) - \phi_w(y)\rvert^k}$
Chebyshev	$\max_{w \in L} \lvert\phi_w(x) - \phi_w(y)\rvert$

Utilizing the filter function in the next step of the algorithm, these distance metrics allows this method to project data points onto R^1; selecting the filter function is as important as selecting the metric because any interpretations of the output depends upon the choice of filter function.

5.2.3 Filter Functions

Crucial steps in the algorithm involve using a function to map data points to values in R^1, partitioning the range of values restricted to this set of data, and devising a cover for this range. One important point to note is that this function relies on the ability to compute distances between points; again, the metric determines how this distance is computed. Focusing further on the filter function, the purpose of this function is to summarize all the relevant features of the input data. The outcome, as a result, becomes heavily dependent upon the filter function's method of mapping the data points [Singh, 2007]. To illustrate this concept, consider the geometrical point cloud data and the two filter functions in Figure 5. Filter function f computes the height of a point from the input data, whereas filter function g summarizes the input data from a horizontal perspective. To understand the output, imagine first a horizontal plane moving from the top of the Y-shaped point cloud to the bottom. The intersections between the plane and point cloud result in the Y shape to the right of the point cloud. For the filter function g, consider a vertical plane that moves horizontally from left to right of the data. Beginning from the left, there is one connected component that results in one point of the corresponding loop in the

output. Continuing, the intersections become two lines on the plane, resulting in the top and bottom curves of the left loop in the output. The intersection between the middle section of point cloud and the plane is a connected component that allows the two curves to meet and create one loop. Symmetry of the point cloud indicates how the remaining section creates the right loop.

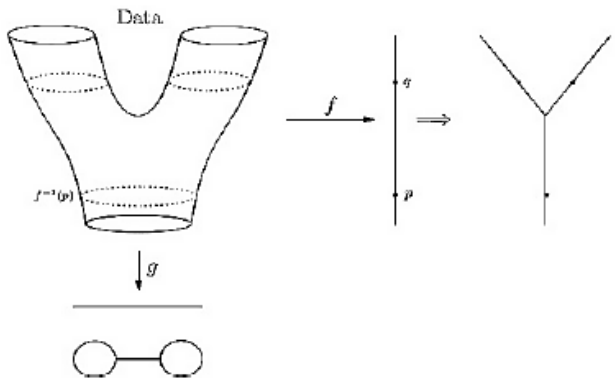


Figure 5: Illustration of Filter Function Representing Input Data [Kim, 2015]

Figure 6 illustrates the same concept but using an H-shaped point cloud.

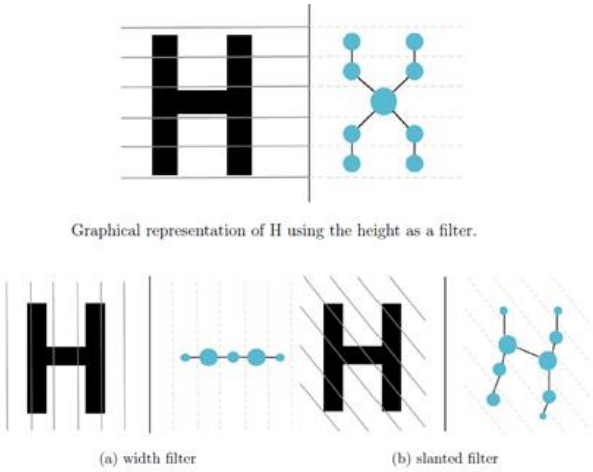


Figure 6: Another Example of Filter Functions [Bayless, 2015]

Provided in the Python Mapper, there are seven filter functions that represent or highlight different properties of the data. The first set of functions aim, "to understand a mathematical context surrounding the data points in the space" [Kim, 2015]. These functions, eccentricity and Gaussian density, highlight the geometry of the data set. For Gaussian density, the mapper

implementation utilizes a kernel density estimator with a multivariate, radially symmetric Gaussian kernel. Gaussian density assumes that all values within a data points are Euclidean data in order to compute the distribution of the data. Looking at the corresponding formula in Table 2,

Goal	Suitable Type	Filter	Description	Note
to understand a mathematical context surrounding the data points in the space	geometric filter	Gaussian density $\sum_Y e^{-\frac{d^2(X,Y)}{2h}}$	considers each row as Euclidean data to estimate the density	- uses the selected metric - estimate density is a fundamental task in statistics
		L1 centrality: $\sum_Y d(X,Y)$	measures row centrality relative to others	- measures how central or peripheral data is
		L-infinity centrality $\max_Y d(X,Y)$	associates a maximum distance from **X** to others	
to magnify difference between groups	projection filter	PCA COORD 1&2	generate PCA coordinates with the highest and the second-highest variance component	- assumes data is using the Euclidean metric - use both together for XY coordinates of 2-d embedding
		Neighborhood 1&2	generate a two-dimensional embedding of the k-nearest neighbor graph by connecting point to its nearest neighbors	- uses the selected metric - emphasize the metric structure of data - do not equalize filter function

Table 2: Available Filter Functions

epsilon controls the smoothness of the density estimate on the data set. Moving to eccentricity, the intuition is that this function determines which points in the data are different from the center of the data without designating a center point(s). Understanding how the functions compute distances, however, is meaningless if one cannot interpret the shape of the resulting mapper output.

5.2.4 Interpretation of Visualizations

Each filter function provides different meanings and interpretations of the data, so this section gives a brief summary of them. A possible interpretation that may originate from using the eccentricity filter function would be that low-valued nodes are closer to the center of the data and any high-valued nodes branching from this center infers anomalous behavior. In terms of the Gaussian density estimator, the resulting output should infer the distribution of distance values. For example, looking at Figure 5, one can infer that the data set is bi-modal; such an inference would indicate that the data points near the top of the network in Figure 5 showcase more unique

traits than the rest of the data. Although these functions are a subset of the available filters in the Python Mapper, this section reinforces the idea that the choice in filter function is essential in producing the desired output.

5.3 Ayasdi

Perhaps the most prominent company at the forefront of topological data analysis research and application, Ayasdi is an advanced analytics company that has taken Mapper beyond simply visualizing data and has created a new product to help their clients solve complex data analytics challenges. Ayasdi's product incorporates "the latest machine learning, statistical and geometric algorithms with topological data analysis and big data infrastructure" [Ayasdi, 2016]. The aspects concerning topological data analysis constructs compressed representations of the data, from which data scientists and analysts search for variables that impact the patterns found in the representation. This process, consequently, results in features holding predictive power that lay the foundation for machine learning algorithms.

5.4 Applications of Topological Data Analysis

In this section, the main topic of discussion is applying the Mapper algorithm to identify shape and to solve real problems. The first section describes the algorithm's ability to capture the shape of a hand and other topological objects from sampled points. Afterwards, the second section discusses Mapper's application within medicine and business. One application comes from breast cancer study and the other application involves employee information from a company.

5.4.1 Mapper on 3-D Shapes

The first application is representing a two-dimensional picture of a human hand; the result was representing a data set of a thousand points by a network of 13 nodes and 12 edges [Lum, 2013]. Following the steps in the Mapper algorithm, the process begins with sampling

data points from the image in order to create point cloud data resembling a hand. After processing the data with a filter function and partitioning, clustering and network construction produces Visualization D in Figure 7. This output captures the overall shape of the hand in size and orientation as well as the position and length of the fingers and thumb. Additionally, the branches in the network reflect the location in which the fingers and thumbs extend from the palm of the hand.

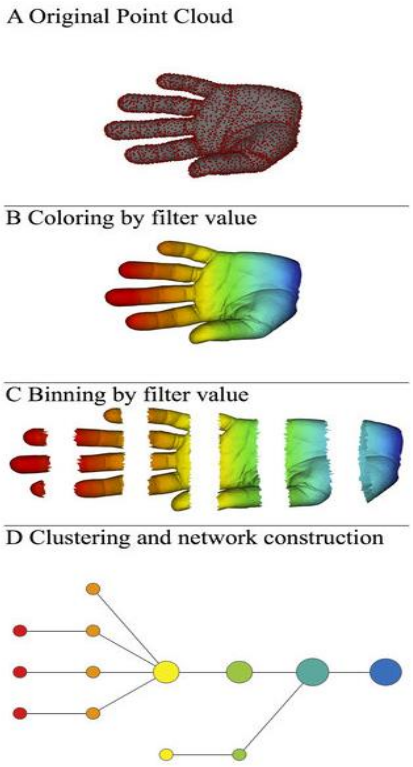


Figure 7: TDA Pipeline [Lum, 2013, Figure 1]

For the next few examples, each of the point cloud data resembles a topological object such as a circle, torus, and a knot. Beginning with Figure 8, the point cloud consists of a circle of data points and several other points that serve as noise. Using the Singular Vale Decomposition filter with the Euclidean metric, single-linkage clustering, and a balanced cover, the network to the right of the point cloud represents the circle as a loop in the network. The noisy data does not interfere in representing the circle; in fact, the algorithm places the noisy points along the path in

the network. Notice as well that the four clusters of red nodes each receive their own branches in the topological summary.

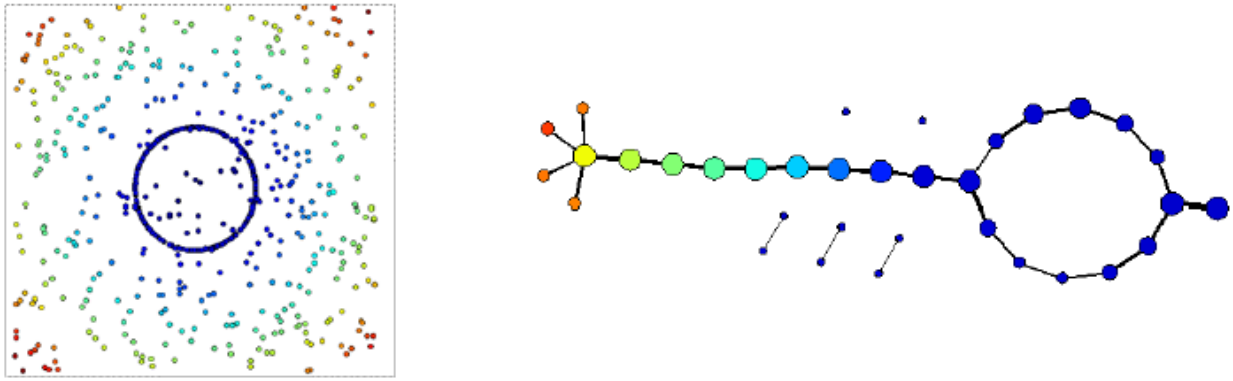


Figure 8: Topological Summary of Noisy Circle [Kraft, 2016, Figure 4.7]

From Figure 9, sampled data points from a three-dimensional torus compose the point cloud data for the next example. These summaries use the SVD filter functions, Euclidean metric, single linkage clustering, and a balanced cover. Focusing solely on the point cloud data, some initial topological features include a hole enclosed by several data points and, overall, one connected component. Both topological summaries reflect these features, even though they use the same filter function with different parameters. The top and bottom networks exhibit a single-connected topological object and a single loop.

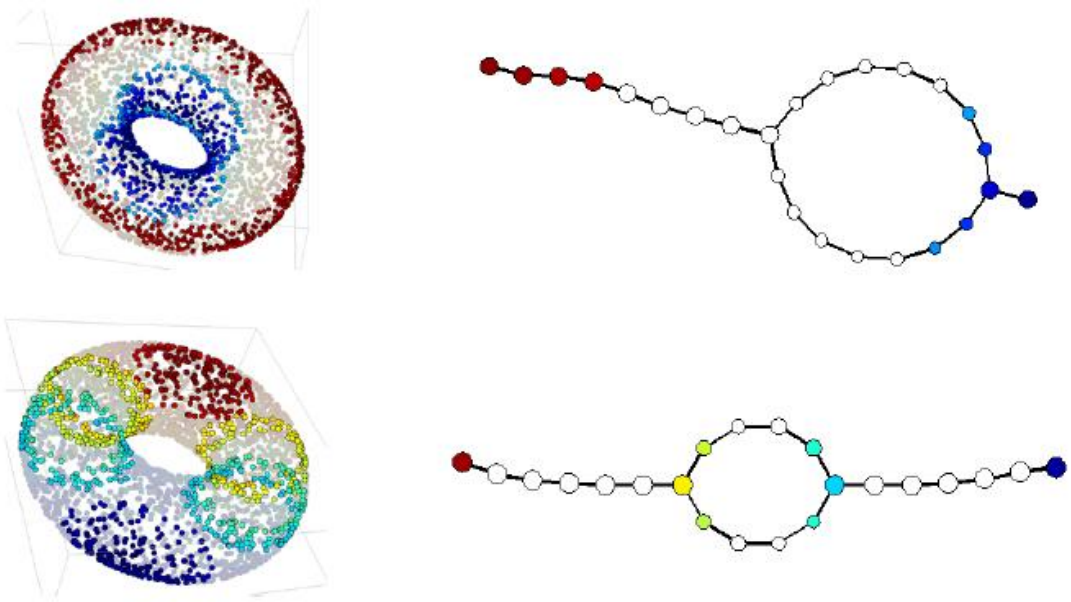


Figure 9: Summary for torus using first SVD (Top) and Summary for torus using second SVD (bottom) [Kraft, 2016, Figure 4.7]

Of course, these examples illustrate the algorithms ability to capture essential topological features and shape, but the data sets so far have been synthetic data samples that reside in two or three dimensional space. The next section touches upon real-life data sets collected from both industry and medical practice.

5.4.2 Real-life Applications

In the following studies, researchers apply the Mapper on real-world data. The first application pertains to a breast cancer study whose usage of Mapper led to the discovery of a new subgroup of breast cancer tumors. Data used for the study comprised of *Nederlands Kanker Institut (NKI)* data composed of 295 tumors, *Breast Cancer Normal (BCN)* data consisting of 13 normal breast tissue samples, and data from Ulleval University Hospital (ULL) containing 46 tumors and HERSCH containing 188 primary breast tumors [Nicolau, 2011]. In the last application of Mapper, the problem was determining indicators of loyalty for customers buying two brands of orange juice. The available data set came from a marketing study of customer purchases of orange juice at five supermarkets consisting of 1070 observations on 17 predictors

[Kraft, 2016]. These applications should showcase how the Mapper method can be adapted for data from various sources.

Since the purpose of topological data analysis is to provide a global representation of the data, incorporating Mapper into the analysis of breast cancer data assisted in addressing the high dimensionality of the biological data. The paper referred to the resulting method of analysis as Progression Analysis of Disease (PAD); PAD was the study's "approach to the data analysis of disease that unraveled the geometry of data sets and provided an easily accessible picture of the data through Mapper and Disease-Specific Genomic Analysis (DSGA)" [Nicolau, 2011]. This new method added an additional level of pre-processing before running the data through Mapper; the preprocessing entailed transforming the breast cancer data set into new vectors highlighting specific properties. Defined by a mathematical method known as DSGA, the transformation measures the extent of deviation between diseased tissue and healthy tissue. By transforming the data using DSGA, the new data brings out unique biology, allows for wide variability of normal, and emphasizes the degree of deviation from healthy tissue [Nicolau, 2011]. Doing so, in other words, customized the Mapper algorithm to fit the nature of the medical data. Once the transformation of the data is completed, PAD applied filter functions provided by Mapper to the transformed data set; in this study, researchers defined filter functions that computed the vector magnitude of the L^p norm. Applying PAD on the NKI data formed the graph shown in Figure 10. Note that, since the filter function highlighted the deviation from normal, blue-colored nodes would pertain to normal and normal-like tumors; red-colored nodes, consequently, indicate that the deviation from healthy tissue is large. In this figure, there are two progression arms or paths that branch out from the blue nodes. The progression arm with red nodes, as discovered in the study, identified a new subgroup of "Estrogen Receptor-positive breast cancers that express high

levels of c-MYB and a high survival rate in patients" [Nicolau, 2011]. Even though this new subgroup does not fit into previously identified breast cancers and exhibit these properties, the researchers were able identify the subgroup only with the help of Mapper. The next application only reinforces the idea that the Mapper algorithm strives to highlight certain topological information.

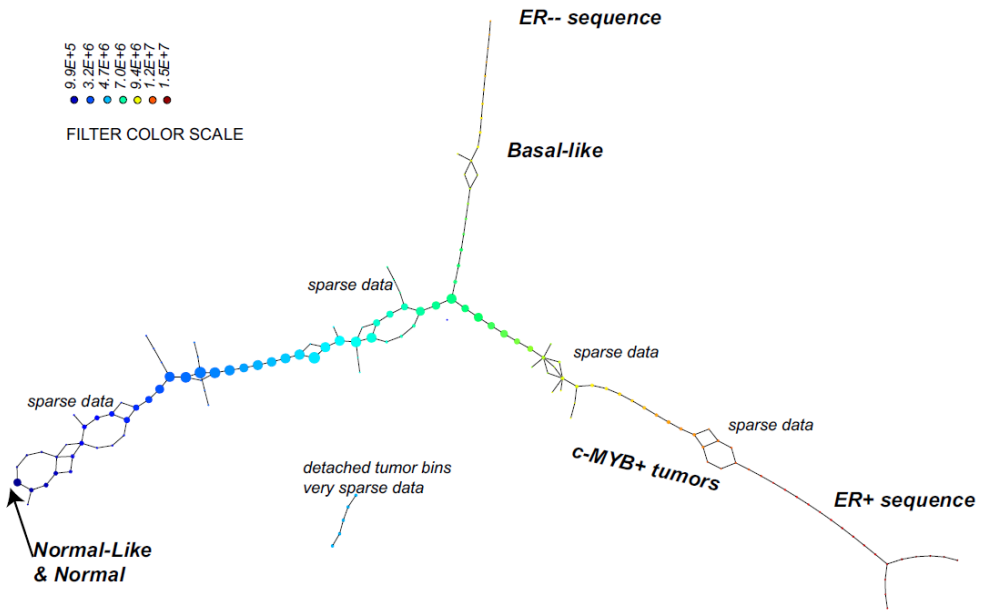


Figure 10: PAD Analysis of NKI data [Nicolau, 2011]

In this study concerning marketing research data, the goal of the study was to understand which factors influence a customer's decision to purchase between two brands of orange juice, Citrus Hill (CH) or Minute Maid (MM). Some variables utilized within the study include sale prices for each brand, price differences, discounts offered, week of purchases, and loyalty of customers. An initial statistical approach using penalized logistic regression with L1 penalty concluded that good predictors for purchasing CH were loyalty to CH, price difference between MM and CH products, and an indicator variable for a specific store. This store was a good predictor for purchases because, according to the data, it offered more special offers and the best

sale prices for CH [Kraft, 2016]. The study then went onto to find predictors by using a combination of Mapper and penalized logistic regression to validate predictors from the previous approach and possibly find new predictors. Figure 11 shows the resulting topological summaries of the data set colored by first SVD filter function (left), by whether or not the purchase was CH (center), and by whether or not the purchase was MM (right).

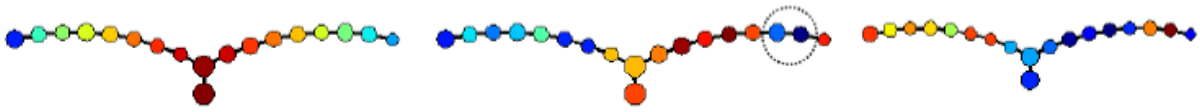


Figure 11: Initial Colorings of Topological Summaries [Kraft, 2016]

Observing the center and right topological summaries, customers purchasing CH gathered on the right side of the graph whereas most customers purchasing MM were localized in the left branch. Notice that the two nodes in the dotted black circle have similar colorings in both summaries; comparing the two groupings concluded that MM customers in the right nodes possessed distinct purchasing reasons differing from MM customers in the left side. In fact, penalized logistic regression again selected loyalty to CH, price difference, and Store 5 as the best predictors that distinguish two nodes on the right from the left branch. Figure 12 illustrates the colorings by these predictors.

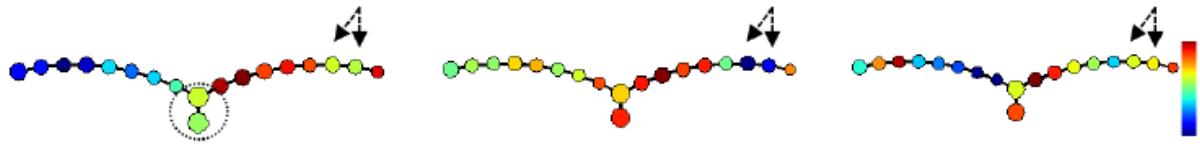


Figure 12: Topological Summaries colored by LoyalCH (left), PriceDiff (center), StoreID5 (right). [Kraft, 2016]

One interpretation gained by these colorings would be that, within the two nodes designated by the arrows, customers are moderately loyal to CH but choose to buy MM when the price

difference (sale price MM – sale price CH) is low [Kraft, 2016]. Additionally, observe the two nodes in the dotted circle. These colorings indicate that these nodes contain customers who are moderately loyal to CH and high price differences between CH and MM; alternatively, these customers prefer lower sale prices from CH. With Mapper, researchers from these studies were able to understand the data intuitively and, then, guide their efforts for further study. The next section discusses the cyber data involved.

5.5 Cyber Data and Analytics

In this section, the main focus is on the cyber data available and the cyber analytics used in both filtering and validation of the data. The cyber data is a product of the BRAWL project; BRAWL is an abbreviation for Blue versus Red Agent War-game Evaluation. One goal of this project is to generate and collect real-time synthetic data from within a computer network. Pertaining to cyber analytics, another project involves collecting analytics that indicate malicious activity and to storing them in a repository.

5.5.1 BRAWL Data

Data collected through BRAWL is synthetic data capturing the simulation of an intrusion. To summarize the procedure of BRAWL in a sentence, two teams of cyber agents compete against each other; the blue team handles detection of red team activity whereas the red team simulates an adversary gaining access to the system [Wampler, C., personal communication, July 2016]. These games occur on a hyper-sensing network that generates data and records logs of network activity in real-time. BRAWL data is essentially information from Windows event logs and, thus, conforms to an XML-based schema in its raw form. After parsing and extracting information from the event logs, the data takes on a Comma-Separated Values (CSV) format because data in this format are easier to read row-by-row and process. The resulting data set contains as much as 153 fields; however, much of these fields repeat information and only

contribute to the distribution of missing data. Fields found in the data include log type, host name, severity, keywords, process ID's, group ID's, source port, destination port, image path, command line, parent information, integrity level, executables, and hashes. The data is a record of activity from the red and blue team agents as well as from normal users on the network. With a list of red team activity used as ground truth, identifying the red team agent activity was one goal of this project.

5.5.2 Cyber Analytics and Analytics Repository

After each BRAWL cyber game, the output is a multitude of indicators that highlight malicious behavior or by-products of red team activity. Each indicator, also referred to as a cyber analytic, no matter how trivial or groundbreaking goes into the Cyber Analytic Repository (CAR) for further study and testing. While in storage, additional information is necessary to describe the new cyber analytic; such information pertains to likelihood of malicious activity, stage of development, and related ATT&CK technique (Appendix 4). All cyber analytics begin as ideas and, after testing and validation, go onto further stages of development or become refuted. The indicators highlighted in this paper include Powershell execution, processes spawning CMD.exe, and Server Message Block Write requests. See Table 3 for summaries of each available analytic.

Cyber Analytic Summaries	
Analytic 1	Typically, when a user runs a command prompt (cmd.exe), the parent process is explorer.exe or another instance of the prompt. Spawning the process cmd.exe from certain parent like Microsoft Outlook may be more indicative of malice.
Analytic 2	Detects Powershell scripts. Powershell is a scripting environment included in Windows that is an easy way to circumvent security measures.
Analytic 3	Detects SMB Write Requests. SMB provides a means for remotely managing a file system. SMB Write indicates lateral movement to a host.

Table 3: Analytic Summaries

5.5.3 Applying Mapper to Cyber Data

Implementing the aforementioned cyber analytics along with other analytics in the repository have accomplished the task of detecting red team intrusions, and most analytics in the repository have undergone improvements. The overall goal of applying the Mapper to this data was to gain qualitative understanding of the data from a different perspective. Rather than focus on a subset of the fields, the Mapper provides a global view of the data while still considering all the fields provided from the cleaned data set. Gunnar Carlsson, one of the researchers who co-founded Ayasdi and developed Mapper, stated that topological modeling can lead to taxonomy generation, hotspot outcome-based analysis, and geometry-based feature selection [Carlsson, 2015]. From this method, one desired outcome of applying Mapper was a taxonomy user activity or at least a distinction between malicious and benign user behavior on the network.

6 Methodology

The overall goal of the project was to develop indicators of attack using Topological Data Analysis as well as to enhance the Python Mapper by designing a customizable base application to call Mapper functions. This chapter describes the design of the topological data analysis system, the cloud storage service that holds all the cyber data, and the case studies and their desired outcomes on the data. In addition, this chapter introduces the data analysis process that data scientists utilize when extracting generalizations and conclusions from the output of Mapper.

The system overall consists of two layers of abstraction that begin with extracting data from a cloud storage service like ElasticSearch and ends with the analysis layer for the data. These two layers are referred to as the data extraction layer and the topological data analysis layer. As for the proposed experiments with the data, they consist of experimenting with filter functions, tweaking parameters within the TDA layer, and applying mapper on a different data

set. For the final section, most of the discussion pertains to the mathematical component of this paper.

6.1 System Design

The Python Mapper implementation lacks two key aspects that limits the data analysis stage; these aspects are the ability to extract data directly from a source and the ability to analyze the data beyond creating visualizations. Extracting data directly from the source entails being able to take a subset from the data source, to clean it by removing any non-numeric attributes, and to package the cleaned data in a readable format such as a CSV-formatted file. Thus, integrating abstractions of these aspects with the implementation should improve its capability. Generally, for all following sections, the design aspect consists of brief descriptions of the classes involved.

6.1.1 Data Extraction Layer

The main task of the data extraction layer is to pull in all requested data from a cloud and format the data into a readable file for the next layer, cleaning and parsing all the data in the process. In terms of structure, there are five main classes that include the ESManager class, File class, Parser class, JSONFile class, and CSVParser class. Both the File class and Parser class are parent classes for the JSONFile and CSVParser classes, respectively; the reason is that future versions of the system would have the freedom to implement additional format changes such as reading from a plain text file. Establishing a connection with the ElasticSearch cluster, querying for data, pulling the data are responsibilities of the ESManager. The ESManager is also responsible for reading in pre-configuration files containing server names or port numbers for designating the Elasticsearch cluster. As for the File and JSONFile classes, these classes have the task of reading in a data file formatted one way and outputting a file formatted according to user-preference. Output from Elasticsearch uses JavaScript Object Notation (JSON), but the Mapper

toolchain requires Comma-Separated Values (CSV) format. Thus, the JSONFile class transfers the data to a CSV file as desired. The purpose of the Parser class and CSVParser class is to represent any non-numeric data as numerical values either by assigning integers for all possible values within a column or transforming each data point into a vector of continuous values. As with the file classes, the reason for using only numeric representations for the data is an implementation-specific requirement of the Mapper toolchain. Aside from structure, the process of data extraction follows five steps.

First, data extraction begins with an ESManager object pulling data from an ElasticSearch cluster. The output from this operation is, as stated before, a file formatted in JSON. Next, a JSONFile object would export all information from this file to a CSV-formatted file. A CSVParser object would then clean the data by removing extraneous columns or repeated columns from the file and parse the file. Again, parsing the file means assigning numerical values to all non-numeric data as well as categorical values like ID numbers. Finally, the last step in data extraction is compressing the cleaned and parsed CSV-formatted file into a .gz file. Now, the data would be ready for the next layer in the system, the topological data analysis layer.

6.1.2 Topological Data Analysis Layer

The main purpose of this layer is to execute the steps in the Mapper algorithm and output both a visualization of the data and CSV-formatted files for further analysis. To accomplish this goal, the Mapper class inherits all methods from the mapper python module. The OutputManager class is responsible for showing a visualization for the data along with creating CSV-formatted files corresponding to each node in the visualization. In the first step of the procedure, the data analysis layer takes in an archived file (.gz extension) and reads in the data; as discussed in the algorithm, the system stores all the data points into a matrix. A Mapper object would go through all steps of the algorithm from filtering to clustering.

One alternative way for implementing this TDA layer was to simply modify the source code for the Python Mapper, changing the GUI to incorporate more options and more filter functions. Instead, starting from scratch would be a more straightforward process since only certain methods from the mapper python module were required. In order to know which methods were necessary, the Python Mapper had a feature for generating standalone python scripts that would accomplish the same task as using its graphical user interface. Using these python scripts as a starting point would make the system more adaptive to future changes and allow integration to be more direct.

6.2 ElasticSearch

Originating from Windows Event Logs, the cyber analytic data had contained logs of process and user data recorded every second for the duration of a month. ElasticSearch, thus, was a suitable tool for searching through and storing this data. Essentially, ElasticSearch was a distributed search engine built for the cloud, providing features such as various API's, long term persistency, full-text search in real time, and the functionality of Lucene [Dixit, 2016]. These features have attracted many large and small organizations to adopt ElasticSearch for not only a search tool but also a data storage tool. This section will discuss the inner workings of ElasticSearch by first introducing Lucene.

6.2.1 Foundation of ElasticSearch: Lucene

Written in Java, Apache Lucene is a search engine library from which ElasticSearch inherits its functionality. ElasticSearch, as a result, is able to index documents and search them with its full text search capabilities [Dixit, 2016]. Before touching upon this functionality, there are a few terms to understand. A document is known as "the main data carrier used for indexing and search within ElasticSearch; alternatively, it is a JSON object that contains the actual data in key value pairs" [Dixit, 2016]. Having the data formatted in JSON allows for fast lookup

because one can reference values by keywords, similar to indexing in a Python Dictionary data structure or Hash table. Fields compose sections of documents, and they consist of a name and a value. Finally, terms are units of search representing a word from the text and a token represents "an occurrence of a term from the text of the field" [Kuc, 2015]. In other words, documents are containers of information, fields provide organization for the information held within documents, and terms are the indices involved in searching for documents.

Keeping those basic concepts in mind, one can now understand how ElasticSearch handles querying. Apache Lucene stores information using a data structure called inverted index; this data structure "maps the terms in the index to the documents" [Kuc, 2015]. As a result, the data in an inverted index relies heavily upon the terms rather than the documents, a major difference from relational databases. The process of mapping the data in the documents to the inverted index as well as changing the query text into terms for searching purposes is known as analysis in the realm of ElasticSearch. Analysis has an important role in indexing and querying. The analyzer, composed of a tokenizer and some number of filters, carries out this process. The tokenizer component has the responsibility of dividing the text from documents into tokens, preparing the tokens for filter processing [Kuc, 2015]. With Lucene at its roots, ElasticSearch is able to build upon these foundations and become not only a search tool but also a data storage tool.

6.2.2 Key Concepts of ElasticSearch

To fully understand ElasticSearch, there are a few more concepts to learn in addition to documents, tokens, and terms. Utilizing a bottom-up approach, an index is the next concept in the structure of ElasticSearch architecture. An index is a "logical namespace under which ElasticSearch stores data and may be built with more than one Lucene index using shards" [Kuc, 2015]. Similar to how data is read from and saved to databases, an index stores documents for

future queries. Shards, in turn, are containers that consist of Lucene segments; shards enable ElasticSearch to distribute the data within an index and allow for a vast amount of storage space. Moving up a level in the architecture, nodes represent a single instance of an ElasticSearch server where one or more shards can reside. Types of nodes include master nodes, data nodes, and tribe nodes. Finally, a set of ElasticSearch nodes is known as a cluster; with its distributed nature and the collaboration between clusters, ElasticSearch can manage a large volume of data and work done multimode clusters can go uninterrupted even during repairs or maintenance [Kuc, 2015]. The system for TDA will incorporate the capabilities of ElasticSearch with the Python Mapper in order for direct access to data.

6.3 Data Scientist's Workflow

The main phases of the Data Analytics Lifecycle are discovery, data preparation, model planning, model building, communication of results, and operationalization [EMC, 2015]. As noted in an interview with an Ayasdi data scientist and various studies using Mapper, there is a similar workflow that data scientists from Ayasdi have adapted for interpreting Mapper's constructed visualizations. This workflow begins with data conditioning and filtering and ends with topological summaries of the data; repeating the process can lead to gaining a better understanding of the data [Symonds, 2015]. The desired end result from following these steps is to uncover patterns and predictors for the given data.

The first step before visualizing the data with the mapper is data preparation. Probably the most critical task or at least most labor-intensive, data preparation affects the next stages in data analysis because any data conditioning or data transforming would determine what patterns are shown in subsequent models or summaries. In fact, teams aside from Ayasdi have commonly spent 50% of a data science project's time understanding the data and deciding how to represent the given information in a way that conforms to the requirements of the project [EMC, 2015]. A

good portion of the project was spent learning how to represent each data point as an event log from a host relaying messages within the network to the mapper. However, not all fields provided in the event log were necessary for data analysis. The attributes that would be good representatives of the data set were fields from the Data Model data structure.

The Data Model was an organization of objects that encapsulate consistent fields throughout the event logs and possess both actions and specific fields. Some example objects included processes, flows, drivers, files, and modules. Note that fields associated with the data model object contained most if not all the information within the data set; the other fields duplicated this information. Although each data model object (e.g. flows or processes) had different actions and fields, the cleaned data set included all types of data model objects. In total, each point in the data set would consist of 47 fields of only data model fields, timestamp, and port number fields. In order to ensure that the amount of missing data after cleaning was much lower, the validation method selected was to pull four sample data sets and compare the percentage of blank cells in these samples with cleaned versions of the same four sample sets. Each sample set contained 500 data points from different time intervals. This method would quantify the amount of missing data in the process of cleaning.

Understanding the underlying structure of the output was a challenge by itself, so any initial analysis resulted in more observations and questions rather than conclusions. Before starting the analysis process, the first step in the workflow was to select initial features from the data set. Then, using Mapper, the next step was to create topological summaries of the data based on the chosen features. Any non-trivial topological features like loops, flares, or connected components were significant features to highlight during analysis of the output. Additionally, any localizations of data points represented anomalous or interesting behavior. Afterwards, the next

and probably the most difficult step was to generalize the underlying reasons for both the structure of the output and the groupings of data points. Knowing the properties highlighted by the filter function also contributed to the understanding of the output. Once identifying the common features resulting from the localizations, the final steps were to utilize this information to develop a better topological summary and to repeat this process. The goal of the following case studies was to explore the cyber data and observe the Mapper implementation's ability to represent the data.

6.4 Case Studies with Data

Based on the numerous applications of TDA, the desired result with this data was a visualization that identified a distinction between malicious activity and benign events or a taxonomy of events on a network. To reach such a result required much experimenting with the data. Experiments consisted of changing filter functions, using a combination of filter functions, or even adjusting the parameters for each filter function.

6.4.1 Continuous vs. Categorical

As previously stated in the system design section, parsing of the data was either through enumeration of values within each column/attribute or by computing a vector of continuous values for each data point. By using both methods, the result was having two different data transformations on the original data set. Each would require its own measure of distance or similarity; for instance, the continuous values used Euclidean distance and data set with categorical data utilized the Hamming metric. In terms of filter functions, the case study would use the L-Infinity Eccentricity function in order to detect any data points distinct from the center of data. Comparing the resulting visualization should have indicated whether or not the Mapper algorithm was better equipped to finding malicious activity using only the enumeration or using another level of pre-processing.

6.4.2 L-Infinity vs. L-1 Filters

The difference between the L-Infinity and L-1 filter functions is that the L-Infinity function computes the maximum distance between a given data point and the center of data, whereas the L-1 filter function computes Euclidean distance between a given data point and the center. An alternative way to observe this difference is that the norm of a vector with p = 1 differs from the infinity norm in a similar fashion. By comparing the output after filtering by both functions, there should be evidence supporting whether or not the L-1 filter overlooks any interesting groupings/topological features or vice versa.

6.4.3 Experimenting with Resolution

Since the number of intervals and percentage of interval overlap affects the resolution of the data visualization, experimenting with these parameters would provide insight into how much of an impact these parameters have upon the output. Taking the resulting visualization from Figure 9, this brief case study focused on the 2-Torus because it is a well-known object studied in topology. Under the same filter and clustering method, the modifications to the parameters consist of increasing and decreasing only the intervals, increasing and decreasing only the overlap, and tweaking both parameters together. One possible outcome would be that decreasing the overlap should result in more disconnected components in the network.

6.4.4 Applying Mapper to WBCD Data

When testing the capabilities of the Mapper implementation, using the method on medical data seemed to be a good approach given its other applications in the field. The Wisconsin Breast Cancer Diagnostic (WBCD) data set was a good candidate since each data point had labels for ground truth and contained only real-valued features. Coming across the data set in Kaggle, there was already CSV-formatted file to parse on and allow the Mapper to process. There were 569

data points generated from 569 patients, 212 with cancer and 357 patients with fibrocystic breast masses (benign) [Wolberg, 1995].

Features in the data included ID number, diagnosis (Boolean value for malignant or benign), and ten real-valued features computed from a digitalized image of a fine needle aspirate (FNA) of a breast mass. These ten features describe cell nuclei characteristics evident within the image; computed for each cell nucleus, they are radius, texture, perimeter, area, smoothness, compactness, concavity, concave points, symmetry, and fractal dimension [Wolberg, 1994 and Wolberg, 1995]. More information about these features can be found in both articles by W. H. Wolberg listed in the List of References, but essentially radius and area describe nuclear size of the cell and the rest refer to the nuclear shape. For each feature on each image, there is a mean, worst, and standard error value, resulting in a total of 30 features in addition to the ID number and diagnosis.

In applying the Mapper method, 299 data points made up the testing set, consisting of 256 benign points and 43 malignant points. The parameters involved were the Chebyshev metric, the Singular Value Decomposition filter function, and a balanced 1-dimensional cover. Coloring on different features showed how the method created groupings of the data. After finding these features, reading through relevant medical literature pertaining to this data set verified that these features were significant. The last step in this application of Mapper was to confirm any findings with a validation set composed of the remaining data points.

6.5 Evaluation of Visualizations

Initial observations of the Mapper output could be confusing if not all factors are considered. Interpreting the visualization required understanding what the selected filter function conveys about the data, how transformations of the data change its meaning, how different colorings of the data may provide some insight into the data, and various other aspects. Since

working the synthetic cyber data had been an exploratory effort, evaluating the visualizations was a challenge, but there were three main aspects that determined whether or not the output was useful or meaningless. These aspects included localizations/groupings or malicious activity, number of significant topological features like loops or flares, and the persistence of shape evident in the visualization after a few perturbations in parameters.

For Mapper and other clustering methods, any localizations or groupings of red team activity would indicate that there were certain factors distinguishing them from normal user behavior; these factors/features would most likely be indicators of cyber-attack. Alternatively, if the chosen filter function highlighted the probability of malicious activity and the resulting visualization was able to group all adversarial activity in high-valued nodes, then that visualization should provide valuable insight into what are good cyber analytics for cyber threats. Thus, the accuracy or ability to create pure groupings within the visualization was an important component of a good visualization.

As previously stated about the Ayasdi data workflow, looking for topological features in the Mapper visualization would be essential in comprehending the meaning behind the shape of the data. These topological features may possibly highlight specific behavior within the data. Most notably, many applications of Mapper like the Breast Cancer study and study with diabetic patients have identified these special topological features to be loops (continuous circular segments) or flares (long linear segments) [Lum, 2013]. Loops in the Mapper output have typically inferred periodic behavior within the data. For example, a student used Ayasdi's commercial product to represent four decades of macroeconomic and market data as a large circular loop; the student later discovered that this shape represented the economic cycle by identifying its distinct stages such as expansion, contraction, and recovery within the network

[Ramanan, 2016]. Flares, on the other hand, usually represent a taxonomy of subgroups within the data like in the Breast Cancer tumor study and market data study. Now, identifying topological features within the output composes a significant part of the analysis, but one must also validate the persistence of the feature.

The final method of evaluating a visualization was observing how the visualization changes after experimenting with the parameters. Such parameters included the number of intervals and the percentage of overlap. Increasing or decreasing the intervals controlled the resolution of the visualization, meaning that less intervals resulted in less nodes in the output but the nodes became larger. In contrast, more intervals led to several smaller nodes. Having a high-resolution in the visualization could possibly allow the user to detect more topological features, but such features would be insignificant compared to some larger topological feature. Concerning overlap, this parameter controlled how much the intervals were allowed to intersect each other. An overlap greater than 50% could result in much denser visualizations where all nodes connect to more than one node, whereas an overlap less than 50% could lead to more disconnected networks. The next chapter will discuss the results of creating data visualizations based on the data sets and their evaluations.

7 Results

Before describing the work done on the cyber data and the output from the Python Mapper, note that this work has been largely exploratory so far and the following visualizations may not seem too interesting at first, but such is the nature of the methodology behind Mapper. The focus now is to highlight the progress of developing cyber analytics including work completed for data preparation and some initial case studies on the cyber data. Evaluation of the data visualizations will consist of the three components of a useful visualization described in the previous chapter.

7.1 Minimizing Missing Data

As part of understanding data and the fields, one of the first steps to conditioning the cyber data was selecting the data model fields to keep consistency throughout the data set. Table 4 and Table 5 show the results from the brief study described in the Methodology Chapter. Particularly, Table 4 displays the number of blank cells contained in the data before and after cleaning as well as the percentage of missing data of the whole data set. Each sample data set, originating from a simulation occurring from March 1, 2016 and March 2, 2016, had 230 columns/fields and 500 data points, resulting in 115,000 cells in total. According to Table 5, the average percentage of blank cells before conditioning was 73.5% and the average percentage afterwards was 51.4%, an average drop of 22.1% in missing data.

Missing Data Results				
	Subset1	Subset2	Subset3	Subset4
Number of Blanks Cells (Before)	84291	84762	84238	84755
Percentage of Blank Cells (Before)	73.296	73.706	73.25	73.7
Number of Blank Cells (Cleaned)	12996	13266	12919	13206
Percentage of Blank Cells (Cleaned)	50.964	52.023	50.662	51.788
Percentage Difference	22.332	21.683	22.588	21.912

Table 4 Missing Data Amounts and Percentages

Population Means and Std. Dev.	
Average Number of Blanks (Before)	84511.5
Avg. Percentage of Blanks (Before)	73.5
Average Number of Blanks (Cleaned)	13096.8
Avg. Percentage of Blanks (Cleaned)	51.4
Average Percentage Difference	22.1
SDEV Number of Blanks (Before)	286.0
SDEV Number of Blanks (Cleaned)	165.7
SDEV Percentage of Blanks (Before)	0.249
SDEV of Percentage Difference	0.407

Table 5: Missing Data Means and Std. Dev.

The data sets that trimmed off any non-data model field were left with 51 attributes where four of these fields held blank columns. Even so, missing values composed approximately half of data samples; the reason, again, was that each set contained various data model object that only possessed values for a distinct collection of fields. For example, flow object typically had values for IP address fields and port name fields. When transforming the data, the Data Extraction Layer section explains how it handles missing values. Aside from the percentage of missing values after conditioning, these results still reinforce the motivation to remove all fields except data model fields.

7.2 Applying Mapper to WDBC Data

The results from applying Mapper to this data set were good in terms of creating groupings of benign and malignant data points. Based on previous research and applications of mapper, flares or branches of one color typically indicated clear distinctions/groupings defined by the Mapper. With a one-dimensional balanced cover of 15 intervals and 50% overlap, Biggest Gap cut-off, and single-linkage clustering, the Mapper created a network (tree) of three flares. Figure 13 shows this result, exhibiting how Mapper grouped all benign points in the right branch (shown in blue) and created two other branches for malignant data points (shown in red).

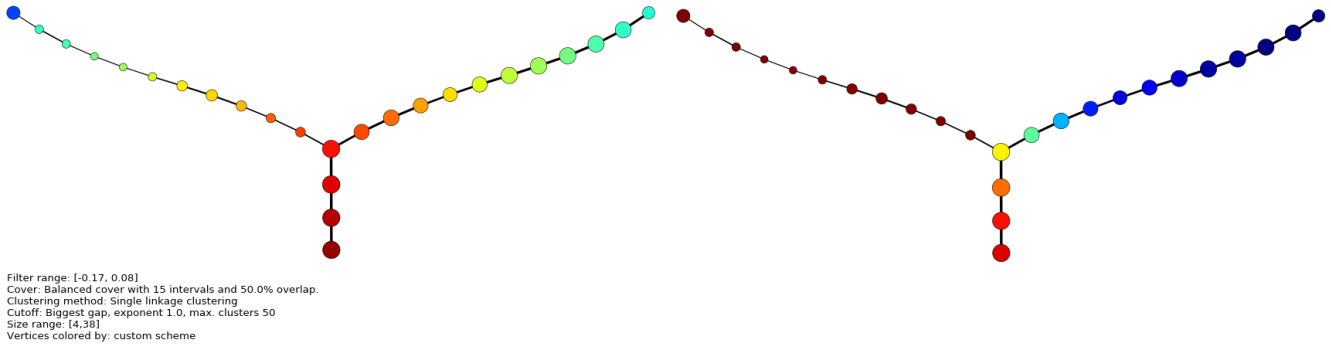


Figure 13: Default Coloring (Left), Coloring by Diagnosis (Right)

After coloring the graph by each of the 30 features separately, three good features that colored the malignant and benign branches differently were the mean texture, worst area, and worst smoothness. Observe in Figure 14 that the branch containing benign data points have low values for the three features, whereas the other branches/flares have a combination of low and high values. In particular, the middle flare contains data points with high values for worst smoothness, but low values for the other two features. According to an article citing this dataset titled *Computer-Derived Nuclear Features Distinguish Malignant from Benign Breast Cytology*, an inductive machine learning model using these three features achieved a cross-validation classification accuracy of 97.5% [Wolberg, 1994].

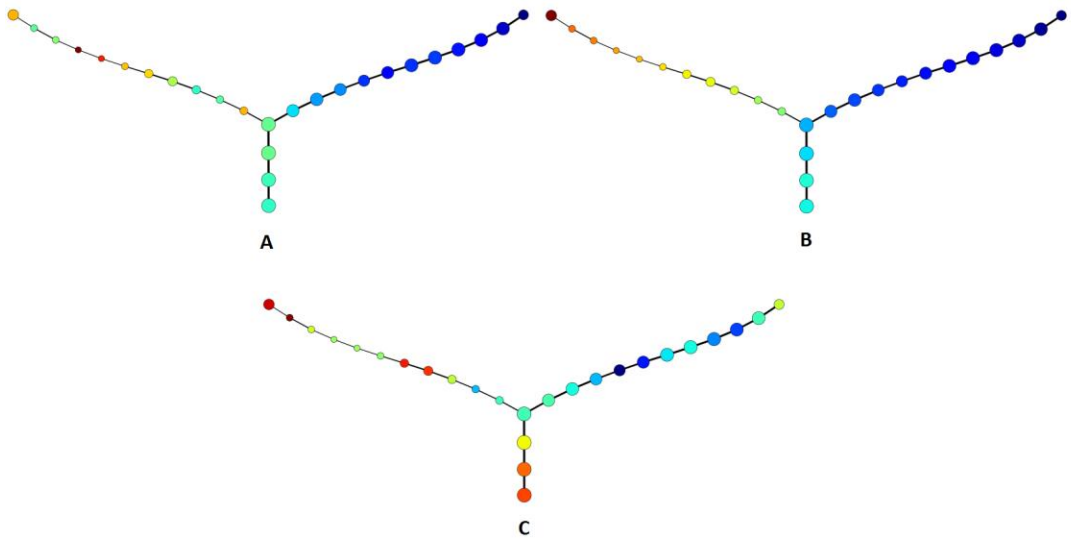


Figure 14: Coloring by Mean Texture (A), Coloring by Worst Area (B), and Coloring by Worst Smoothness (C)

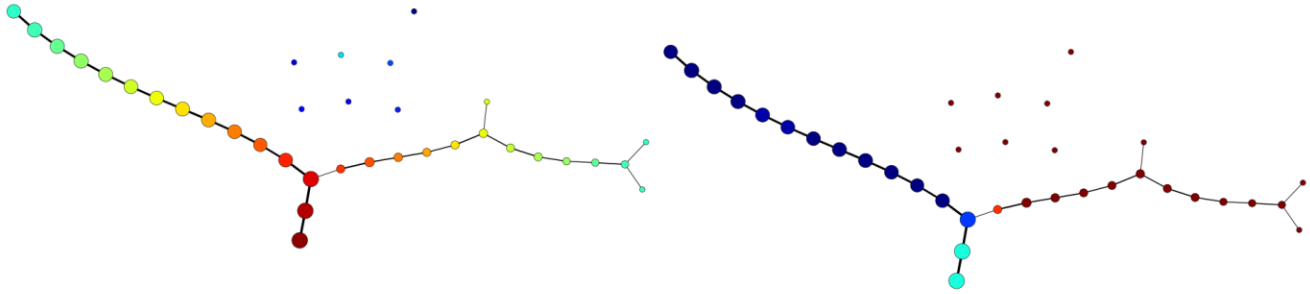


Figure 15: Validation Set; Default Coloring (Left), Coloring by Diagnosis (Right)

In addition to these three features, the branch containing benign data points had high values in the following features: Standard deviation Smoothness, Standard deviation Symmetry, and fractal dimension. Characteristics of the left branch (Figure 14) containing malignant points were high values for worst radius and worst parameter.

In the validation set consisting of 270 data points, shown in Figure 15, there is a similar structure of three flares as in Figure 14. The resulting output has the same parameters as the previous figure; notice again that in the coloring by diagnosis, benign and malignant data points have been separated into different branches. However, in this case, the middle branch has a mixture of benign and malignant data points.

7.3 Visualizations on Cyber Data

The data set utilized held 4,687 data points that recorded activity from both red team agents, blue team agents, and neutral hosts. Data points consisted of event logs that helped monitor the network for intrusions, credential escalation, logins, and error reports. For this simulation, there were in total 69 events where the red team agents were successful in carrying out an assigned task; these data points imitated malicious events on the network. However, only 11 out of 69 red team data points appeared in the 4,687 data points. The following sections describe and evaluate visualizations using different filter functions and parameters on the cyber data set.

7.3.1 Continuous vs. Categorical

Since the fields in the data set were mostly composed of nominal data types like process ID's or event codes, an initial visualization should check if malicious activity could be localized by the differences for each field. The first visualization utilized the L-Infinity eccentricity filter function with the Hamming metric and single-linkage clustering. For the resolution, this visualization used 60 intervals and 50% overlap. Figure 16 and Figure 17 exhibit the output from Mapper colored by default values, Analytic 1, Analytic 2, and Analytic 3. Knowing which data points were red team agents, the blue region on the left side of the network had no malicious data points. The turquoise region excluding nodes in the loops contained two processes executed by the same red team host. In the turquiose loops, there was a powershell command executed by the same red team host and a net command under another red team host. A net command, used in the command prompt, manages many aspects of a network. 3 unknown processes executed by red team agents and 6 powershell commands composed the malicious activity located in the yellow region. Finally, the orange region contained 1 unknown process under a red team agent and, specifically, the orange loop contained 1 powershell command.

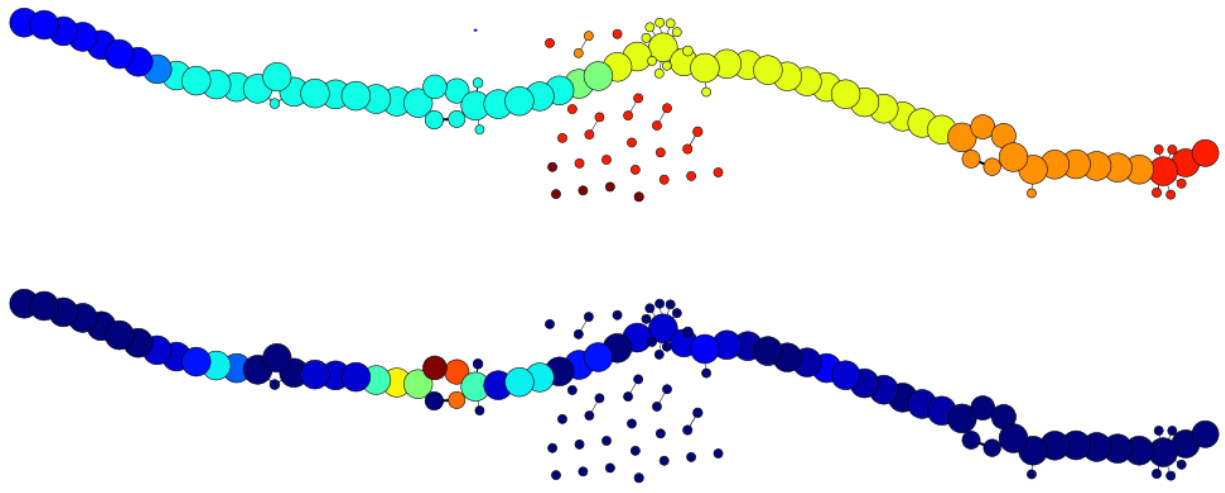


Figure 16: Default Coloring (Top) and Coloring by Analytic 1 (Bottom)

Overall, there were three loops in the output and 22 disjoint components located in the middle of the network. Also, notice that the coloring by Analytic 1 (Figure 16) and Analytic 2 (Figure 17) have a concentration of likely malicious activity in the corresponding turquoise region of the default coloring.

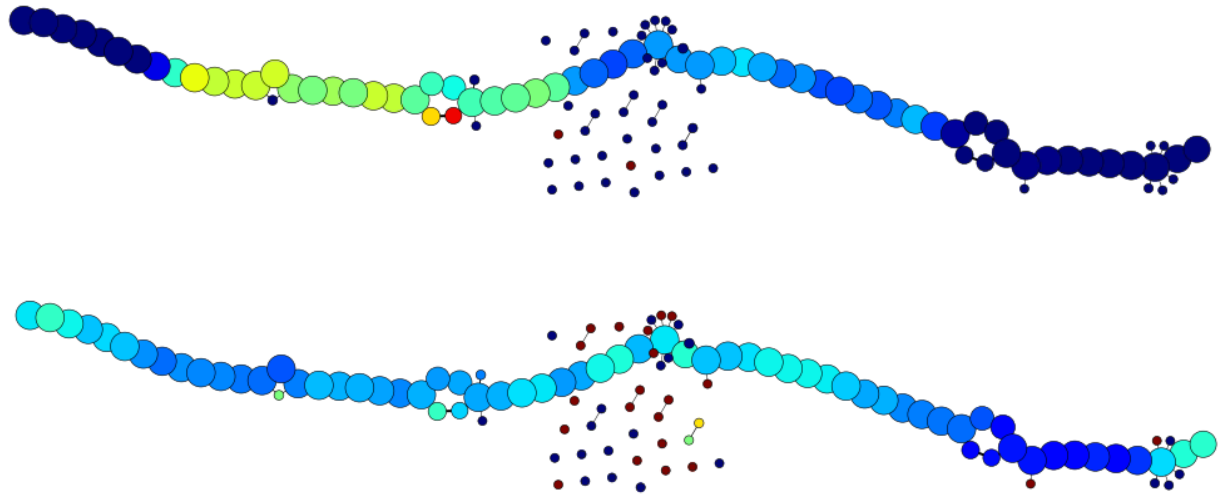


Figure 17: Coloring by Analytic 2 (Top) and Analytic 3 (Bottom)

The next set of visualizations were from transforming each data point into continuous vectors whose components were based on some likelihood of malicious activity. As stated in the Case Studies with Data section, this transformation was a way to incorporate meaning to each data point instead of an enumeration of values. Each of these visualizations used a specific field as a filter function rather than any filters defined in the implementation. For example, considering the first visualization based on Analytic 1, the correspodning filter function would be a mapping between a data point and its score for Analytic 1. Again, the parameters included 60 intervals, 50% overlap, and single-linkage clustering.

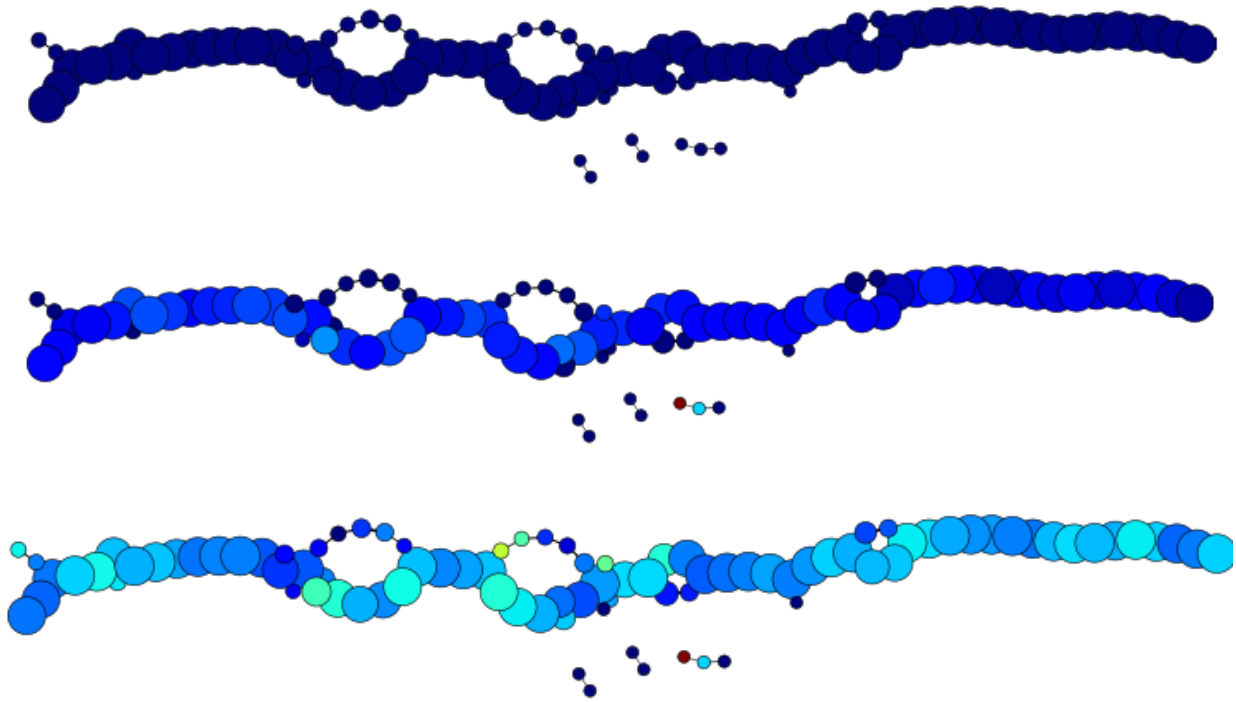


Figure 18: Coloring by Default (Top), Analytic 2 (Middle), and Analytic 3 (Bottom)

Again, Analytic 1 detects processes that spawn a command prompt. Most notably, Figure 18 shows that the visualization for Analytic 1 possesses two large circular loops and three other smaller loops. Reviewing the data points in the first large loop on the left side of the output, there were two unknown processes under two different red team host and 7 instances of non-malicious powershell commands. The second largest loop had 2 unknown processes by two different red team hosts and 2 net commands. For both smaller loops on the right side, there were no malicious data points. There were, however, 2 powershell commands by a red team agent in the smallest loop on the left.

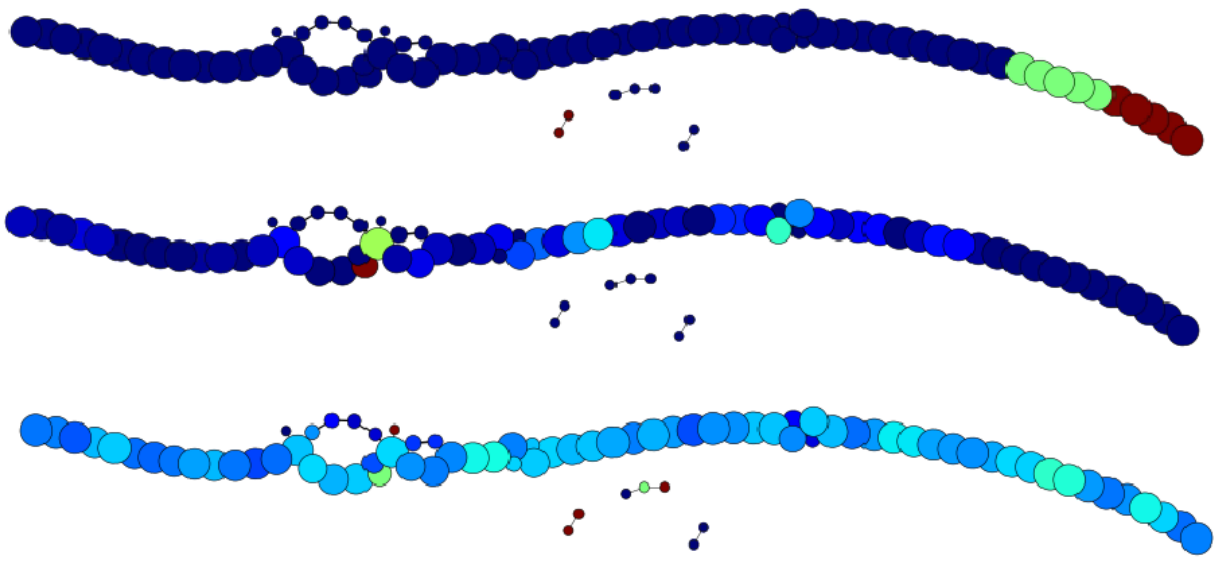


Figure 19: Coloring by Default (Top), Analytic 1 (Middle), and Analytic 3 (Bottom)

Next, Analytic 2 involved detecting powershell executions. Its resulting visualization had a similar structure to the first categorical visualization. The first large loop contained 2 unknown processes by two different red team agents, 1 net command instance, and 1 powershell execution. Both loops beside each other shared a red team data point and 1 instance of a powershell execution, but the smaller loop on the right had another powershell execution and another unknown process. As for the smaller loops on the right side of the graph, the first loop from the left held 2 unknown processes and 3 instances of net command whereas the last loop contained no malicious activity. Observing the red region in Analytic 2 visualization revealed that there were 3 red team powershell executions.

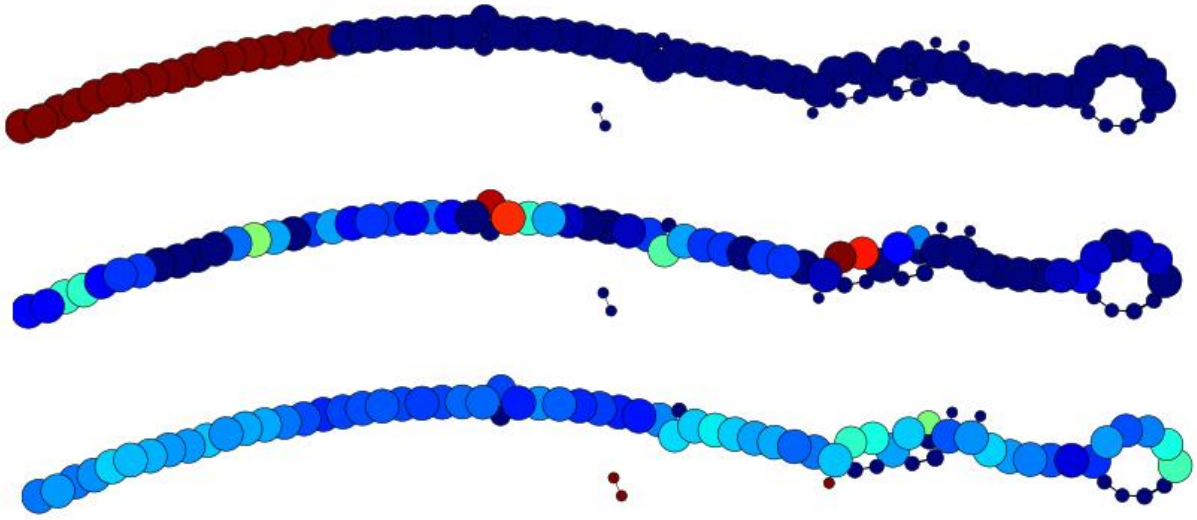


Figure 20: Coloring by Default (Top), Analytic 1 (Middle), and Analytic 2 (Bottom)

Finally, this visualization pertained to Analytic 3; Analytic 3 had the purpose of detecting Session Message Block write requests. Figure 20 shows that there was 1 large loop and 4 other smaller loops scattered along the graph. Starting with the largest loop, there was only 1 instance of a powershell execution. Moving from right to left, the next loop contained 1 unknown process and a net command, whereas the third loop had 1 unknown process and 2 powershell executions. There was 1 net command in the fourth loop, but the fifth loop held no malicious data points. Concerning the red region of the default coloring, there were two instances of unknown processes under the same red team agent. One observation about the loops in all data visualizations was that the part of the loop with smaller nodes typically consisted of logins, logoffs, and processes spawned from a network monitoring program. Note that the network monitoring program, called Splunk, was not indicative of malicious activity. After comparing the categorical and continuous filters, the next step outlined in the Methodology chapter was exploring the difference in visualizations between eccentricity filters.

7.3.2 L-Infinity vs. L-1 Filter Functions

The visualizations for this section involved using both L-Infinity and L-1 eccentricity filters and comparing them to showcase any differences in structure or localizations of data. Pertaining to the transformed data used, only the categorical version of the cyber data by value enumeration was necessary for this case study. Intervals and overlap were fixed at 60 and 50%, respectively.

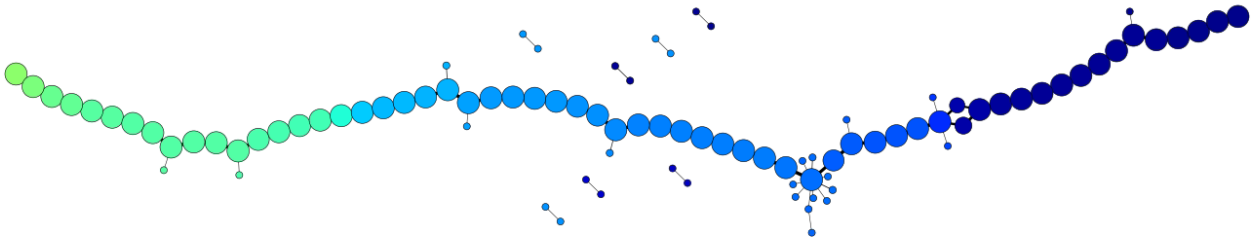


Figure 21: Visualization using L1-Eccentricity Filter

As one can see from Figure 21, the structure of the graph is different from the one found in Figure 16. There was only one loop in this visualization and the color scheme only goes from blue to light green; in contrast, the visualization in Figure 11 had four loops and its color scheme spanned the whole color spectrum (i.e. red to blue).

7.3.3 Results for Modifying Parameters

The following graph outputs from Mapper were results for the brief case study of experimenting with intervals and the number of overlap amount. In addition to using sampled data points of 2-Torus, choices for filter function and metric remain the same as outlined in Section 4.4.1. Some features to observe would be that loops in the output mostly occur in the same colored regions where the single large loop in Figure 9 resided.

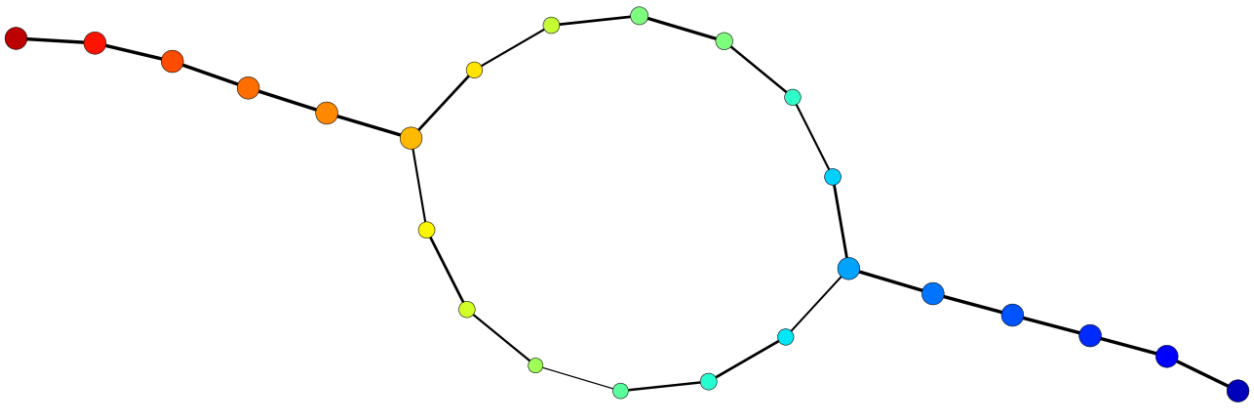


Figure 22: Visualization from Section 4.4

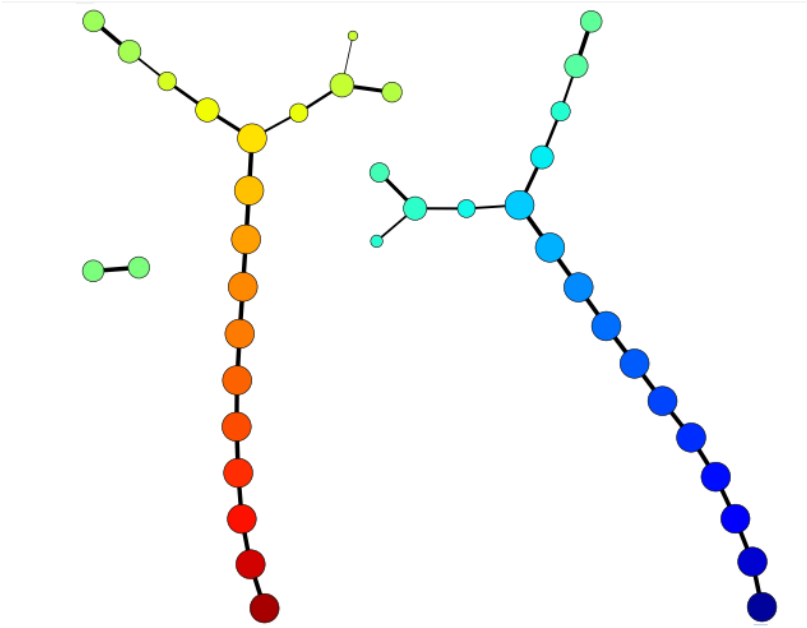


Figure 23: Intervals: 30 and Overlap: 40%

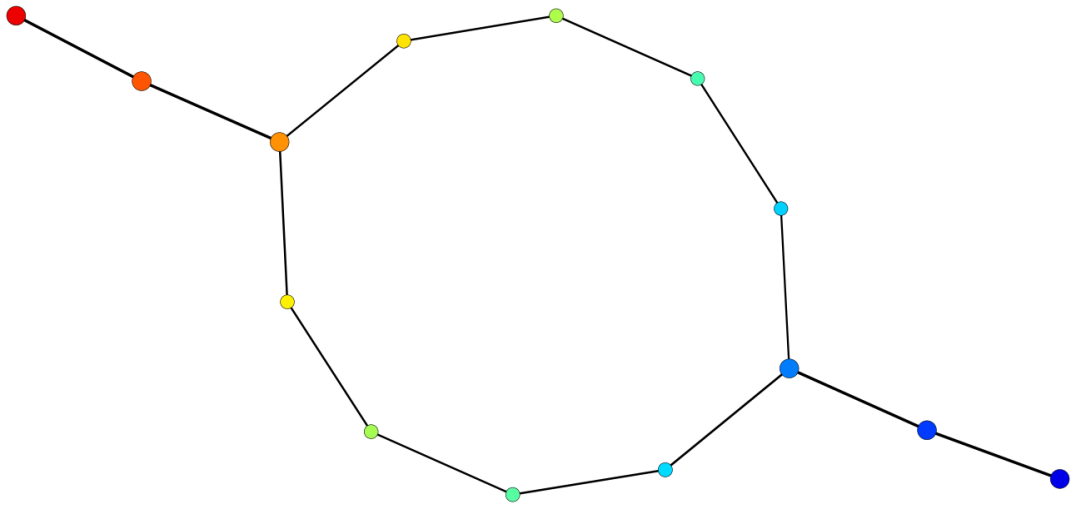


Figure 24: Intervals: 10, Overlap: 40%

From Figure 23 and Figure 24, these visualizations were the outcomes of increasing and decreasing the number of intervals. The increase in intervals seemed to have split the network into two components. Figure 24, in contrast, has remained as one connected component but has fewer nodes in the graph.

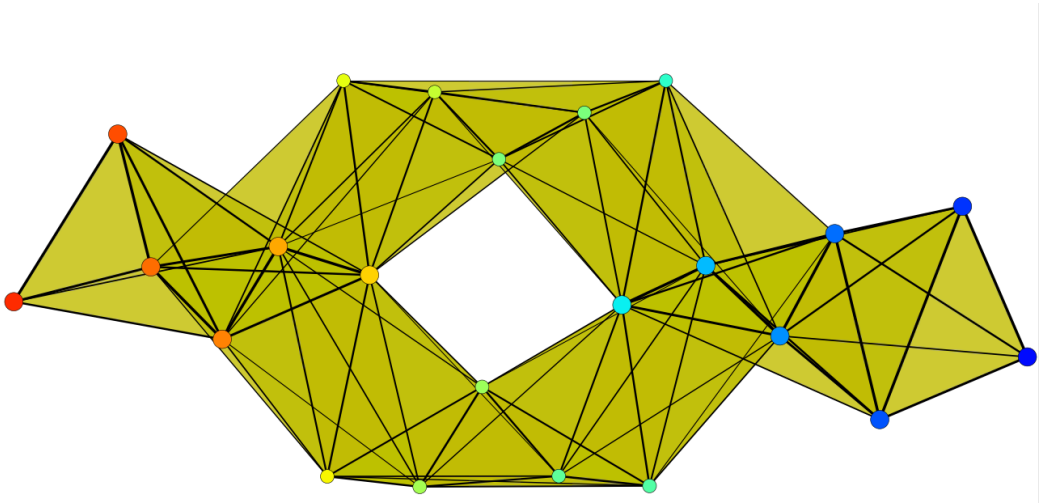


Figure 25: Intervals: 18, Overlap: 80%

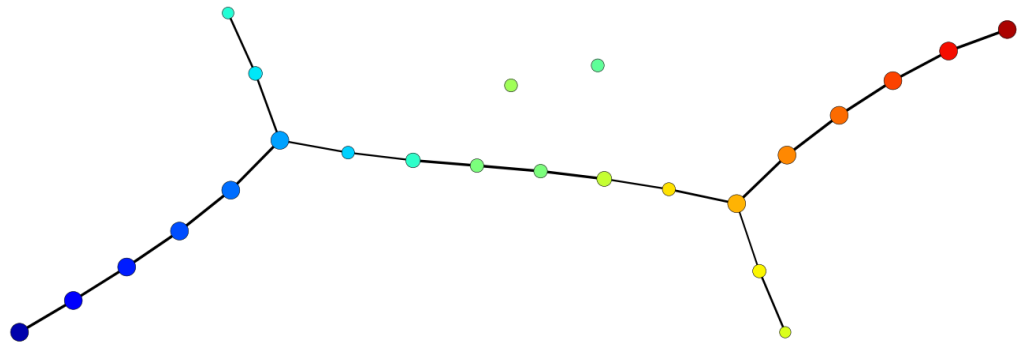


Figure 26: Intervals 18, Overlap: 20

Exhibited in Figure 25 and Figure 26, the visualizations were showcasing the result of increasing and decreasing percentage overlap. Both graphs can be seen to be one connected component, but the visualization from Figure 25 is much denser as a result of the 80% overlap. The loop from the default parameters remained within the visualization from Figure 25 but does not appear with an overlap of 20.

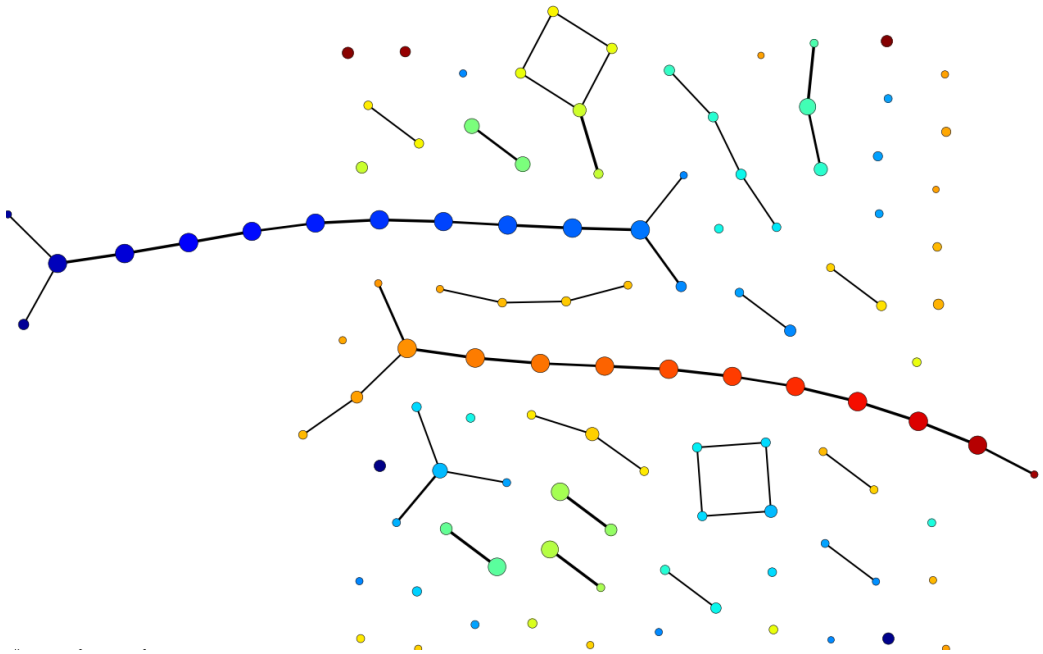


Figure 27: Intervals: 30, Overlap:10%

The purpose of this last visualization was observing the deviations from the original graph after increasing the intervals but decreasing the percentage overlap. Looking at Figure 27, one can clearly see that there are multiple connected components; two components that stand out were the blue and red subgraphs. After reviewing all of these data visualizations, one final observation from Figure 23, Figure 26, and Figure 27 would be that color regions where nodes branch off of each other are the same color regions where the loop occurs in the other visualizations.

8 Conclusions and Future Work

After reading about the successes of the Mapper on various data sets, the visualizations based on the cyber data should have created distinct groupings of malicious and benign behavior or track malicious activity and lateral movement. The next steps afterwards should have been to interpret the underlying reason for that behavior and develop new cyber analytics. Due to the exploratory nature of the research, unfortunately, none of the analysis on the visualizations provided evidence of such phenomena. Any number of factors, like insufficient amount time spent on data preparation or inadequate choices in filter functions, could have led to this outcome. However, there were other findings and conclusions that resulted from progress done in the Results chapter.

8.1 Finding 1: Fields Besides Data Model Can Be Removed

After analyzing the distribution of the missing data from four sample data sets of the same size, the conclusion was that the data set extracted directly from the cloud storage service could reduce its dimensionality further without loss of information. Removing any fields other than data model variables, timestamps, and port numbers would be one way of doing so. In addition to contributing to an approximate 22% drop in missing values in the data set, removing these fields would also rid the data of duplicated information. Another hidden benefit of leaving out these fields would be a reduction in waiting time for extracting the data from the cloud; not requesting for these fields would be an optimization decision.

8.2 Finding 2: Similar Structures in Categorical vs. Continuous Comparisons

Similar to transformations in the Breast Cancer study, using the analytics to transform the data points into vectors of probabilities was a worthwhile data transformation. Going back to the data visualizations in the Results chapter, notice that they have similar structures in that each one has 3-4 loops. Each visualization had different filter functions that highlighted different

properties on the same data set, but there were loops in all of them with parameters for intervals and overlap fixed. Coincidence or not, the loops in each visualization confirm some cyclic behavior prevalent in the cyber data. What cyclic behavior specifically would require further analysis and a more time spent in data preparation.

8.3 Finding 3: Loops in the Visualizations were not Indicators of Attack

Although the analytic-based visualization consisted of loops that contained some malicious data points, there were several other benign and noisy data points accompanying them. The data set had much fewer red team data points in comparison to the rest of the data. Even if malicious data points appeared in any of the loops, it was quite challenging to infer the underlying reasons as to why these point became grouped together. Thus, whether or not these circular loops represented any malicious behavior was inconclusive. However, the existence of these loops in all visualizations of the cyber data was significant. This notion of having fewer malicious data points leads to the next finding.

8.4 Finding 4: Data Requires More Malicious Points

As stated in the previous finding, the number of normal user and blue team data points far outnumber the red team data points. Of course, this situation would replicate a real-world scenario where adversarial activity can be lost in a wave of network activity. However, the problem with this proportion of red team data to normal data was that any groupings of the data did not provide any evidence for a cyber analytic. Other applications of the Mapper noted in web blog posts by Gunnar Carlsson and Ramanan drew upon sufficient proportions of all possible subgroups within the data. For example, the data used to represent the economic cycle originated from Macroeconomic data spanning 1972 to 2015, touching upon different stages of the economic cycle. In order for the Mapper to incorporate similar localizations of malicious data

points, future applications of Mapper on cyber data should require more instances of malicious activity.

8.5 Future Work

Future work should build upon the findings and work done in this project and progress towards developing cyber analytics for cyber threat intelligence. One direction for further study would the incorporation of machine learning into the Mapper workflow. In terms of filtering, a model trained using a machine learning algorithm and labeled cyber data can act as a filter function for assigning numerical scores to each data point based on likelihood of malicious intent. Of course, data preparation should entail labeling each data point on whether or not it was a blue team or red team member. Once new indicators of attack have been created from analysis, a new model trained using these indicators would act as part of a cyber detection system. Another idea for future work is developing a better data transformation that would highlight the purpose of each data point and its impact on the network. Although transforming the data into continuous vectors using available cyber analytics was interesting notion to experiment with the Mapper, this transformation was only one way of incorporating meaning into each data point

9 List of References

Australian Cyber Security Centre (ACSC) (2016). *ACSC threat report 2015*. Canberra: Australian Cyber Security Centre. Retrieved from https://www.acsc.gov.au/publications/ACSC_Threat_Report_2015.pdf.

Ayasdi (2016). *The Leader in Topological Data Analysis.* Retrieved from https://www.ayasdi.com/company/

Bayless, R. (2015). *Topological analysis of MOBILIZE Boston data* (Doctoral dissertation, California State Polytechnic University, Pomona).

Bere, M., Bhunu-Shava, F., Gamundani, A., & Nhamu, I. (2015). How Advanced Persistent Threats Exploit Humans. *International Journal of Computer Science Issues (IJCSI)*, *12*(6), 170.

Carlsson, G. (2009). Topology and data. *Bulletin of the American Mathematical Society*, *46*(2), 255-308.

Carlsson, G. (2015, Aug. 31). Topological Modeling and its Use [Web Blog Post]. Retrieved from https://www.ayasdi.com/blog/www-ayasdi-comblogpredictive-analytics/topological-modeling-and-its-uses/

Carlsson, G. (2014). Topological pattern recognition for point cloud data. Acta Numerica, 23, pp 289-368. DOI:10.1017/S0962492914000051

Chintakunta, H., Robinson, M., & Krim, H. (2016, March). Introduction to the special session on Topological Data Analysis, ICASSP 2016. In *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (pp. 6410-6414). IEEE.

Dalziel, H., & Books24x7 IT Pro Collection. (2015). *How to define and build an effective cyber threat intelligence capability*. Waltham, Massachusetts: Syngress.

Dey, T. (2013, August). Computational Topology and Data Analysis [PDF documents]. Retrieved from http://web.cse.ohio-state.edu/~tamaldey/course/CTDA/CTDA.html

Dietrich, D., Heller, B., Yang, B., Books24x7 IT Pro Collection, EMC Education Services, & Safari Books Online. (2015). *Data science & big data analytics: Discovering, analyzing, visualizing and presenting data* (1st ed.). Indianapolis, IN: Wiley.

Dixit, B., & Safari Books Online. (2016). *ElasticSearch essentials* (1st ed.). Birmingham: Packt Publishing, Limited.

Edelsbrunner, H., & Harer, J. (2010). *Computational topology: an introduction*. American Mathematical Soc..

EMC. (2015). *Data science & big data analytics: Discovering, analyzing, visualizing and*

presenting data. (Data science & big data analytics.) Indianapolis, Indiana: Wiley.

Fireeye (2013, February) *APT1: Exposing One of China's Cyber Espionage Units*. Retrieved from https://www.fireeye.com/content/dam/fireeye-www/services/pdfs/mandiant-apt1-report.pdf

Fireeye (2015, July). (APT29) *Hammertoss: Stealthy Tactics Define a Russian Cyber Threat Group*. Retrieved from https://www2.fireeye.com/APT29-HAMMERTOSS-WEB-2015-RPT.html

Ghrist, R. (2008). Barcodes: the persistent topology of data. *Bulletin of the American Mathematical Society*, *45*(1), 61-75.

Haggard, S., & Lindsay, J. R. (2015). *North Korea and the Sony Hack: exporting instability through cyberspace*. East-West Center.

Hegarty, F. (2012). *Computational Homology of cubical and permutahedral complexes* (Doctoral dissertation, National University of Ireland, Galway).

Kim, H. E. (2015). Evaluating Ayasdi's Topological Data Analysis for Big Data. Frankfurt, Germany: Goethe University Frankfurt. Retrieved from http://www.bigdata.uni-frankfurt.de/wp-content/uploads/2015/10/Evaluating-Ayasdi%E2%80%99s-Topological-Data-Analysis-For-Big-Data_HKim2015.pdf

Kraft, R. (2016). Illustrations of Data Analysis Using the Mapper Algorithm and Persistent Homology.

Kuc, R., Rogozinski, M., Hussain, A., Youe, R., Srivastava, S., Siddiqui, S.. Safari Books Online. (2015). *Mastering elasticsearch: Further your knowledge of the elasticsearch server by learning more about its internals, querying, and data handling* (Second ed.). Mumbai, [India]; Birmingham, England; Packt Publishing.

Langner, R. (2011)."Stuxnet: Dissecting a Cyberwarfare Weapon," in *IEEE Security & Privacy*, vol. 9, No. 3, pp. 49-51. DOI: 10.1109/MSP.2011.67. Retrieved from http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5772960&isnumber=5772950

Lum, P. Y., Singh, G., Lehman, A., Ishkanov, T., Vejdemo-Johansson, M., Alagappan, M., ... & Carlsson, G. (2013). Extracting insights from the shape of complex data using topology. *Scientific reports*, *3*.

Marinos, L., Rekleitis, E., Belmonte, M. A., & European Union. (2016 January). *ENISA Threat Landscape 2015*. Heraklion: ENISA. Retrieved from: https://www.enisa.europa.eu/publications/etl2015

Morris, S. A., & University of New England. (1989). *Topology without tears*. Armidale [N.S.W.]: University of New England.

Müllner, Daniel and Babu, Aravindakshan (2013). *Python Mapper: An open-source toolchain for data exploration, analysis and visualization*. Retrieved from http://danifold.net/mapper.

Nicolau, M., Levine, A. J., & Carlsson, G. (2011). Topology based data analysis identifies a subgroup of breast cancers with a unique mutational profile and excellent survival. *Proceedings of the National Academy of Sciences*, *108*(17), 7265-7270.

Ramanan, D. (2016, May 11). Student Finds Shape Has Meaning with Four Decades of Macroeconomic Data [Web Blog Post]. Retrieved from https://www.ayasdi.com/blog/financial-services/student-finds-shape-meaning-four-decades-economic-data/

Singh, G., Mémoli, F., & Carlsson, G. E. (2007, September). Topological Methods for the Analysis of High Dimensional Data Sets and 3D Object Recognition. In *SPBG* (pp. 91-100). Retrieved from https://research.math.osu.edu/tgda/mapperPBG.pdf

Symonds, Jonathan (2015, April). *KDnuggets InterviewL Anthony Bak, Ayasdi on How to Get Started on Topology* [Blog Post]. Retrieved from https://www.ayasdi.com/blog/culture/kdnuggets-interview-anthony-bak-ayasdi-get-started-topology/

Tabish, S. M., Shafiq, M. Z., & Farooq, M. (2009, June). Malware detection using statistical analysis of byte-level file content. In *Proceedings of the ACM SIGKDD Workshop on CyberSecurity and Intelligence Informatics* (pp. 23-31). ACM.

Wang, K. G. (2012). *The Basic Theory of Persistent Homology*. Retrieved from http://math.uchicago.edu/~may/REU2012/REUPapers/WangK.pdf

Wildberger, Norman [njwildberger]. (2012) *Algebraic Topology* [Youtube Lecture series]. Retrieved from https://www.youtube.com/playlist?list=PL6763F57A61FE6FE8.

Wolberg, W. H., Street, W. N., & Mangasarian, O. L. (1994). Machine learning techniques to diagnose breast cancer from image-processed nuclear features of fine needle aspirates. *Cancer letters*, *77*(2), 163-171.

Wolberg, W. H., Street, W. N., Heisey, D. M., & Mangasarian, O. L. (1995). Computer-derived nuclear features distinguish malignant from benign breast cytology. *Human Pathology*, *26*(7), 792-796.

Zomorodian, A. J. (2005). *Topology for computing*. Cambridge, UK: Cambridge University Press.

Appendices

Appendix 1 – Install Python Mapper Guide

Requirements for Python Mapper

- Python 2.6 or higher. The GUI needs Python 2 since it depends on wxPython and PyOpenGL; Python Mapper itself can be run under Python 2 and Python 3. Recommended to install pip if it is not already included in Python distribution. (Without **pip**, replace all instances of **pip** in the instructions with **easy_install**)
- NumPy and SciPy
- Matplotlib
- Graphviz
- Optionally, cmappertools (Mullner's Python module written in C++) that replaces slower Python routines with fast, parallelized C++ algorithms.
- Fastcluster is another Python module written in C++ for fast hierarchical clustering.

Standard installation

Simply type

```
$ pip install mapper --user
```

on a command line. If everything worked, you may stop here and start using Mapper. The steps below describe alternatives and optional steps.

Windows Installation

Here are step-by-step instructions to install Python Mapper and all its dependencies in Windows. If there are errors with rejected connections, look to the Troubleshooting section. Keep in mind that Python Mapper does not contain any platform-specific code and depends only on cross-platform packages.

1. Download and install Python 2.

   Go to https://www.python.org/downloads/windows/. Go to "Latest Python 2 Release". Download the "Windows x86-64 MSI installer". Start the installer and follow the instructions.

2. Download and install wxPython from http://wxpython.org/download.php. Make sure to match your Python version and architecture (32/64bit).
3. Download the Windows NumPy package from http://www.lfd.uci.edu/~gohlke/pythonlibs/#numpy. You'll need the package later. Again, make sure to match your Python version and architecture.

   Note: Download version with 27 if Python 2.7, 34 if Python 3.4

4. Download the Windows SciPy package from http://www.lfd.uci.edu/~gohlke/pythonlibs/#scipy.

   Note: Download version with 27 if Python 2.7, 34 if Python 3.4

5. Download and install Graphviz from http://www.graphviz.org/Download.php.
6. Open a Command Prompt.
7. The directory with the Graphviz executables must be appended to the PATH environment variable. There are lots of tutorials for this step in the internet, look at section below.

   (On my system, the directory is C:\Program Files (x86)\Graphviz2.38\bin.)

8. Close the command prompt and open a new one to process the changes in the PATH variable.
9. Check that the search path is configured correctly by typing:

```
neato -?
```

   If Graphviz's "neato" program responds with a help message, everything is alright.

10. Go to the `Scripts` folder in your Python installation, eg.:

```
cd /d C:\Python27\Scripts
```

11. Type: (If the following command does not work because of SSL certificates or rejected connection, include SSL certificates in C:\Python27\Lib\site-packages\pip_vendor\requests)

```
pip install wheel
```

12. Install the downloaded NumPy and SciPy packages as follows:

```
pip install C:\Users\MYUSERNAME\Downloads\numpy-A.B.C+mkl-cp2X-none-
win_amd64.whl
pip install C:\Users\MYUSERNAME\Downloads\scipy-A.B.C-cp2X-none-
win_amd64.whl
```

13. Install the remaining Python packages:

```
pip install matplotlib
pip install pyopengl
pip install fastcluster
pip install mapper
pip install cmappertools
```

14. The Mapper GUI should work now:

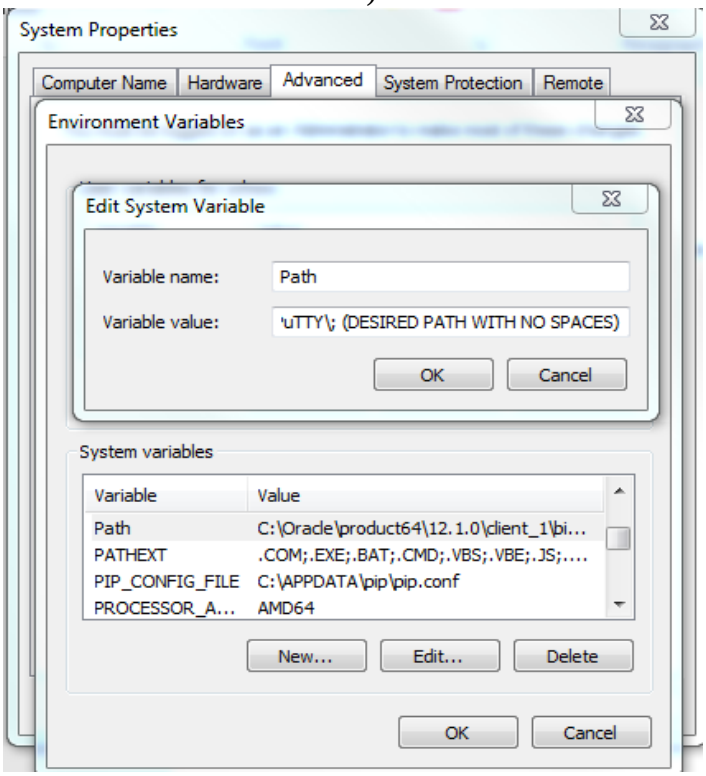
```
C:\Python27\Scripts\MapperGUI.py
```

I recommend to always execute the GUI from the command line. This way, debugging information is displayed on the terminal if an error occurs.

15. *Optional:* You may want to add the Python `Scripts` directory to the search path, in the same way as you did it for Graphviz. On my system, it is `C:\Python27\Scripts`. This way, the Mapper GUI can be started by the command `MapperGUI.py` from any directory.

Editing PATH variable for Windows:

1. Go to My Computer > System Properties > Advanced System Settings > Click on Environment variables
2. Look for Path in System Variables > Select > Click on Edit…
3. To add in another path, append a semi-colon to the last entry and paste in the path to your desired directory (which contains .exe or other file).



Installation: Ubuntu

Note that installation has been tested with Ubuntu 14.04, 15.04, and 16.04.

1. If you have not already done so, go to "System Settings" -> "Software and Update" and activate the item "Community-maintained free and open-sourced software (universe)"

   WARNING: The universe component is a snapshot of the free, open-source, and Linux world. It houses almost every piece of open-source software, all built from a range of public

sources. Canonical does not provide a guarantee of regular security updates for software in the universe component, but will provide these where they are made available by the community. Users should understand the risk inherent in using these packages.

2. Open a terminal and enter the following commands:

```
$ sudo apt-get install python-numpy python-scipy python-matplotlib python-
wxtools \
python-opengl python-pip graphviz libboost-all-dev
$ pip install fastcluster --user
$ pip install mapper --user
$ pip install cmappertools --user
```

3. Optional: Add the directory $HOME/.local/bin to the search path so that the Mapper GUI can be started more conveniently.

   To do so, open the file .bashrc in your home directory, e.g. by

```
$ gedit ~/.bashrc
```

   and insert the following lines at the end:

```
# set PATH so that it includes user's private bin if it exists
if [ -d "$HOME/.local/bin" ] ; then
    PATH="${PATH+$PATH:}$HOME/.local/bin"
fi
```

   Now save the file and type

```
$ source ~/.bashrc
```

   in the terminal window to read the modified file in.

4. Try Mapper out by typing **MapperGUI.py** into the command line.

Mac OS

- There are no installation instructions for Mac OS as of October 2016
- You can find installation tips at: http://danifold.net/mapper/index.html

Appendix 2 – User Guide for TDA System

About

The Mapper algorithm is a method for topological data analysis invented by Gurjeet Singh, Facundo Mémoli and Gunnar Carlsson. The Mapper method is the key part of a processing chain with (minimally) filter functions, the Mapper algorithm itself and visualization of the results. Python Mapper is a realization of this toolchain, written by Daniel Müllner and Aravindakshan Babu. It is open source software and is released under the GNU GPLv3 license.

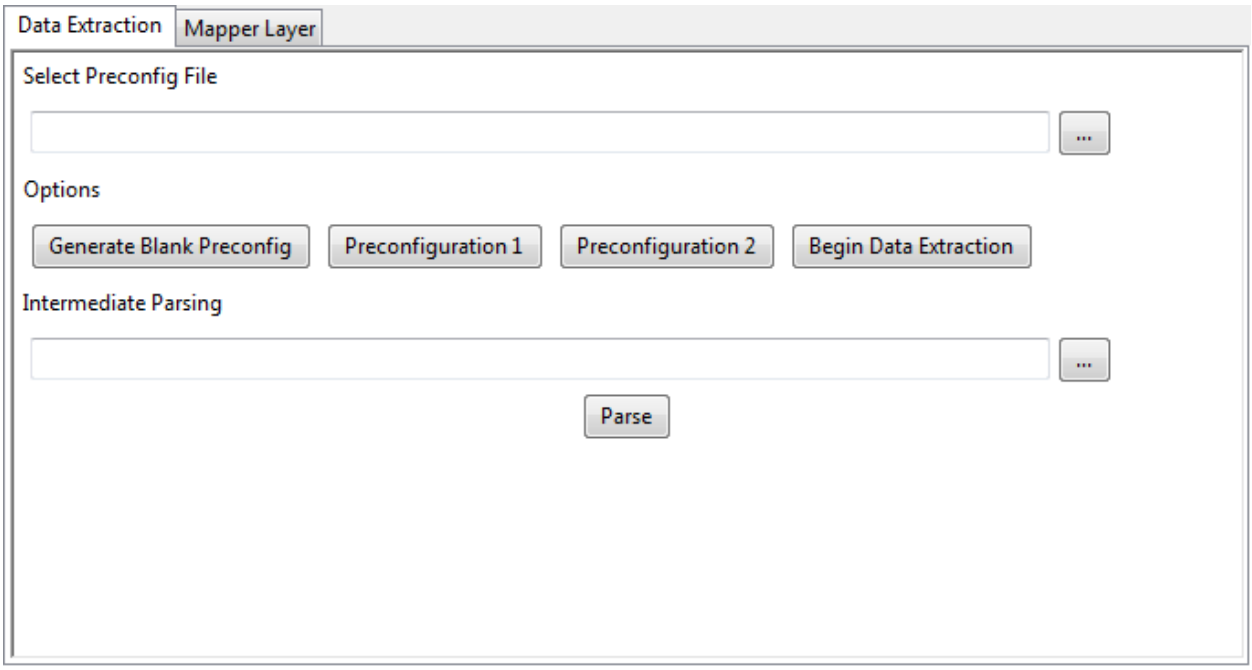
Elastic Mapper is a tool that incorporates automation to the Python Mapper. First, Elastic Mapper allows the user to pull data directly from a cloud storage service and parses the data into a readable-format for the Python Mapper. Then, Elastic Mapper outputs a visualization of the data where the user can interact with the nodes of the visualization; this interactivity is inherited from Python Mapper.

Requirements

- Python 2.7 or greater
- Installation of Python Mapper allows this system to user Mapper module and wxPython module
- Directories that should be Elastic Mapper's working directory are *Nodelist*, *DATA*, *ES_preconfig*, and *Mapper_preconfig*
- Python Mapper only accepts .gz files of CSV-formatted files (values within CSV files must all be float data types)

Begin invoking the application by either double-clicking TDA_Sys_GUI.py or running TDA_Sys_GUI.py in the same directory with all the source code files.

Data Extraction



Starting from Scratch

1. In order to pull from the Elasticsearch cluster, either press on one of the pre-configurations or press the Browser button to select another configuration for the parameters. User can also generate a blank configuration file and replace Null keywords with desired input.

```
{
    "index": null,
    "http": null,
    "format": null,
    "start_time": null,
    "end_time": null,
    "port": null
}
```

The parameters in the configuration files are parameters used to select the desired Elasticsearch server, desired cluster, and information used for querying results.

Http – The node in Elasticsearch cluster to connect to.

Port – port number of the host

Index – describes further where you are pulling data from

Format – format you want the data in before preparing the data (Currently does not matter)

Start_time – Beginning of the time interval of the data

End_time – End of the time interval

Is_continuous – Boolean variable for determining if you want the data parsed into continuous values

Is_categorical – Boolean variable for determining if you want the data parsed into categorical but numerical values (Enumerating through all values and assigning a number)]

2. Once configurations have been selected, press the *Begin Data Extraction* to begin pulling data from the cloud and storing all the data in CSV-formatted files.

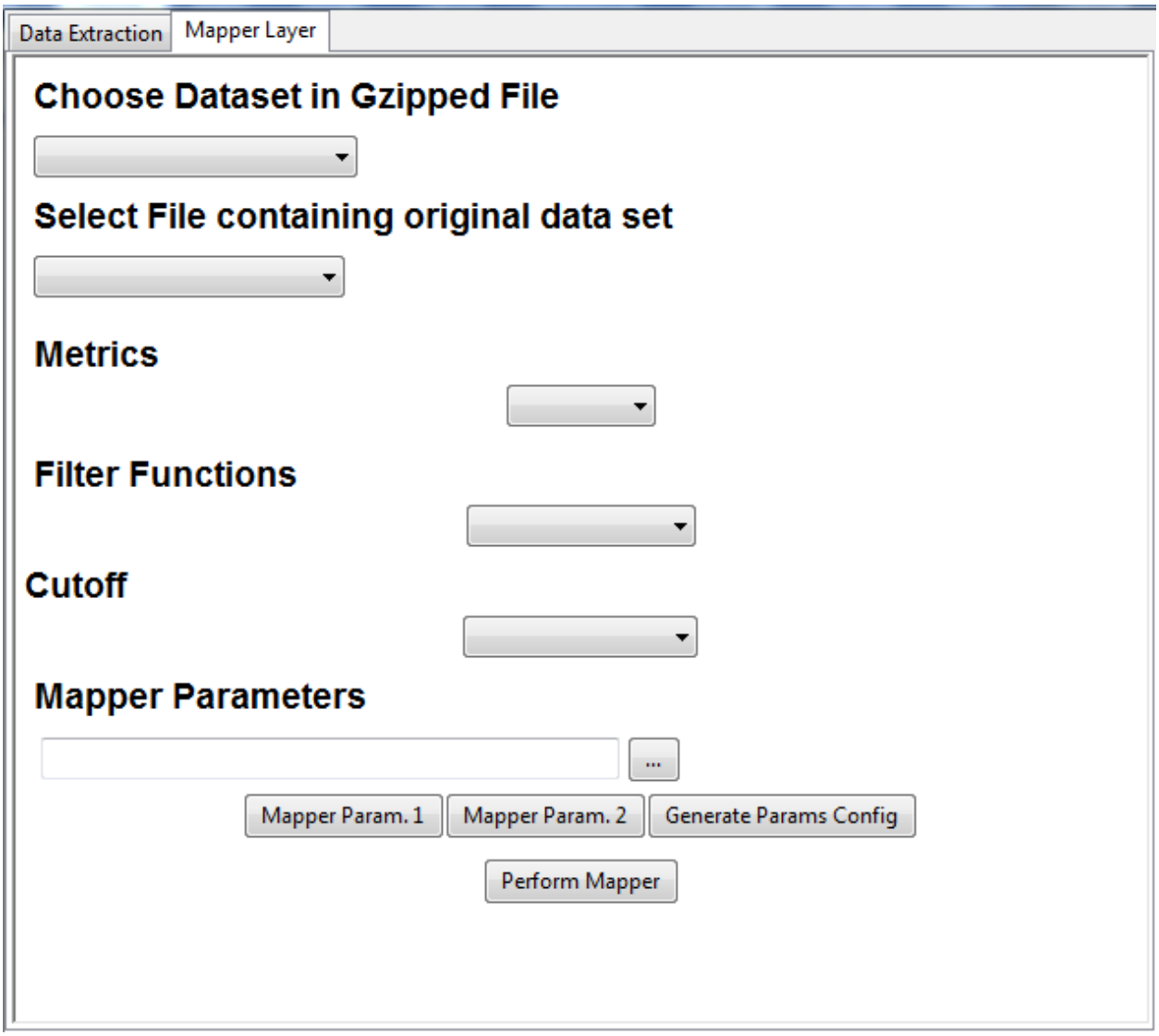
Beginning with clean data

1. If you already have cleaned data set in CSV-formatted file (if in different format, be sure to define your own Parser object in the user_defined.py file), select the file by pressing the Browser button in the Immediate Parsing section.

   Important: Make sure that data file is in DATA directory

2. Press Parse and the resulting file can be found in DATA directory.

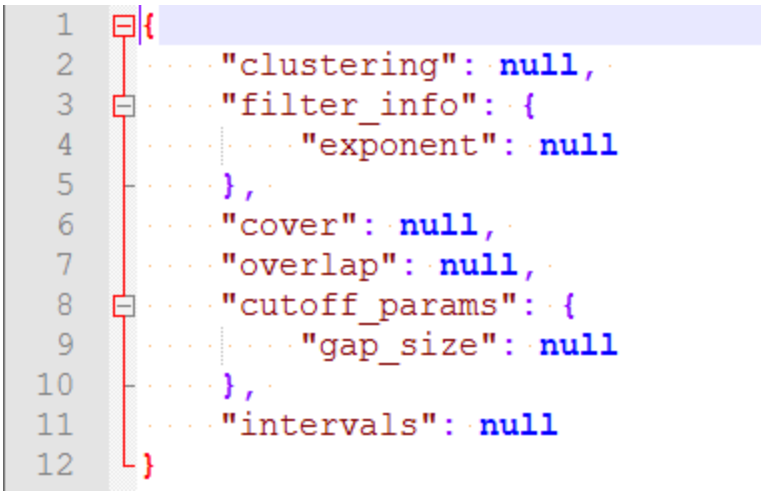
Mapper Layer



Starting from Scratch

1. Select the Gzipped file of your parsed data.
2. Select the original version of your dataset (The exact same dataset before parsing or cleaning). Otherwise, select a version of the dataset that provides you with the most meaning. For example, you can output node lists that contain the indices of the data points corresponding to the row in the CSV-file.
3. Select metric, filter function, cutoff algorithm, and then press the Generate Params Config button.

4. Find the blank configuration file in the Mapper_preconfig directory and fill in the fields. The parameters that appear in this file depends on your choice in filter, metric, and clustering algorithm.



Clustering – clustering algorithm for partial clustering in Mapper method

Filter_info – filter function arguments

Cover – Choice for various covers of the filter range. Choices include cube_cover_primitive and balanced_cover_1d.

Overlap – The percentage of overlap between intervals

Cutoff_params – Parameters for the cutoff algorithms

Intervals – Number of intervals to partition the range, controls resolution.

5. Select the new configuration file by pressing the Browser button under the Mapper Parameters Section
6. Press *Perform Mapper* button to begin the process of running the Mapper algorithm and outputting a visualization.

Quick Start

1. Press either *Mapper Param 1 or Mapper Param 2* to select preconfigured parameters. You may also use these pre-configurations as examples. *Mapper Param 1* locks in the Euclidean metric, Eccentricity filter, and Big Gap 2 cutoff. *Mapper Param 2* fixes the same parameters to the same values except for choice in filters; *Mapper Param 2* fixes its filter to the Singular Value Decomposition function.

```
{
    "cover": "balanced",
    "clustering": "single linkage",
    "intervals": "15",
    "overlap": "50",
    "filter_info": {"exponent": "1.0"},
    "cutoff_params": {"exponent": "0.0", "max clusters": "50"}
}
```

```
{
    "cover": "balanced",
    "clustering": "average",
    "intervals": "50",
    "overlap": "50",
    "filter_info": {"mean center": "True", "order": "1.0"},
    "cutoff_params": {"exponent": "0.0", "max clusters": "50"}
}
```

Figure 28: Mapper Parameter Configurations, Mapper Param 1(Top) and Mapper Param 2 (Bottom)

2. Press *Perform Mapper button.*

Analysis

After beginning the Mapper algorithm, the system carries out two tasks. First, the system stores CSV-formatted files composed of data points for all nodes in the visualization; the user can look at the data points residing in a specific node. In the Mapper Output Window, the user can also select a single node or a group of nodes and save the corresponding node list to some directory.

Appendix 3 – ATT&CK Model

Persistence	Privilege Escalation	Defense Evasion	Credential Access	Discovery	Lateral Movement	Execution	Collection	Exfiltration
Accessibility Features	Accessibility Features	Binary Padding	Brute Force	Account Discovery	Application Deployment Software	Command-Line Interface	Automated Collection	Automated Exfiltration
AppInit DLLs	AppInit DLLs	Bypass User Account Control	Credential Dumping	Application Window Discovery	Exploitation of Vulnerability	Execution through API	Clipboard Data	Data Compressed
Basic Input/Output System	Bypass User Account Control	Code Signing	Credential Manipulation	File and Directory Discovery	Logon Scripts	Graphical User Interface	Data Staged	Data Encrypted
Bootkit	DLL Injection	Component Firmware	Credentials in Files	Local Network Configuration Discovery	Pass the Hash	InstallUtil	Data from Local System	Data Transfer Size Limits
Change Default File Association	DLL Search Order Hijacking	Component Object Model Hijacking	Exploitation of Vulnerability	Local Network Connections Discovery	Pass the Ticket	PowerShell	Data from Network Shared Drive	Exfiltration Over Alternative Protocol
Component Firmware	Exploitation of Vulnerability	DLL Injection	Input Capture	Network Service Scanning	Remote Desktop Protocol	Process Hollowing	Data from Removable Media	Exfiltration Over Command and Control Channel
Component Object Model Hijacking	Legitimate Credentials	DLL Search Order Hijacking	Network Sniffing	Peripheral Device Discovery	Remote File Copy	Regsvcs/Regasm	Email Collection	Exfiltration Over Other Network Medium
DLL Search Order Hijacking	Local Port Monitor	DLL Side-Loading	Two-Factor Authentication Interception	Permission Groups Discovery	Remote Services	Regsvr32	Input Capture	Exfiltration Over Physical Medium

Hypervisor	New Service	Disabling Security Tools		Process Discovery	Replication Through Removable Media	Rundll32	Screen Capture	Scheduled Transfer
Legitimate Credentials	Path Interception	Exploitation of Vulnerability		Query Registry	Shared Webroot	Scheduled Task		
Local Port Monitor	Scheduled Task	File Deletion		Remote System Discovery	Taint Shared Content	Scripting		
Logon Scripts	Service File Permissions Weakness	File System Logical Offsets		Security Software Discovery	Third-party Software	Service Execution		
Modify Existing Service	Service Registry Permissions Weakness	Indicator Blocking		System Information Discovery	Windows Admin Shares	Third-party Software		
New Service	Web Shell	Indicator Removal from Tools		System Owner/User Discovery	Windows Remote Management	Windows Management Instrumentation		
Path Interception		Indicator Removal on Host		System Service Discovery		Windows Remote Management		
Redundant Access		InstallUtil						
Registry Run Keys / Start Folder		Legitimate Credentials						
Scheduled Task		Masquerading						
Security Support Provider		Modify Registry						

Service File Permissions Weakness		NTFS Extended Attributes						
Service Registry Permissions Weakness		Obfuscated Files or Information						
Shortcut Modification		Process Hollowing						
Web Shell		Redundant Access						
Windows Management Instrumentation Event Subscription		Regsvcs/Regasm						
Winlogon Helper DLL		Regsvr32						
		Rootkit						
		Rundll32						
		Scripting						
		Software Packing						
		Timestomp						