Big Data for Savings Groups: A Study and Application of Data Visualization

Maria Bolt

Point Loma Nazarene University

Table of Contents

Abstract

Big data, despite the inherent challenges it poses, has become more and more influential in recent years due to the ever-growing prominence of digital data as well as the constantly improving data management technologies. It has become a powerful tool for analysis in a wide range of industries and fields, even including applications to international development. However, the ability to gain insight from big data is dependent upon good data visualization. Similarly, the savings group model, an implementation of microfinance in developing countries, has risen in popularity in recent decades. This model has been successful by emphasizing local empowerment, but further advancement in the field is limited by the difficulty of data collection. In this study, it is argued that the application of big data analysis to savings group data could provide the needed support for the continuous improvement of the savings group model. Also, the DreamSave mobile application for savings groups, created by DreamStart Labs, provides a platform from which to begin synthesizing these fields. This research includes a theoretical examination of the aforementioned concepts, as well as a practical data visualization project using data from the DreamSave app. Finally, the study documents the challenges and lessons involved in the realization of the practical project, highlighting how they reflect common issues faced in the data-driven software development industry.

Keywords: big data, data visualization, savings groups, VSLA

Introduction

This research was inspired by the opportunity to begin building a bridge between two important fields in a way that has never been done before. Firstly, the field of big data is a rapidly evolving topic, with a scope of influence that continues to expand and touch new areas. Likewise, the study and implementation of savings groups is rapidly becoming popular in the realm of international development. The highly technological world in which big data lives is not typically a consideration in the space of international development; however, the recent creation of the DreamSave mobile application has provided a means for that to change. Thus, the purpose of this study is to take the initial steps in exploring this new relationship. First, a literature review is conducted to examine the fields of big data and of savings groups individually, explaining their history, growth, challenges, and considerations. These concepts lead to a discussion of how, thanks to DreamSave, the tools and concepts of big data can be utilized to support advancement the field of savings groups in a unique way. This theoretical study is then supported by a practical application, which, though limited in scope, attempts to visualize DreamSave app data in meaningful ways. Finally, the challenges and lessons learned in creating this exploratory project are documented, serving as an example of what it looks like to work through the complications of bringing theory into practice in this realm. Therefore, all of the work included in this research is part of an effort to explore what it looks like to synthesize the digital world of big data with the continuous improvement of savings groups.

Part 1: Conceptual Literature Review

Big Data and Visualization

A handful of years ago, the term “big data” began to be identified as a buzz word: it emerged as a popular topic of conversation, an intriguing area of research, and an important

business target. Now, in 2019, “big data” is no longer just a buzz word—it is a well-developed and established phenomenon that has implications in a wide range of commercial and scholarly fields. In fact, a search for the term “big data” in the Web of Science Core Collection database in 2015 returned an impressive 3347 results, and just a year later returned over 4000 results (Batistič & van der Laken, 2019). The two main reasons for this rapid growth are simple: first, the fact that every day we have more data in the world than ever before, and second, the fact that we continue to develop better and better ways to store and analyze that data (Marr, 2016). These two realities are what launch the field of big data into a more and more prominent place in business and academic efforts.

Even with such a rise in attention towards the topic of big data in recent years, the term itself has been tricky to define (unsurprisingly, as the word “big” is quite ambiguous). What exactly is meant by “big data” is usually something slightly different in each context. However, scholars generally agree that the essence of big data can be understood in terms of what is known as the “three V’s”: volume, velocity, and variety. Each of these describe a specific challenge that is inherent to big data (Simon, 2014). “Volume” implies that the data in question is in such a large quantity that it exceeds the limits of traditional data servers. Next, “velocity” refers to the ever-increasing speed at which data is being accumulated, and “variety” refers to the many distinct facets and forms of modern data that one will find. Many authors have gone even beyond the core three by identifying more “V’s” that describe challenges of big data, which include veracity (accuracy), value, validity, vulnerability, volatility, and more. (Firican, 2017). They key takeaway from these “V” characteristics is that big data is set apart because it cannot be sufficiently managed by the single-server relational databases that have become commonplace in the data world. Instead, innovative storage and analysis solutions are required in order to handle

the problems that come with the incredible size, speed, and diversity of data. In the words of Brath and Jonker (2015), “Big data is not strictly defined by how big it is, but by the fact that it is large and complex enough that it defies management and analysis using traditional systems and approaches” (p. 419). It is for these reasons that big data is sometimes referred to as the “big data problem” (Landset et. al., 2015).

The specific volume of big data in the world is impossible to nail down, and it is growing constantly. With the extreme digitization that this generation has seen, from smartphones to commercial software to social media and everything in between, it is no surprise that the amount of data we are generating quickly accumulates to numbers that are difficult to comprehend. For example, according to a 2014 statistic from the software company Asigra, the amount of data generated by regular business/consumer life was 2.5 quintillion bytes per day—and no doubt the rate has increased since that then (Simon, 2014). It was also predicted that by the year 2020, there will be over 50 billion devices in the world connected to the internet, the rate of new data creation will be over 1.7 MB/second, and the total amount of data in the world will reach 44 zettabytes (4.4×10^{22} bytes) (Marr, 2016; Landset et. al., 2015). As notable author and data scholar Phil Simon writes, “The Data Deluge has arrived, and it’s not going anywhere” (p. 19). It is clear that digital data has become our new reality, and it only promises to grow with time.

The Transition into Big Data

The data world has achieved the size it is today as a result of the technological advancements that have provided both an inexhaustible source of data and more efficient ways to manage it. Before the rise of the information age (the last about 30 years), data collection and analysis was reserved for targeted research efforts, simply due to the fact that it required significant time and energy to extract it. Now, however, we have undergone a transition in which

the vast majority of the data in the world is generated as a byproduct of the digitization of our everyday lives: for example, in an average minute in 2012, Twitter users shared about 100,000 tweets, there were about 700,000 posts on Facebook, and Google saw about 2,000,000 search terms (Hilbert, 2015). None of these sites were built for the purpose of gathering data for research; nevertheless, they generate huge quantities of information to be tapped. The same can be said for the countless other technologies that have arisen in the recent decades.

Therefore, data storage and analysis frameworks have had to evolve and improve in order to both keep up with the scale of data being produced, as well as to allow it to continue to grow. The rise of the big data movement has been made possible in the past 30 years due to huge advances in centralized and decentralized data storage systems, telecommunication bandwidth over networks, and digital computation capacities (Hilbert, 2015). For example, before the days of the internet, most organizations transported their digital data using a process called extract, transform, load (ETL). As the name suggests, this process consists of extracting data from the source system(s), transforming it into a format that is conducive to the target system, and finally, loading it into the target system (usually, a data warehouse). Although ETL is still relevant, it is steadily being replaced by the use of APIs, which first came onto the computing scene in the 1980s (Simon, 2014, Ruecker et. al., 2015). An Application Programming Interface (API) is a defined set of protocols which allow an application's data and functionality to be programmatically accessed from the outside. This serves to enable collaboration and communication between different organizations, as well as to allow individual organizations to practice "separation of concerns": dividing a program's code into distinct bodies which communicate between each other via APIs (Ruecker et. al., 2015). The documentation of the available functions of an API is what programmers can use know how to interact with the data

and services of that particular program. Thus, using a database API (or creating a new one) has the potential to add an extra layer of protection, optimization, simplicity, and control to working the with a dataset. APIs are more efficient for data access and handling large amounts of data, they are easier to debug, and because they can be open-source (released for public use), they help foster a culture of collaboration and continuous improvement (Simon, 2014).

The adoption of APIs is just one example of how technology frameworks have evolved over time to keep up with the growing amounts of data collected. A few of the other important advancements have been the advent of parallel processing, which allows a set of data to be processed in a fraction of the original amount of time, and the rise of cloud computing, which has made big data storage and analysis accessible by eliminating the need for organizations to purchase their own hardware (Landset et. al. 2015, Marr, 2016). There is not time nor space here to discuss all of these things in depth; however, a few specific examples of technologies that have responded to the big data problem will be analyzed later in this paper.

NoSQL for Big Data

With the rate at which modern data is accumulating, it is of utmost importance in the computer science world to come up with solutions to store and access it efficiently and reliably. With the volume, velocity and variety of big data, it has become clear to database engineers that the traditional relational databases do not even begin to have the capacity to store (and much less analyze) that data (Simon, 2014). Therefore, to solve this problem, there was a need to develop a new option for database structure—that is where NoSQL comes in. The concept of NoSQL databases has brought about large strides in the information technology landscape over the recent years. The following will attempt to explain how NoSQL differs from traditional, SQL-based relational databases in design, function, and capacity.

Before describing the characteristics of NoSQL, it is important to have a basic understanding of what exactly constitutes a SQL database. SQL stands for structured query language, which is a data definition and manipulation language that is used for what are known as “relational” databases. The schema of a relational database consists of logical tables, where data entity instances are understood as rows, and the columns represent different attributes. Within the relational structure, unique keys are used to identify data entities, and corresponding keys relate entities to each other across different tables. This strict, pre-defined structure of a relational database makes it simple to understand and work with data. Also, SQL is a powerful, versatile, and widely-used language that makes it easy to write complex queries (Smallcombe, 2019). However, this strict structure does not always work for the modern forms of big data that exist today. In addition, because SQL was invented in the early days of databases, it was designed for making the most out of resources such as storage and processor power, but this is much less of a concern today, as those things are now much cheaper and more accessible (Madison et. al., 2015). Relational databases operate on a single server, and in order to increase the capacity of the database, it is necessary to use vertical scaling methods of adding additional CPU power, RAM, etc. to the current hardware. A few popular examples of SQL database management systems include Oracle, MySQL, and Microsoft SQL Server (Smallcombe, 2019).

In contrast, NoSQL, or “not only SQL,” refers to databases that do not use a traditional relational structure. The concept was first introduced in 1998 and began to grow in popularity starting in the 2000s. As larger, web-based applications became much more prominent, it became clear that there was a need for better solution for storing and processing the wide diversity of big data than SQL could offer. According to Madison et. al. (2015), “Social networking, big data and business intelligence applications, for instance, all pushed traditional relational databases to their

limits, thus helping to spur the creation of non-relational, horizontally scalable, distributed databases" (p. 5). Admittedly, NoSQL was not meant to eliminate the need for relational databases, but rather to provide another alternative structure to use (often along with a relational database) in order to maximize the benefits of both flavors (Madison et. al., 2015). Popular examples of NoSQL databases include MongoDB, Apache HBase, Apache Cassandra, CouchDB, and more (Smallcombe, 2019).

There are two key strengths of NoSQL databases that sets them apart from relational databases and makes them conducive to handing big data. Firstly, they are able to work with many different structures of data. To explain, the NoSQL category actually encompasses a range of different architectures, including key-value stores, document-based, graph-based, and wide-column stores (Smallcombe, 2019). This is incredibly valuable for handling the varying nature of big data, and it allows database engineers to make a decision on what type of schema to use based on the unique needs of their data. Secondly, NoSQL databases are designed for a distributed system, using a cluster of hardware. This enables what is called horizontal scaling, which refers to the ability to add new machines to a system, both to increase storage space and help ease the workload by sharing it between multiple machines (Sharma, 2019). This is crucial for big data, because it means that if a database system is starting to become overwhelmed by the quantity of data, it is fairly simple and inexpensive to add more machines into the cluster. Vertical scaling, on the other hand, is usually much more costly, because it is difficult to add extra resources to an existing machine.

Of course, NoSQL also has its drawbacks. Firstly, unlike with SQL, there is not one simple, powerful query language that works universally across the distinct types. Depending upon the database management system, there will be different syntax involved in reading,

writing, and transforming the data, and there are usually not as many functions available as there are with SQL (Madison et. al., 2015). Often times, a new NoSQL database will be developed by an organization realizing that none of the available products suit their needs, so they decide to create their own—but this practice has resulted in a lot of fragmentation in the world of NoSQL (Landset et. al., 2015). Some efforts have been made to create a more general query language for NoSQL databases, and enable communication across platforms, but to this date there is still nothing that comes close to being universal (Madison et. al., 2015). Another drawback of the distributed cluster system is that it does not maintain the level of data consistency that a single-server system does. Distributed databases often have strong fault-tolerance capacities built into their schemas, in which there is a system of data replication across the nodes (machines) in a cluster. This is generally a good thing, because should any one node fail, the data would not be lost. However, it also leads to a compromise in data consistency, because when an update is written to one node, it may take some time for it to be replicated to the other applicable nodes. According to Madison et. al. (2015), "NoSQL systems tend to relax requirements on consistency in order to maximize availability and partition tolerance, thus settling for eventual consistency rather than strong consistency" (p. 3). Thus, some authors have pointed out that while SQL databases conform to the "ACID" standards for data (atomicity, consistency, isolation and durability), NoSQL standards fit better into the "BASE" acronym (basically-available, soft-state, eventually-consistent) (Madison et. al., 2015). Therefore, NoSQL is not an objectively better model for database design than SQL, but rather is a compromise of values which favors the unique needs of big data.

Significance of Big Data

Big data has not become such a buzz topic simply because of the exponential growth in worldwide digital data, nor because of the way that the technological environment has evolved along with it. The essential intrigue about big data lies in its potential to be harnessed for learning, improvement, and profit in all kinds of distinct fields. "Independent from the specific peta-, exa- or zettabytes scale," Hilbert (2015) claims, "the key feature of the paradigmatic change is that analytic treatment of data is systematically placed at the forefront of intelligent decision-making" (p. 138). The majority of the study that has been done about big data is concerned with how it can be analyzed to draw out insights that translate into organizational value. Evidence has shown that in general, participation in big data analytics makes organizations more future-oriented, adds business value, and stimulates data-driven decisions, which are often more fruitful than intuition or experience-driven decisions (Batistič & van der Laken, 2019). As such, there are three principal characteristics of big data that make it a revolutionary source of information. One, as previously established, is that it is generated largely without effort and in large quantities. Secondly, it is often available in real-time, so there is potential to react immediately to trends in a way that has never been possible before. Finally, it potentially eliminates the need for statistical sampling, because it is capable of representing an *entire* population (Hilbert, 2015). Of course, this statement needs to be made with caution, as will be addressed later; however, it is fair to say that the practical limitations that previously required the use of sampling are effectively eliminated with big data technology.

To state it simply, in the words of Bernard Marr (2016), "No matter what job you are in and no matter what industry you work in, Big Data will transform it" (p. 1). Marr, in his book *Big Data in Practice* (2016), goes on to describe case studies of 45 different companies that have

found ways to utilize big data analytics to enhance their business. Chapter titles range from "Netflix: How Netflix Used Big Data To Give Us The Programs We Want" to "ZSL: Big Data In The Zoo And To Protect Animals" to "The US Government: Using Big Data To Run A Country." By providing a survey of some of the best examples of big data in action, Marr effectively demonstrates how big data can be leveraged in almost any context to provide real benefits. Clearly, more and more organizations are recognizing the value in big data analytics and are choosing to invest in it.

Big Data for Development

The big data revolution of the past 20-30 years has seen applications in a large variety of different business and research-related fields: finance, supply-chain, marketing, entertainment, social science, behavior analysis, and the list goes on. More surprisingly, there are also instances of big data studies being applied to the field of international development—denominated appropriately as "big data for development" (BD4D) (Hilbert, 2015). The first literature about BD4D was published in 2007, and the concept has been applied in various efforts, examples of which include predicting poverty, measuring economic growth, mapping population distribution, and analyzing housing prices over time (Albanna & Heeks, 2018). BD4D could not exist were it not for the fact that the explosion of internet information exchange over the past decades has happened in both developed *and* developing countries. For example, the four countries with the highest amount of Facebook users in 2013 were India, Brazil, Indonesia and Mexico. Also, in 2011, Kuwait and Brunei had more Twitter users per capita than both the United States and the United Kingdom (Hilbert, 2015). The digital revolution has become so widespread that it is not confined solely to areas of wealth; in fact, just like in first-world countries, there is constant real-time data that is flowing in from developing countries that could potentially offer insight into

how these populations function. For instance, big data analytics using Twitter language alone has been able to predict cultural identities, migration patterns, and tourism mobility in places such as Malaysia and the Philippines. It has been suggested that by using big data merged from different sources, it would be possible to gather insight about indicators such as well-being, stress, fear, optimism, trust, etc. of developing populations. This has the potential to offer a far more intricate picture than the basic life expectancy, literacy, etc. statistics that the Human Development Index provides (Hilbert, 2015).

Be that as it may, the uptake of big data in the field of development has been slower than in other areas, and it has yet to become well-established, and for good reason. As Hilbert explains, "The prerequisites for making Big Data Analytics work for development are a solid technological (hardware) infrastructure, generic software services and human capacities and skills" (p. 140). In developing countries, these prerequisites are not always present. When working with the big data that is available from developing countries, the danger of misrepresenting a population due to unequal access to technology is amplified. Still, in 2014, mobile phones had penetrated 95% of the globe as a whole, including 75% the places where people make $1/day or less (Hilbert, 2015). Statistics like these demonstrate that technology has reached more people than many of us would imagine. Therefore, even though representational validity should be given special care when applying big data to development, it is not so grave a problem that should invalidate the effort.

Two other important challenges to consider when working with BD4D are the absence of theory and proof of concept skew. "Absence of theory" refers to the fact that predictive analytics using big data are often not based on any kind of theory, but rather they simply seek to identify patterns and trends. However, it will be necessary to have some kind of theoretical framework if

the goal is to identify how variations in inputs cause indicators to change. “Proof of concept skew,” on the other hand, has to do with analytic research efforts being overly theoretical, and mainly removed from any practical applications to decisions. In order to make big data work for development, the insights have to have practical value that can be employed by development efforts to produce positive change (Albanna & Heeks, 2018).

One interesting potential area of application of BD4D, identified and discussed by Albanna and Heeks (2018), is the study of positive deviance. Positive deviance (PD) is a development approach which looks for solutions to issues based on the knowledge asset of “positive deviant” individuals: people who have been more successful than their peers because they do things differently. Put simply, the goal is to identify individuals that have thrived against the odds, figure out what enables their success, and then try to spread that technique to others so that they can be successful as well. It is an approach that was first formally used in 1976 but did not gain real traction until the 1990s. Since then, according to the Positive Deviance Initiative, PD has been successfully implemented as a strategy in over 60 countries and has reached more than 30 million people worldwide. An important factor about the PD premise is that because it is based on learning from individuals who have found their own ways to thrive, it naturally reduces the amount of external reliance in development, focusing instead on empowering local models and resources (Albanna & Heeks, 2018). However, the PD approach is not without its limitations, the biggest of these being the difficulty in bringing it from theory into practice. It costs considerable effort to gather enough data in order to confidently label someone as a positive deviant, and even then, it can be difficult to single out with statistical significance the factors that make them thrive. In the words of Albanna and Heeks, “PD tends to rely on in-depth primary data collection in identifying PDs and then community mobilization in disseminating

and scaling successful practices. Identification is therefore time and labor intensive, with costs proportional to sample size" (p. 2). Thus, limitations on cost and time can have a negative effect on the sample size of the study, which in turn makes the task of identifying positive deviants and the practices that enable their success more difficult. That is where big data comes in. Albanna and Heeks propose that the application of big data to positive deviance (BDPD) offers a solution to some of the current challenges that this strategy faces, because harnessing the abundance of real-time, dynamic, digital data would greatly reduce the costly PD identification process. In their words:

> Despite the effectiveness of PD as a problem-solving approach for international development, its uptake by developing countries in domains other than public health has been very limited, and its application has been concentrated in rural areas of just a few countries…Big data can thus expand the scope of PD, enabling it to break from its current path dependency. (p. 16)

Thus, the study of positive deviance would be one way in which big data for development concepts could be applied to produce fruitful results (Albanna & Heeks, 2018).

Data Visualization

Just as the influx of digital data has driven the improvement of data management frameworks, it has also led directly into the growth of the related but distinct field of data visualization. In contrast to big data, data visualization is a term that is fairly straightforward to define—Valkanova et. al. (2015) describe it as "the creation and study of interactive and graphical representations intended to make sense of data" (p. 5). (To clarify, data visualization can refer to this practice of creating graphical representations, but can also refer to an individual data visualization, which is specific information that has been abstracted in some sort of graphic

schema.) As the amount of data being collected and stored in the world has grown, so has the motivation to create visualization solutions. Especially when it comes to the overwhelming size of big data, visualization does not simply aid in discovering new insights, but it is absolutely necessary in order to have any understanding of the data as a whole (Simon, 2014).

Visual representations have incredible power to transform our perception of data, and by extension, truth. Scientific study has proven that human brains are wired to understand by seeing—in fact, our eyes take in more information than all of our other senses combined. Therefore, a simple formatting of data can help our brains to easily draw out patterns and insights (Simon, 2014). Consider this illustration described by Brath and Jonker: back when graphics processing computers were first emerging, they presented a slide containing a table of numbers to a group of Fortune 500 executives and challenged them to find a pattern. After the executives struggled to do so, the presenters showed them a slide with the same numbers, but this time displayed a line graph. Suddenly, the patterns were easy to see (Brath & Jonker, 2015). This is a simple story, but all the same—given visual input, the human brain has incredible capacity to discern patterns, trends, relationships, and outliers. In addition, seeing visualizations can aid in memorization and the grasp of abstract ideas (Myatt and Johnson, 2011). The concept of creating graphical representations to aid and shape our understanding of information is the foundation of the efforts to make sense of big data using visualization.

Building off of that truth, an important next step is to realize that data visualization not only assists in comprehension, but it also assists in *convincing*. For example, in the book *Factfulness*, Hans Rosling discusses how many first-world citizens have wrong ideas about the current state of the developing world (we tend to think that things are worse than they really are). Drawing on his experience from a life dedicated to communicating the actual facts about the

world to people, Rosling expresses that when people are told the data, and it goes against their conception of the world, they often refuse to believe it. However, when they are shown the data, their ideas start to change (Rosling et. al., 2018). Also, in order for data analysis to actually have a positive impact on decision-making, it is almost never enough for just one person to understand the data—the insights need to be shared and agreed upon, and visualization is the most effective means of accomplishing that. As Abramson and Dohan (2015) explain, "providing access to data and analysis in this way helps readers see patterns, understand the analyst's interpretations, evaluate reliability, and gain a sense of an argument's scope and grounding" (p. 300). Thus, visualizations that are built for collaboration are the key to motivating data-driven judgements, which then lead to actions. As summarized by Simon (2014), the short-term goal of data visualization is to communicate information, but the long-term goal is to inform better decisions.

The need to visualize data naturally leads to the question of *how*. The strategy of selecting, designing, and implementing graphical representations of data is a world of research in itself—and people are consistently coming up with new and creative ways to do it. The theory of data visualization design is critical when it comes to taking information from a raw dataset and graphing it effectively. Just as a well-designed graph will greatly improve understanding, a poorly designed graph will, at best, fail to communicate any valuable information, or at worst, confuse or mislead the viewer. Consider, for example, a statistical phenomenon entitled Simpson's Paradox. Simpson's Paradox describes the situation in which a trend that is clear to see in several related groups of data individually is suddenly reversed when those groups are displayed together. To illustrate: imagine three side-by-side scatterplots representing data from three different companies, and they each show a negative correlation between the X and Y variables. Now, if those three companies are grouped together and their data points are plotted on

the same graph, the way that the company data points stack on top of each other makes the correlation between X and Y appear to be positive. In this case, the company is a confounding variable, making the correlation between X and Y look like the opposite of what it truly is (Holtz & Healy, 2018). Simpson's Paradox is just one example of how data visualization design choices can greatly influence the overall message that is communicated to the viewer. Therefore, it is incredibly important for those who make visualizations to consider how they can display the data truthfully, without bias, distraction, excessive complexity, or misleading designs.

When beginning the process of creating a visualization of data, an important first step is to identify the desired question which the data can help to answer. Especially in a world dominated by multi-variate, diverse and unstructured big data, starting by asking a specific question provides essential focus for the graphic (Yau, 2018). Once a question is identified, as well as the variables of the dataset that are applicable to answering that question, the next step is to determine the nature and classification of that data. Data can be classified in a number of different ways, depending upon both its primitive type and its context. On a basic level, data can be either be categorical (qualitative) or numeric (quantitative). To take it one step further, categorical data can be measured using either a nominal or an ordinal scale. The nominal scale says that there is no implied ordering of the different categories; for example, comparing the number male and female students at a given school. In contrast, the ordinal scale is used when categories do have an implied ranking, such as comparing restaurants that are one, two, three, four or five-star. Another distinction that can be made is whether the data is continuous, meaning it can take any value within a range of values, or discrete, meaning it is the result of counting and expressed in whole numbers only (Sahay, 2017). Determining how to classify the data appropriately is what enables the selection of a visualization design that will be the most fitting.

Once a question has been asked, and the relevant data has been classified correctly, what remains is to choose a type graph which will effectively show off the data in a way that enables the viewer to answer their question. Options for types of graphs range from the classic scatter plots, bar graphs and pie charts, to more creative and unconventional graphs like heatmaps, ridgeline plots, dendrograms, spider plots, and many more. Each type of graph is designed to communicate a specific kind of relationship in the data being displayed. For example, scatterplots, heatmaps and bubble plots (scatter plot with the dot size representing a third variable) are all meant to display a possible correlation between variables. Alternatively, boxplots and histograms reflect the distribution of a variable; bar graphs and spider plots show size and ranking; pie charts and dendrograms demonstrate parts of a whole. Map visualization is another category in itself, which includes options such as choropleth maps, hexbin maps, and cartograms (Holtz & Healy, 2018). Although there is not time to give detailed explanations of each of these types of graphs, it suffices to say that there exists a wealth of visualization solutions, and the right one to choose depends entirely on the nature of the data available and the question to be answered.

Finally, even though there are many creative and exciting ways to visualize data, the ultimate goal of communicating new insights truthfully must not be lost. For example, a certain unconventional graph may be visually appealing, but it still needs to be tested in terms of how well it allows the viewer to understand and gain insights about the data. Making design choices that are counter-intuitive, overly cluttered, etc. will slow down the learning process (Holtz & Healy, 2018). For example, animated graphs can be an eye-catching way to show how data is changing over time, but the overuse of animation will make it difficult for the viewer to dig down into the graphic and understand everything that it communicates. According to Falaki

(2015), effective visualizations are interactive, collaborative, shareable, reproducible, and have control over the details. These larger goals cannot become second to uniqueness or creativity. All of this being said, the study of visualization design is an extensive and ever-growing field, with both many ways to succeed and many ways to go wrong. Those who work with data need to be well-versed in the complexities of using visualization as a tool for analysis and learning.

The Savings Group Model for Development

"In a self-reliant development effort, credit cannot be generated without savings" (Wickrama & Keith, 1994, p. 367). This quote embodies a core truth behind the concept of the "savings group," a phenomenon which has begun to take the development world by storm. This development model, in which a group of people come together to form their own autonomous, small-scale bank, was first introduced in the 1990s and has already proved to be revolutionary in its potential to empower people (especially women) living in poverty. Yet, as with any practical development effort, this system is not without challenges, limitations, and room for growth. This paper will discuss the main purposes and intentions in the emergence of the savings group model, as well as its impact and the ways in which we can continue to move forward in the field.

Savings Groups in the World of Microfinance

Microfinance intuitions (MFI's), such as cooperative credit unions, have taken great strides in the financial inclusion of the world's poor. Micro-loans, which are small, low-interest loans provided as a service to people of low income, often offer a solution for those who would not otherwise have the credit to access traditional loans. A small amount of startup capital can be a great way to kickstart small businesses with and allow them to reach the next level of production. Especially for individuals who have a sufficiently-established business to be able to benefit from a loan, or who live in an area with adequate economic activity, this type of micro-credit has the

potential to be very beneficial (Ashe & Neilan, 2014). Nevertheless, we are beginning to see that this model is not always sufficient nor sustainable.

One major disadvantage of traditional MFI's is that they fail to reach the poorest of the poor well (Wickrama & Keith, 1994). These are often the people living in rural areas who don't have access to the same markets that those who live in urban areas do. Despite the fact that MFI's have become very widespread and established over recent decades, they have failed to set down strong roots in rural areas of low economic activity, because in these situations the institutions are almost always unable to maintain their costs (Allen, 2006). Organizations that do attempt to offer micro-loan services to the poorest of the poor are often NGO's or nonprofits that draw much of their capital from donations, because there is less of a pressure for them to be sustainable based on loan interest capital. But this is not a good long-term business model, nor is it difficult for the effort to begin to cost more than it is worth. According to Mark Havers (1996), even though these institutions are built for the purpose of providing help to a community in need, that does not mean that the economic sustainability of the organization is not a valid or important goal. However, when it comes to serving the poorest rural communities, the cost of running an MFI becomes too much to sustain.

Another major issue with MFI's is that they do not offer what many of the poorest people in the world could really benefit from: a platform for saving. MFI's by definition always begin with debt, not savings, which of course requires outside capital (Ashe & Neilan, 2014). For people who don't feel secure in their earning power, saving is usually seen as a much less risky way to begin to build assets than by taking a small business loan (Allen, 2006). When the concern is true financial inclusion, it is important to be aware that not all people are in a position to be able to benefit from a loan—no matter how small, paying back that loan will eventually become a

burden. In the words of Vonderlack and Schreiner (2002), “although not all people are credit-worthy or want debt, all people are deposit-worthy and want assets” (p. 603). Thus, though micro-credit is potentially helpful for some, it is not the financial solution for all.

This is where savings groups come in. Savings groups have two major advantages over MFI’s: one is, of course, that they offer savings services; secondly, they are incredibly low-cost, meaning that they can be sustainable even in the poorest of communities. As an example, in 2006, the average cost for CARE (a US-based NGO) to oversee their savings groups was somewhere between $25 and $40 per client, with some costs as low as $15 per client in India. This is less than 10% of the cost per client for a traditional MFI (Allen, 2006). Even these numbers, though, are high: ideally, after a group has been trained and is well-established, it should require no outside management or oversight at all. As savings groups are built on intimacy and transparency, and do not rely on outside startup capital, they simply do not require the heavy regulation that credit unions and other types of MFI’s do (Ashe & Neilan, 2014). In addition, according to Havers (1996), even though savings groups do involve loans, the fact that they are rooted in saving naturally impacts the behavior of the borrowers in ways that make the program more sustainable. The reasons for this include that borrowers will be more motivated and cautious with loans taken from the same schema with which they are saving, that the model provides some collateral security, and that savings encourage people to view their money in a more long-term perspective. Exactly how savings groups achieve these things will be made clear in the continuation of this paper. The following section will cover the emergence and design of the savings group model, demonstrating how it is set up provide an alternative to the limitations of MFI’s—and how the benefits go even beyond.

The What and How of Savings Groups

A savings group is sometimes referred to as a Village Savings and Loan Association, or VSLA. This is the name of the concept that was first formally implemented in 1991 by CARE, a well-established worldwide NGO (Allen, 2006). Today, NGOs and nonprofits such as Oxfam America, Freedom from Hunger, Catholic Relief Services, Plan International, the Aga Khan Foundation, Pact, and others all have some sort of savings group or VSLA program implementation (Ashe & Neilan, 2014). However, the design of these groups was not formulated out of thin air by these NGOs; rather, it was inspired by communal saving practices have long been occurring in various forms in different developing communities all around the world. With the goal of finding a better way to financially include the poorest of the poor, especially women, the individuals who formalized the savings group concept decided that the best place to start would be with informal mechanisms that poor women are already voluntarily using. The continued existence of these mechanisms proves that there is a desire to save among people in impoverished communities, and enough that the women who participate in them are willing to work hard at it (Vonderlack and Schreiner, 2002). The most prominent of these informal systems is termed the Rotating Savings and Credit Association (ROSCA). ROSCA's have been observed primarily in Africa, and go by different names in different regions, such as *tontines*, *iqqubs*, *merry-go-rounds*, *osusus's*, etc. In the ROSCA model, a community will pool their savings into a collective pot, and then distribute that pot to a single member of the group on a rotating basis. That way, in turn, each person has access to a lump sum of money that they can use to make purchases or investments that they never would have been able to make with their personal savings alone. The advantage of these groups is that they provide a very low-maintenance way to drive savings; however, they do not provide a platform for loans or investment (Allen, 2006).

Thus, it is within the context of MFI's that fail to include the poorest communities, as well as the tradition of informal savings mechanisms that people already participate in, that the current savings group model has emerged. The savings group system is designed based on a set of core principles, identified by Jeffrey Ashe and Kyla Jagger Neilan in their book *In Their Own Hands: How Savings Groups are Revolutionizing Development* (2014). The principles include building on what is already in place (the ROSCAs), being committed to sustainability, keeping costs low, not providing giveaways, insisting on local control, designing for viral replication, and embracing learning and innovation. According to Ashe and Neilan, these fundamental concepts are what enable savings groups to survive even the toughest of circumstances. Thus, these ideas are foundational for the savings group model, and they will continue to be elaborated upon in the continuation of this paper.

To illustrate how the basic structure of the model works, a savings group will usually consist of about 25-30 people in a community, typically women, that come together to form a sort of informal group bank. They meet at regular intervals—weekly, biweekly, etc.—to complete transactions. During these meetings, members will invest money into the group fund, sometimes by buying a certain number of shares, or sometimes with a simple monetary amount, depending upon how the group has been structured. Over time, this group fund becomes large enough to support being loaned out. Members can make proposals to the group for how much money they would like to request and what they plan to use it for, and then the group will decide whether or not to grant them the loan. Once granted, the borrower is able to use the money for whatever they had planned, be it new livestock, children's school fees, etc., and they have a set amount of time in which they must pay that loan back with interest. The interest rates and the loan durations are set in the group constitution, a contract of by-laws that is mutually decided

upon before the group begins its proceedings. Ideally, a high percentage of the group fund is loaned out at all times, and members pay back their loans plus interest consistently, so that there is less risk of robbery and the money is continually growing. All of the group's financial transactions are recorded in a ledger book, unless the majority of the group is illiterate, in which case a more simplistic design approach is taken which relies completely on oral accounting. Finally, at the end of the cycle (a time period that is most typically one year), all the loan repayments are collected, and the group holds what is called a "share-out." At this event, all the money that was contributed plus the interest earnings is redistributed back to members according to how much they invested (Ashe & Neilan, 2014). Having a set share-out date at the end of every cycle forces all outstanding issues to be resolved, and it prevents the group money from accumulating to an unmanageable level, therefore ensuring a clean slate every so often (Allen, 2006). After a cycle is over, members have the chance to decide if they want to participate in another cycle.

Policy and procedures can vary widely, so the rules of conduct for each group must be clearly defined in their constitution. Having a documented set of standards gives the group members a basis on which to hold each other accountable. If a member breaks policy—for example, they do not show up for a group meeting—there is a monetary fine associated with the infraction, and that money is added to the group fund. As opposed to how it may seem, the lack of an outside stakeholder to motivate people to pay back loans with integrity is not usually an issue for groups; on the contrary, the presence of peer pressure to be responsible for one's dues and remain in good standing with one's friends and neighbors has proven to be a highly effective system of accountability (Duffy-Tumasz, 2009). Also, as a practical security measure, the

group's funds are held in a safe deposit box with three keys which are entrusted to three separate people, providing reasonable insurance that the money can't be tampered with.

Additionally, savings groups have the opportunity to utilize their congregations as platforms for more than just loans and investments. Often, they will establish what is called the social fund, which acts as a small-scale version of insurance. Members contribute small amounts to the social fund each meeting, and if someone in the group has an emergency (such as a family member sickness, death, etc.), they can be granted a portion of the social fund to help cover the unexpected costs. Also, group meetings are opportune times to host social education initiatives, such as a malaria prevention presentation, family planning lesson, or whatever else it may be. These types of events are usually organized by an implementing NGO or nonprofit (Duffy-Tumasz, 2009).

All details aside, the fundamental characteristics of VSLA's are that they are self-managing, they operate with high levels of transparency, and they promote financial independence. They are also attractive because they can offer a high return on investment (even exceeding high rates of inflation), as well as access to loans higher than one's amount of saving. And although they are often started up by a facilitating nonprofit, the need for external input decreases over time as groups become completely autonomous and even self-replicating, creating spontaneous growth (Allen, 2006). All these factors come together to create a savings group model with incredible potential to do good in communities, and to help people lift up each other up.

Impact and Empowerment

In order to understand the true effect of savings groups/VSLAs, it is important to look beyond just the model design intentions and attempt to actually measure impact. One way to

begin doing this is by observing their expansion and growth. In certain situations, savings groups have and continue to experience explosive growth. As an example of how fast a single program can take off, the NGO Oxfam America began its first savings-group trial program in Mali in 2005, under the campaign name Saving for Change. The program was successful and began to expand rapidly, such that in 2008, Saving for Change had 110,000 members across western Africa (Duffy-Tumasz, 2009). Jeffrey Ashe, the man who was responsible for the inception of Saving for Change, reports that from 2008 to 2014, participation in savings groups worldwide (not just Saving for Change) grew from one million to ten million people, and the experts predict that with enough effort, that number has the potential to jump to 50 million by 2020 (Ashe & Neilan, 2014). Also, the VSL Associates consultancy reports that VSLA's now exist in more than 61 countries worldwide (Ksoll et. al., 2016). Some of this rapid growth is due to NGO's that have recognized the power of savings groups and put in effort to spread the idea. However, some of it is also due to spontaneous replication—when savings group members themselves, having been motivated by their positive experience, voluntarily help others create new groups. These numbers demonstrate that the savings group model has become a relatively widespread phenomenon.

One of the biggest positive impacts of the savings group model is the empowerment and solidarity that it enables, especially through the organization of women (Ashe & Neilan, 2014). The design of savings groups is founded upon putting knowledge and control of finances in the hands of group members themselves. This is a significant improvement to taking loans from MFI's, where the borrower is only a small part of a complex system and is dependent upon the institution to provide both the funds and the loan management. It is also a large contrast to taking loans from local village traders, which is another thing that occurs in poor communities. Local

trader loans are often very dangerous, because they come with high interest rates and strict obligations to the lender. In these types of situations, where there is a transaction between a powerful elite and someone who is powerless, it is easy to see how one can be caught in a vicious cycle of indebtedness (Wickrama and Keith, 1994).

Savings groups, on the other hand, are just the opposite. They are agreements between peers rather than transactions between one more powerful body and one much less powerful. In his foreword for *In Their Own Hands*, Francis Moore Lappé writes, "Most aid interventions collapse as soon as outsiders leave… but the savings group approach transfers some very basic tools and gets out of the way. It enables people to tap into their own ingenuity, determination, and creativity" (Ashe & Neilan, 2014, p. xi). Any development model that creates dependency is not going to be empowering; however, organizing people, letting them manage their own money, and allowing them to understand the process and participate in group constitution decisions is very empowering (Ashe & Neilan, 2014). This one of the reasons why these groups most often geared for women in particular. (The other major reason, now a well-known fact proven by multiple studies, is that women are more likely to spend micro-loans well and pay them back than men are) (Duffy-Tumasz, 2009). Women living in poverty are often fully engaged in unpaid family work, such as cooking, gardening, and caring for children. Access to resources for advancement are not as available for them as they are to men, and married women rarely have any say when it comes to financial decisions in the household (Wickrama & Keith, 1994). However, many women who have participated in savings groups have reported that bringing in their own savings has earned them more respect from their husbands. In addition, being part of an organized support group of women has given them courage to be bolder in their households (Ashe & Neilan, 2014).

It is clear that savings groups have become popular and are designed to be empowering, but it is another matter to measure true impact that they have on individuals and communities. To this end, there have been various formal studies of savings groups over the years that attempt to come up with a quantifiable answer to how effective the model really is. For instance, CARE began introducing a pilot VSLA project in Zimbabwe in 1999 called Kupfuna Ishungu (KI), and conducted a study of it in 2004. According to their findings, KI was the only financial system in the country that was producing a 15% positive real return on savings, during a time when inflation of their currency was at 250%. The KI study also reported a majority positive impact on measurables such as product assets, level of food consumption, access to health services, status in the community, and feeling of cooperation between husband and wife (Allen 2006). Another study, performed by Ksoll et. al. (2016), focused on a randomized cluster of villages in northern Malawi, observing the introduction of VSLAs to 23 villages in the area in comparison to 23 control villages from 2009 to 2011. They measured success in terms of food security, amount of income-generating activities, and amount of household income. A few of the positive impacts on families due to participation in savings groups that they reported were the ability to even out food consumption over the agricultural cycle, some insurance against illness, emergency, death, etc., increased commitment to saving, and built-up trust and relationships that can help facilitate economic activity. Although, for various reasons, the study certainly did not find the savings groups villages to be significantly better off than the control villages in all areas, they did conclude that the introduction of VSLAs caused improved food security and strengthened household income (Ksoll et. al., 2016).

Although there have been research efforts, such as the two previously mentioned, that have tried to quantify the positive impact of the savings group model, the truth is that there are

not many studies that have been able to use data to conclude a strong positive impact. In fact, research on the impact and efficiency of savings group models in different areas of the world has been identified by many sources as one of the most desired areas for growth in this field. In regard to VSLA, Ksoll et. al. (2016) states, "despite this prevalence and increasing popularity among donors as a means of improving the financial infrastructure in remote rural areas, very little is known about [VSLA] impact on household welfare" (p. 3). Part of this lack of concrete evidence can be attributed to the huge range of variables which play into the economic stability of developing communities, making it difficult to isolate causation. But, beyond that, it is also partially due to the difficult and inefficient process that is needed to collect and access data about groups. More about these challenges will be discussed in following sections; however, the emphasis is that despite a lack of concrete data, the savings group model has exemplified success in many individual use cases, and it is continuing to grow in popularity.

Limitations and Challenges

Accounting.

As has been established, a core component of savings groups is that they are autonomous, meaning that the groups are responsible for understanding and managing their own loans, investments, etc. As previously mentioned, group financial transactions are often kept in a ledger book, and calculations are made by hand. Ashe and Neilan (2014), when describing this situation in the context of a savings group in Mali, shared that "at the low levels of literacy found in Mali, ledger entries would generate a lot of waiting time and frustration in meetings while a few people struggled over computations and other members waited for results they only partially understood" (p. 86-87). This observation captures what has been identified by NGO's and researchers across the board as a challenge that savings groups face. Ultimately, the reliability of

groups can only be as good as their accounting (Ashe & Neilan, 2014). In order to effectively build on the ROSCA model to meet the needs of the group, the savings group model must offer a platform for investment and loans—the natural consequence being that the issue of money management becomes a bit more complex. Thus, the challenge for savings group implementers is to find ways to keep the process simple enough to be maintained by even illiterate groups, which is not an easy task (Allen, 2006).

Research and Data Gathering.

Even though the savings group model does not technically require outside capital in order to function, nonprofit organizations that spend time and labor trying to grow programs, train leaders, and spread groups to new communities require funding. However, this funding can be difficult to come by as the model not incredibly well-known, and there is little information or technology involved to excite donors about it. Even though there is an abundance of personal success testimonies that have emerged from savings groups, as Ashe states, "from the perspective of a government official, international donor organization, or investor, however, it is important not only to hear these stories but to see concrete evidence showing how much a program has affected a community" (Ashe & Neilan, 2014, p. 117). Therefore, what is needed is a serious effort put towards research and gathering of data-based evidence of the effectiveness of savings groups, which can be used as a tool to secure more funding. The problem is that the research itself requires money as well, and so the situation appears stuck in a perpetual loop.

Conclusive research would not only be helpful for securing funding, but it is also fundamental in the process of continuous improvement. Because of the high amount of variability in exactly how groups are started, how they are trained, and how they design their constitutions, there are always opportunities to make changes with the goal of creating more

successful groups. Ashe and Neilan refer to this idea of continuous improvement as a "self-reinforced learning loop," which means "constantly updating our manuals based on what we learned from the field" (p. 83). They recognize that progress is just as important as the result. However, continuous improvement efforts cannot be simply random. Discovering what the most effective versions of the savings group model are requires research based on data, which the field is severely lacking. As Allen (2006) expresses, "most VSL[A] programs are locally implemented and give very little attention to data gathering and analysis… There is, then, an urgent need to develop a set of ratios and standards of financial measurement that enable programs to assess their own effectiveness vis-à-vis peers and to enable the industry as a whole to assess the cost-effectiveness of VSL[A]" (p. 67). If research and data are able to discover which groups are most successful and why, those design aspects can and should be imitated in all development initiatives (Ashe & Neilan, 2014).

Growth and Moving Forward

Savings groups have brought about a new level of financial inclusion and have grown and spread rapidly since their inception. Being built upon the already existing ROSCA model, while still keeping in mind the limitations of MFIs and the needs of communities, they are designed to provide exactly the financial platform that women and men in poor communities need. VSLA does have its limitations; however, in the words of Hugh Allen, "it costs very little; has a high rate of success; is enthusiastically supported by participants and, most important of all, provides access, simplicity, safety and speed of service delivery" (p. 68). As such, the savings group experiment, which started out small, has now reached considerable scale and importance in the world of development. The challenge, currently, is for organizations who implement savings groups to transition their small-scale approach to accommodate a large-scale movement,

"especially in terms of inter-regional learning and technology transfer" (Allen, 2006, p. 68). Finding solutions to the difficulty of group accounting as well as the need for data and research are ways that organizations can attempt to both keep up with and spur on the growth of this model. Fortunately, the DreamSave mobile application for savings groups provides a platform that can address both of those things.

Utilizing Big Data for the Savings Group Model

Introduction to the DreamSave Application

DreamStart Labs is a technology startup company that was founded in 2017 by two successful Silicon Valley business and technology leaders, Wes Wasson and Henrik Esbensen. These two were introduced to the revolutionary world of savings groups, and they recognized an opportunity to utilize technological innovation to make an impact on global poverty alleviation. So, they set out to build a mobile app designed for savings groups. Three years later, in 2020, their app called DreamSave is in production and has been adopted by its first few waves of savings groups. It is well on its way to becoming a widely-used application in many regions around the world.

The DreamSave Android application is a product that has been developed to improve the implementation, organization, maintenance and record-keeping of savings groups. As has already been mentioned, one of the biggest hurdles that savings groups must overcome is the difficulty and inefficiency of keeping their financial records. Now, if at least one member in the group has an Android smartphone (which the statistics say is fairly probable even in poor areas), the group can easily record all their transactions using DreamSave. Doing complex calculations by hand, which have long become a thing of the past in the developed world, will now no longer be necessary for savings groups who use the app. Additionally, DreamSave provides members

with a formal, documented way to build credit history, something that normally they would never be able to do without being a part of an official bank or microfinance institution. Having this ease of access to money management can also help to increase the financial literacy of savings group members beyond the official bookkeeper (DreamStart Labs, 2019).

When a group begins using DreamSave, they will create user accounts for members, register as a new savings group, specify the terms of their constitution, establish the necessary security measures, and begin entering events as they happen: a meeting is held, a share is purchased, a loan is provide, a fine is imposed, etc. The UI of the app is pleasantly designed, with easy navigation and dashboards to encourage users to make the most of their participation in the group and meet their personal goals. It is not necessary to explain the functions of the application in detail here; however, it is significant to understand that DreamSave was very intentionally and thoughtfully designed for the unique needs of the context. For example, the fact that the app can be used even if the group has only one smartphone between them creates accessibility for groups with very few resources. But, if other members do have smartphones, they have the option to download the application to view their information; likewise, members with basic phones can receive notifications of their transactions and balances by text. Also, because network connectivity in the rural areas where many groups reside is often inconsistent at best, the app is able to be used completely offline. However, when the device does become connected to a network, the data will be automatically backed up in the cloud. Furthermore, the designers also knew that each group operates with different policies; therefore, it is simple to input those regulations into the app during the initial setup, and as events are recorded, the app will prompt the group to comply with them. These are just a few examples of the robust features

within DreamSave that make it ideal for use by diverse savings groups and resilient to the unique challenges of creating a mobile application for the developing world (DreamStart Labs, 2019).

DreamSave as a Means of Big Data for Savings Groups

There are many aspects of DreamSave that create opportunity for improvement in the functioning of savings groups, but the one that this research is concerned with is the new source of data that it offers. DreamSave has been designed for scale from the beginning, and the hope of its creators is that the use of the app takes off in the coming years. Should this happen as expected, the digital byproduct of DreamSave will begin producing big data that is all about savings groups. Rather than allowing this information to simply sit in a database, there is an opportunity to find ways to analyze that wealth of data and draw out insights that can contribute to the continuous improvement of the savings group model. Knowing that one of the biggest limitations of the development effort around savings groups is the lack of data for research, making use of the application as a resource for big data analytics is an incredibly worthwhile endeavor. In contrast to how costly and time-consuming it is to travel from place to place collecting data for research, data from the DreamSave cloud is free and accessible from any location, and available in relatively real time, as long as the group has some way of regularly connecting to a network. Information regarding demographics, the nature of the financial transactions, and the resulting financial performance will be available on the level of individual members, groups as a whole, or aggregated across a country, continent or even the world. The goal is that this big data can be visualized, analyzed, and finally leveraged to both support awareness and advocacy for the model, as well as to inform better decisions regarding its implementation.

The beauty of the goal of visualizing and analyzing savings group data is that utilizes some of the beneficial concepts of big data, but at the same time, maintains a narrow focus of research. The data comes as a natural byproduct of the app and does not require any extra time or money to be gathered, similar to other types of big data. But it is also built into a pre-defined scope, namely, savings group participants who use the DreamSave app. Thus, when making observations about DreamSave users, the entire population is represented (ideally), not just a subset or sample. Conclusions drawn from this data will not pretend to apply to everyone, but only to savings groups who use the app. However, by considering DreamSave users as a representation of the larger community of savings groups members, either worldwide or within a certain demographic or region, insights can be cautiously drawn about the entire strategy as a whole. And as the app continues to be taken up by new groups in new regions, its statistical power will only increase and make it a better tool for research. Therefore, just as the rise of the big data movement has had a significant impact on the modern business and technological scene, it also has the potential to greatly benefit the field of development through savings groups, beginning with leveraging the potential of the DreamSave application.

Part 2: Technical Review of Data Software Utilized by DreamSave

The purpose and scope of this research is to identify how concepts of big data visualization can be applied to the context of savings groups, as well as to investigate the technical data tools that can be used to do this with DreamSave data. It is interdisciplinary in nature, as savings groups fall into the realm of international development; however, the primary goal is to view this topic as an application of computer science. As such, the focus here is on the use of computation in order to present data in a way that enables successful analysis, rather than offering any actual theories or insights about savings groups derived from that data analysis. In

that regard, it is appropriate to first take a deeper dive into the specific backend technologies that will be involved when working with DreamSave data, discovering how they are structured, why they are effective tools for big data, and how to understand them from a database engineer's point of view. Therefore, the following sections will examine in depth the two main technologies that will be touching the DreamSave data in the process of visualizing it: Apache Cassandra and Apache Spark.

Apache Cassandra

Apache Cassandra is a NoSQL database system that was originally developed by Facebook. Maintaining Cassandra was initially an internal effort, but in 2008, Facebook made the project open source, which led to it being adopted by the Apache Software Foundation in 2009 (Tutorials Point, n.d.). Cassandra was designed with the goal of being highly scalable, with the capacity for very large amounts of data; highly performant, with optimized data availability; and highly fault-tolerant, with a replication system that provides layers of protection against losing data due to technical failures. Over the years, this design seems to have proven its worth—Cassandra is currently being utilized by over 1500 companies with large datasets, including Netflix, eBay, GitHub, and Instagram, to name a few (The Apache Software Foundation, n.d.).

The architecture of Cassandra consists of a schema built upon a cluster of physical machines, known as nodes. Though they work together, each node is equal in capability and able to function independently. (To clarify the terminology, a Cassandra *cluster* may contain multiple *data centers*, each of which are collections of related *nodes*) (Tutorials Point, n.d.). There is no "master" node that contains all the data—each one only stores a subset. Nonetheless, each node in a Cassandra cluster is in communication with all the others and is able to perform reads and writes for any data that is requested. Because of this, a client application need only interface with

one of the nodes in the cluster, and it will still be able to access data regardless of the machine that it is physically stored on. Also, an individual piece of data is never stored on just one machine in a cluster, but rather it is replicated across multiple nodes according to a configurable replication factor (three is typical). Having a structure where a dataset is distributed and replicated across a cluster of equivalent machines has significant benefits. First of all, horizontal scaling is easily implemented, as systems which find themselves needing more storage or processing power can simply add another machine into the cluster, and Cassandra will automatically incorporate that into the schema to reduce the burden on the other nodes. Secondly, should one node fail (as hardware sometimes does), it will not cause any loss of data, as the referencing machine can simply go to another node that contains the same data (Brown, 2013).

Unfortunately, there are almost always trade-offs when it comes to database design, and while Cassandra is structured for high data availability, scalability and fault-tolerance, there are other areas in which it does not measure up to the standards set by relational databases. Because of the distributed, replicated nature of data in Cassandra, write-latency and data consistency become factors that must be considered. Every time a piece of data is changed in Cassandra, that update must be written to each of the replicated instances, resulting in more latency than if the data only existed in one physical place (Datastax Inc., 2019). In that situation, there is a period of time in which a piece of data might be updated on one node, but before the change is able to be issued to each replication site, the same piece of data is read at another node. Because the update has not yet been applied to that node when the read happens, the information that is returned is inaccurate, resulting in a stale read. However, in the context of big data, latency and consistency problems are often very minimal in comparison to the value of data replication.

The concept of a distributed dataset on a master-less cluster of nodes is not unique to Cassandra—in fact, there are plenty of other well-known databases that implement this kind of structure. What makes Cassandra different from these has to do with how it does data modeling (Brown, 2013). Data modelling "is a process that involves identifying the entities (items to be stored) and the relationships between entities"—in other words, it has to do with how data is organized into a logical schema. In Cassandra, data modeling is done with a *query-driven* approach, meaning that the schemas are designed with the goal of optimizing queries, reads and writes (Datastax Inc., 2019). Cassandra's structure is designed in such a way that is relatively straightforward for users to create a compatible data model that suits their needs, mostly because it very closely resembles the familiar table structure of a relational database (Brown, 2013). Although, whereas with relational databases, the goal is to normalize the data (remove as much duplication as possible in order to maximize storage space), the objective is slightly different with Cassandra. In a query-driven database like Cassandra, the ideal is to have one query per table, and data can be repeated across multiple tables if necessary. Cassandra focuses on the relationships between entity instances, and duplicates those common attributes across entity tables, rather than focusing on the relationships between tables. This removes much of the necessity for the client applications to do table-joins in order to bring together the data that they need, which are generally costly transactions in terms of performance (Datastax Inc., 2019).

So, Cassandra does organize data into tables, with columns representing attributes and rows representing entities; however, the implementation of these structures is distinct from that of a traditional SQL database in several important ways. Cassandra is considered to be a wide-row-store type of NoSQL database, and it is set up using the concept of a *column family*. A column family is a container of a collection of rows which each have ordered columns. It is often

conceptually equated to a relational database table, except that the columns associated with each row can vary, and they can also be added or taken away dynamically. In contrast, in relational database schemas, the columns in a table are fixed, and each row must at least have a null value for each column (Tutorials Point, n.d.). To explain further, each "wide row" in Cassandra is an individual entity and has an identifier which points to a collection of columns, similar to a key-value hash. Each column also has an orderable identity (key) associated with it (most often a date-based universally-unique identifier), and the columns are always ordered within the container according to that identity. Cassandra wide rows can contain any number of columns (practically, up to about two billion). Column families, therefore, are groupings of these wide rows (Brown, 2013). The identifier which points to a wide row is officially titled a *partition key*, and under the hood, it is the partition key that Cassandra uses to pinpoint which node contains the wide row. The orderable identity that is associated with each column is called the *clustering key*. Finally, column families in Cassandra belong to a certain *keyspace*, which is similar to a namespace and is the biggest logical container in a Cassandra database. It is at the keyspace level where the user can define the *replication factor*—that is, how many times each wide row will be replicated across the nodes (Datastax Inc., 2019).

Although Cassandra's data model is relatively straightforward and familiar for a NoSQL database, it is still a bit more complicated to set up and involves more considerations than the process of setting up a relational database does. One of the shortcomings of Cassandra is that the way the column families are configured will ultimately constrain what can be done with them. This is because in Cassandra, unlike a relational database, there is no concept of table joins or even foreign keys. Neither can one change the column by which rows are ordered—rows are always ordered the same way, according to the attribute configured from setup. Thus, even

though you can always add new columns to Cassandra column families, their actual structure cannot be dynamically manipulated in the same ways that relational tables can. Therefore, it is important when designing a data model for Cassandra that the user creates a structure that is set up to optimize the access operations that will be most commonly used (Brown, 2013). In addition, finding the ideal partition size becomes another important consideration when configuring a data model for Cassandra. The less amount of partitions necessary to query, the more efficient the query will be; but, if the partition itself is too large, that can slow the query down as well (Datastax Inc., 2019). There can also be practical limits when it comes to the size of a partition—since a partition is essentially a wide row and considered a single entity, it is never split up between different nodes to be stored (Brown, 2013).

Just as relational databases rely on SQL to perform their operations, Cassandra has its own querying language called CQL. CQL uses much of the same syntax and allows users to do many of the same database operations as SQL does, such as select, insert, create a table, etc. However, CQL is still not as powerful as SQL: joins, subqueries, offsets, and many of the other advanced options that SQL allows are not possible (Brown, 2013). However, the capabilities that CQL does have offer a level of ease and accessibility that is normally harder to come by with NoSQL databases. Thus, Cassandra can be a great option for organizations who would like to maintain the familiarity of a relational-like structure, but also be able to manage their big data with top-notch accessibility, fault tolerance, and scalability.

Apache Spark

Big data is not only difficult to store, but in order to access and manipulate it in ways that are meaningful to learning, unsurprisingly, a lot of processing power is required. Just as the rise of big data drove the development of cluster database models like Cassandra, the rise of these

cluster databases generated a need to create data processing engines that are designed and optimized to work with such clusters. In the words of Zaharia et. al. (2016), “The very nature of ‘big data’ is that it is diverse and messy; a typical pipeline will need MapReduce-like code for data loading, SQL-like queries, and iterative machine learning” (p. 56). Such requirements for processing big data on a cluster are difficult and costly, but overcoming that challenge is exactly how products such as Apache Spark came about.

Apache Spark was first developed by a team at the University of California, Berkeley, but was released as an open-source project in 2010 and is now managed by the Apache Software Foundation (Landset et. al., 2015). The original purpose of Spark was to create an engine with the capabilities of doing what previously had to be done inefficiently by separate engines, such as data streaming, graph processing, machine learning, and SQL querying (Zaharia et. al., 2016). Today, Spark is the largest open-source data processing engine project, and it has enabled impressive results when it comes to improving performance in this realm. In fact, the largest use case of Spark that has been reported is by a Chinese social media platform, which has Spark running on top of 8000 nodes (Zaharia et. al., 2016).

In order to understand what Spark is and how it works, it is important to first be familiar with its precursor: MapReduce. MapReduce was Google’s first architecture product, which “combined flexible, scalable storage with a single processing framework” (Cloudera, 2019). Put very simply, it defines a framework for working with cluster data that revolves around a two-step procedure of “mapping” and “reducing.” “Mapping” consists of applying some sort of operation to a set of data on a node, and “reducing” refers to synchronizing the data across nodes (Zaharia et. al., 2016). (Some sources describe this as actually a three-step process, differentiating a “shuffle” step from the reduce step. In this method of understanding, “shuffling” is performing

synchronization across nodes, and "reducing" is gathering the results from the different nodes together.) Each MapReduce operation must perform these steps, so a complex job would have to string together multiple iterations of "map-reduce." This framework is ideal in terms of fault-tolerance as well as throughput, but it suffers in terms of latency. Because each operation is calculated based on the result of the previous one, the delay associated with each step of a complex MapReduce job can easily add up and affect the efficiency of job (Urma, 2017).

Apache Spark is a general-purpose data processing engine that was built using the same basic model as MapReduce, maintaining its scalability, distribution, and fault-tolerance (McDonald, 2018). It consists of a higher-level API than that of MapReduce, through which it has modernized, optimized, and added new possibilities to what can be done with just MapReduce (Urma, 2017). It is also a "storage-system-agnostic engine," meaning it can be used on top of any database, and it can even draw from any number of different data sources at the same time, enabling data from different databases to be unified and processed in the same place (Zaharia et. al., 2016). Spark's efficient and thorough API for manipulating data provides the base platform; then, there are various specialized libraries on top of Spark that utilize this API to perform tasks. The four main libraries are the SQL and DataFrames library (to be elaborated on later), Spark Streaming, for analyzing real-time data, GraphX, for graphing and visualization, and MLib, for machine learning. The separation of these libraries allows organizations that use Spark to deal only with those that suit their needs. According to surveys of Spark users, over 80% use at least two of Spark's libraries (including the Spark Core for basic functionality), and 60% of them use at least three (Zaharia et. al., 2016). And, as Spark is still an open-source product, there are new libraries being worked on by contributors all the time.

The key way that Spark improves upon MapReduce is by the way that it optimizes complex data transformation jobs, making smart use of RAM. To illustrate, MapReduce performs its computations through a series of map and reduce steps, synchronizing after each consecutive calculation. In other words, it has to literally "write out its state after every step," which is a performance-costly task, especially when the data is replicated multiple times on a cluster (Zaharia et. al., 2016, p. 63). Thus, the latency of a process is multiplied by the amount of computations. In contrast, Spark, through the use of a data structure called an RDD, is able to take this model but implement it without the limitations. When Spark is given a job that consists of multiple data transformations, it takes the data in question and stores it in an RDD in a node's RAM. Then, it performs the series of operations on that data, but does not write the results back to the hard disk until all of the computations have been completed. Also, instead of relying on synchronization in the same way that MapReduce does, it emulates in-memory data sharing to ensure the integrity of the computations. This way, time is not wasted on replicating intermediate data, and the costly writing-to-disk task is only performed once for each batch job. The one drawback of this strategy is that when Spark finally does perform a synchronization step, the latency tends to be a bit higher than with MapReduce, so simple jobs consisting of one or two computations in Spark may perform worse than they do in MapReduce (Zaharia et. al., 2016). But, as jobs get larger and more complex, Spark quickly pulls away in terms of speed. The efficiency of Spark is especially evident with iterative algorithms over distributed clusters in parallel (McDonald, 2018). In fact, at the Daytona Graysort Challenge in 2014, Spark sorted 100 TB of data three times faster while using only 10% of the machines than the previous record-holder, which was using MapReduce (Landset et. al., 2015).

As previously alluded, the revolutionary invention which has allowed Spark to use such an efficient model for processing data on a cluster is the RDD, or resilient distributed dataset. Spark operates using two types of functions, *actions* and *transformations*. An action, in the Spark sense of the word, is a function on a set of data that returns a value, such as calculating the sum of a set of numbers. Conversely, a Spark transformation is a calculation on a set of data that returns an altered set of data as a result—namely, an RDD (Willems, 2017). RDDs are said to be the building blocks of Spark, as they have been the basis structure of Spark since its inception. Essentially, an RDD can be understood like a collection of Java or Scala objects. RDDs are strongly typed (and therefore compile-time safe), immutable, and distributed across nodes (Willems, 2017 and Damji, 2016). Because these logical entities are physically distributed across the nodes on a cluster, they can be operated on in parallel, which is one of the reasons why RDD operations can be performed so quickly. Another cause for the efficiency of RDDs is that they are stored in RAM rather than on disk, which is a much faster location for the processor to access. Normally, RAM storage would come with a compromise in the data's fault tolerance; however, RDDs maintain fault-tolerance using a strategy called *lineage*: Spark keeps a record of the string of transformations that were used to create an RDD, and it can recreate the RDD if it was lost. According to Zaharia et. al. (2016), "Lineage-based recovery is significantly more efficient than replication in data-intensive workloads. It saves both time, because writing data over the network is much slower than writing it to RAM, and storage space in memory" (p. 59). The third and most important reason why RDDs are a more efficient way of processing distributed data is that they are calculated "lazily." To explain, Spark transformations will return an RDD representing the transformation without actually doing the full computation right away. The client can then continue to apply transformations to that RDD, and later, when the resulting

data set is needed by an action, Spark will create an optimized execution plan for performing all the transformations that lead up to final resulting RDD (Zaharia et. al., 2016).

Therefore, under the hood of Spark lies data structures and algorithms that are highly optimized for complex cluster data processing jobs. In addition, Spark's higher-level APIs contain further optimizations, enabling the unification of libraries that serve very different purposes. Because all of Spark's libraries function on top of RDD's, they are easy and performant to combine, so even though they are running on a single engine, they can often match the performance of a specialized engine, with the added advantage of being able to work with them all from the same platform. Zaharia et. al. (2016) says this about the subject:

> One powerful analogy for the value of unification is to compare smartphones to the separate portable devices that existed before them (such as cameras, cellphones, and GPS gadgets). In unifying the functions of these devices, smartphones enabled new applications that combine their functions (such as video messaging and Waze) that would not have been possible on any one device. (p. 57).

This analogy for Spark's ability to combine functional libraries demonstrates the power and potential of its architecture. Optimizations like these are what enable big, distributed data to be analyzed more efficiently and often.

One of Spark's most important libraries, as previously mentioned, is an API for SQL and DataFrames. This library was created in 2015, and it was developed to address how difficult it can be to analyze the arbitrary data types and objects contained in RDDs (Zaharia et. al., 2016). DataFrames are essentially another abstraction over top of RDDs. They are still immutable and distributed, but they superimpose an organization based on named columns over the objects in the RDD, so that they begin to resemble relational tables. These objects can be manipulated

using a language-specific API that allows the client program to perform SQL-style operations on the data such as filtering, adding columns, aggregating, etc. by simply calling functions in the host language. However, unlike RDDs, DataFrames are not strongly typed, making them ideal for interfacing with languages that do not do static typing—Python and R are the two for which Spark provides an API. However, when these are not appropriate, Spark also provides APIs for the statically-typed languages Java and Scala, and for which there is yet another abstraction used called a DataSet. A DataSet is essentially a collection of DataFrames with a JVM type imposed upon them. Details aside, the DataFrame and DataSet abstractions provided by Spark libraries are important because they make interfacing with the data much easier than working directly with actions and transformations on RDDs, as well as because they provide additional performance optimizations. When the client utilizes these higher-level APIs, Spark's engine will decide how to design the necessary RDD transformations and actions in order to minimize the amount of work that needs to be done and provide the fastest result (Damji, 2016 and Willems, 2017).

Although Spark has a long list of advantages and has proven its worth as a highly powerful processing engine, it also has its share of limitations, as every piece of technology does. Among these include the lack of an in-house file management system: usually, Spark has to rely on the Hadoop Distributed File System (HDFS). HDFS is designed for a limited number of large files, where Spark tends to work in large amounts of small files, so there can sometimes be issues that arise when the two are used together (Macwan, 2018). Another limitation of Spark is that it does not support actual real-time processing. Although Spark does have a library called Spark Streaming that is meant to simulate real-time operations, the RDD structure can really only work in batches. So, Spark Streaming has to compromise through the use of "micro-batching," or

completing many small batch jobs one after another to get a near-real-time effect (Landset et. al., 2015). Finally, although the use of RAM is one of the things that makes Spark as efficient as it is, RAM space is a much more expensive and limited than disk space, making Spark more susceptible to costly bottlenecks (Macwan, 2018). However, even with these limitations, the processing engine that has been created in Apache Spark is an impressive feat and represents an intelligent and modern solution to the problem of big data processing. Its power as a generalized engine with the ability to efficiently perform specialized functions makes it a worthy tool for leveraging big data.

Part 3: Practical Application

Intended Purpose

Having laid the groundwork with a theoretical exploration of the concepts of big data, data visualization and data for development, as well as the savings group model, DreamSave application, and specific big data technologies, the final portion of this project aims to build off of the ideas brought up by that research with a practical application. The understanding of these concepts and technologies provides context for the practical product that was created, and more importantly, to the process of creating it. The original intention for this project was to emulate, in some small way, the concept of using big data visualization for the purposes of international development, by means of building visualizations of data gathered by the DreamSave app. This would achieve the purpose of gaining hands-on programming experience in the field of data analytics, and also provide an opportunity to apply those practices to a context which could greatly benefit from it: savings group research. Thus, the purpose of creating visualizations of DreamSave data for this project is to provide a practical example of the intersection of big data and savings groups.

Scope, Audience and Goals

With the vast complexity of how savings groups look and are implemented, the task of creating a visualization platform from which to analyze VSLA data has the potential to be an incredibly extensive task. From the variety of group constitution policies, to the diversity of people, to the wide array of financial transactions involved, there are many different avenues that it is possible to go down when attempting to gain insights from savings group data. For a researcher or implementer of savings groups, the graphs that would be most appropriate and useful for their learning depend greatly on the specific questions that they are asking. Creating a platform from which a user could derive visualizations to answer any questions that they may have about savings groups would be beyond the scope of possibility for a one-person project such as this one. Therefore, rather than focusing on the needs of academic researchers or various nonprofit savings group project directors, it is necessary to choose an audience that is more specific and limited.

With that reality in mind, this project chooses to take a more exploratory rather than all-encompassing approach in the visualizations created, with the intended end user being DreamStart Labs themselves. As a company whose main purpose is to provide a digital platform with which savings groups can thrive, DreamStart Labs as an organization is less interested in analyzing specific strategies for savings group implementation than they are in seeing who their app is reaching, how it is being used, and how it can do its job better. In this way, they will have different data visualization needs than, for example, a nonprofit organization that wants to evaluate the success of their program and is only interested in the groups their own groups. DreamStart is interested in seeing all the groups together as a whole, who their biggest clients are, how well the app is being adopted, and by whom. In addition, as a company that believes in

their product and hopes to expand their user base, any visualizations that could get people excited about the application would be a valuable tool from a marketing standpoint. These would be visualizations that are more geared toward a lay audience of people who do not know much about how savings groups work, rather than toward researchers or professionals in the field.

Visualization Product Description

With these goals in mind, as well as the concepts of good visualizations, this project includes two small-scale, interactive server applications that will each achieve a themed purpose. The first, a geographic visualization, is aimed at showing where in the world the DreamSave app is being used by savings groups, as well as which NGO projects they are connected to. This server application has two main visualizations which can be toggled. The initial page is a choropleth map of the countries of the world. Countries are colored in shades of green according to how many groups are located there, with darker shades of green signifying a higher number of groups than the lighter shades, and white meaning there are none in that country. A user is able to hover their mouse over a country to display the country name and the number of groups located there. The second page, which can be navigated to via a dropdown menu, displays an interactive map of the world with dots plotted to show the specific latitude and longitude coordinates of each group. The color of the dot represents the project code with which the group is associated (essentially, which nonprofit the group was started by). In order to get a better sense of how these implementing organizations break down, there is a bar chart to the right side of the map displaying the count of groups for each different project code. There can also be groups with a project code of "None," most likely meaning that they are either independent of an implementing agency, or it was a group created to test the app. On this screen, the user is able to

scroll around the map, zoom in and out, and hover the mouse over a dot to display the project code number and the name of the implementing organization.

The theme of the second small server application is a demographic exploratory analysis. The goal of this platform is to provide multiple interactive ways of seeing how the population of DreamSave users breaks down in terms of four basic categories: age group, gender, education level and marital status. In this application, the user can select whether they would like to view a single-variable bar graph or a two-variable, grouped bar graph. This allows for the capacity to examine either one variable specifically, and how savings group members fall according to that category, or to examine two of the variables at a time, subdividing each grouping of the first category into groups of the second category. Additionally, the user can choose whether to view as a bar graph or a lollipop graph, according to their aesthetic preference—both have the same structure, but the former displays rectangular bars while the latter displays lines with a circle at the top. Finally, as an added feature, it is also possible to view a line graph that shows savings group creation over time. Although this graph is unrelated to the demographic analysis, it does provide an understanding of the rate of the uptake of the DreamSave application.

Implementing a Real-World Project: Lessons Learned

These two interactive data visualizations are the tangible result of the practical application section of this project. However, the unseen and perhaps even more significant result were the lessons learned through the experience of working on an out-of-classroom software project. Through implementing a data visualization program using real-world, DreamSave data, this project exemplified how the industry setting is different from the classroom setting in many important ways. Firstly, in the classroom, assignments are typically very controlled; professors will only give assignments that they know their students have the potential to succeed in.

Platforms, context, and instruction are always provided for school projects, and ideally, the professor already has an idea of what complications the students might face along the way, as well as how to get through them. In contrast, when developing something completely new in an industry setting, there is not typically anyone who already knows what the task will entail and can provide a foundation off of which to begin. Throughout the process of completing this project, various complications and considerations were identified that reflect what real-world software developers and data engineers might face in their work for an organization.

Additionally, this project has demonstrated that there is not always a smooth transition from the academic world of theory into the industry world of practice. This is because, in practice, there are outside factors and limitations that play into what implementation actually looks like. Becoming aware of these realities and learning to work through them has been a significant part of the journey of this project. Thus, the following sections of this report will attempt to identify and explain the some of these considerations, drawing on specific examples experienced by the author. The goal of documenting these lessons is to illustrate some of the challenges that those who work in software and data will face and be forced to overcome in the effort to bring theory into practice.

Lesson 1: Working with an Existing Technical Context

Sourcing Data.

When setting out to visualize data, one of the first hurdles that a developer must address is the question of how to source the necessary data. This involves thinking about where the data is physically stored, in what format it is being stored, and what protocols exist in order to pull it from the database and into the context which will be used for processing and/or visualizing. Unlike a school project, where a free, open-source dataset might simply be downloaded from the

internet, a real-world project requires specific data from a specific (and usually closed) source. A sufficient understanding of querying, networks, database management systems, authentication methods, etc. is necessary in order to set up a way to actually access the data that needs to be visualized.

For this project, the data source was a Cassandra database managed by DreamStart Labs and hosted in Google Cloud. Thus, it was necessary to be able pull data from Cassandra and process it using a local engine on the author's laptop computer. Also, in order to access the data remotely while still maintaining the security of the DreamSave database, this laptop had to be established as a recognizable and approved entity by their Google Cloud account. This was achieved through the use of both SSH (secure shell) keys and SSH port forwarding. To set up a connection to the database via SSH, it required generating a new SHH public and private key pair, and then uploading the public key to the DreamSave Google Cloud project. This task involved a few sophisticated details, which included formatting the public key file correctly to be recognized by the GCloud command line tools, being granted access to edit the metadata of the GCloud project, and resolving some other IP address-related technical snafus. However, having all of that established, access to the DreamSave database could be granted to the work laptop through presenting the private SHH key which corresponds to a validated public key in the GCloud project. From that point, a method called port forwarding was used to create a tunnel between the work laptop and the database server. Running the port forwarding command creates a configuration in which, whenever the client computer's localhost address makes a request of a certain port, the SSH client takes that request and sends it through a certain port of the server's address. Altogether, the use of SSH provided both the authentication to access DreamSave data, as well as the means of telling the local machine how to go about asking for it.

All of this to say, though the SSH connectivity approach used for this project is only one example of the methods that exist, the takeaway is that sourcing data from a remote, real-world database is a matter that must be thought about and planned with care. In this project, demystifying the problem of how to source the necessary data was a complicated and sometimes confusing process; however, it is a fundamental skill to have when working with data in industry. Also, having authentication procedures in place, though it can feel like a hassle, is imperative in today's technical environment, and helps to ensure that company data is secure and protected. Thus, the task of safely moving data through networks is something that a data engineer should understand and will often have to do, and it cannot be learned through downloading a free dataset from the internet for a school project.

Working with the Available Hardware.

Working in the software industry, it is impossible to ignore the matter of hardware. The two are intimately connected—software cannot run if there is not hardware. Likewise, software has to exist on hardware, and for that to happen, installation is required. For the purposes of this project, in order to be able to work with data and experiment with different visualization tools, it was often necessary to install various programs and packages onto the working computer. Similarly, companies that wish to stay current must continually make sure that the correct software is installed and updated on their machines, especially as software is constantly changing, and new tools are being developed all the time. In the case of this project, the only hardware available to utilized was a MacBook Pro laptop computer, so any new products that were needed had to be located in the correct version, downloaded and installed. Some examples of these include Apache Spark, GCloud command line tools, Python3, Hadoop, and Apache Zeppelin, as well as selection of packages for Python, including Livy, Matplotlib, Bokeh,

Basemap, Pandas, Geopandas, and others. Often, these products also had dependencies—this meaning that before installing the software intended for use, one has to download and install other programs or packages that it requires in order to run. The installation process can sometimes be simple, as with a basic Python package that can be installed using a pip command, but at other times, it can be more complicated. For instance, when there was a lack of good documentation provided, or when unexpected things happened on the particular machine, installation became a frustrating process. A lesson learned was that installing more technical, developer-tool-type software often requires a functional understanding of installation directories and the high-level processes of a machine's operating system. Consequently, this project illustrated how installation is not always a completely smooth process; however, it is a practical necessity to be able to work with the hardware available.

Another and perhaps more important consideration in learning to work with the available hardware is the limitation of physical resources. A significant portion of this project has been concerning big data and the challenges it presents, as well as technologies like the NoSQL Cassandra database and the Apache Spark data processing engine, which are tools designed to offer more efficient big data solutions. Putting this theory into practice, however, is something that cannot be done without the appropriate hardware. To illustrate, the original intention with this project was to utilize Apache Spark's Python API in order to process DreamSave data before feeding it into a visualization. However, as the only machine available for use was a laptop, Spark had to be installed and run locally, despite the fact that this software's main advantage comes from being optimized for distributed processing on a cluster. Thus, it is not surprising that the improved efficiency which Spark should provide was not apparent in this context—in fact, it turned out to be similar or even worse in runtime to other API options that were tried. Therefore,

though it is true in theory that Spark is a powerful and fast tool for big data processing, in practice where available hardware is a factor, this might not be the case. Thus, though this project demonstrated it on a small scale, hardware limitations such as processing power, memory space, etc. are simply the reality of real-world software projects.

Configuring the Development Environment.

As has been stated earlier, installing software is a necessary hurdle for developers who wish to use a new product. The obligatory next step to installation, however, is configuration. Configuration is the process of setting up variables that the installed software will use to both run correctly on the specific system, and to enable the user's custom needs and preferences. For instance, programs may need to know where to look for dependencies or which settings to use; likewise, developers may want to change the default tools that the program uses, or make it aware of any extra packages being included. Often, installing a program seems relatively simple in comparison to configuring it. Although there may (or may not) be decent default configuration values and documentation about how to adjust them, each case is different, so there is generally little to no precedent to look to for specific concerns. Advanced configuration requires an in-depth knowledge of how the program works behind-the-scenes, and what variables need to be changed in order to build the appropriate environment—requiring one to delve into the installation directories and configuration files in order to find the properties they are looking for.

With the various different products and tools used for this project, there were a handful instances in which significant time was spent demystifying the problem of how to configure the tool properly. One of those cases was setting up Apache Spark. To offer a simple example, the particular Spark library that this project intended to use was the Python SQL/DataFrame API, but in conjunction with Python version 3 rather than the default version of 2.7. So, the solution was

to specify a variable which pointed Pyspark (Spark's interactive shell for Python) to the path where Python3 was installed on the machine. That way, when Spark booted up, it would be able to read that variable and know where to look for a Python instance to use. Also, another slightly more complicated example was figuring out how to configure Spark with the capacity to read data from Cassandra. This involved telling Spark to include an extra package which enabled the conversion of Cassandra tables into Spark DataFrames, specifying an IP address of which to make requests for data, and providing a username and password for authenticating access to Cassandra.

Once Spark itself had been set up, there was the remaining matter of configuring the other tools to work in conjunction with Spark. The first of these was Apache Zeppelin, which is a notebook-style development environment that is built for data analytics and visualization. This software executes blocks of code one at a time, using an interpreter that the programmer can select, and makes it possible to experiment with different languages and produce graphs quickly. Using Zeppelin to run Spark code requires configuring an internal Pyspark interpreter, providing it with the location of a Spark installation and the correct properties to build a SparkContext (client for the execution environment). The second tool that was utilized in this project and had to be configured for Spark was Apache Livy. Livy is REST API service built to enable web applications to submit jobs to Spark without the use of a local SparkContext. In this case, Livy was configured inside of the program code, rather than by setting variables in the package configuration files: this was achieved by passing the appropriate Spark properties into the call to create a Livy Session object. This way, had it been necessary to create more than one Livy Session, it would be possible to give each of them different settings and permissions and still run them simultaneously.

All in all, the given examples of software configuration involved in this project are very specific and slightly complex. However, they are explained for the purpose of demonstrating how in the context of industry, it is often necessary to spend considerable time configuring programs for a project's specific needs. Also, for a variety of reasons, it is not always a straightforward task—it can be difficult to track down the cryptic configuration files, environment variables may need to be set or adjusted, unexpected errors can occur, or whatever else the case may be. Especially for those with less experience or who are setting up a program for the first time, figuring out proper configurations can be a slightly daunting task. Nonetheless, it is an important skill for a software engineer to become comfortable getting into the weeds of installation and configuration, as it is skill that will surely be utilized throughout their career.

Lesson 2: Navigating the Trial and Error Process

Experimenting with Unfamiliar APIs.

In any new and experimental software project, the discovery of the best methods and tools to use is a significant part of the process. The world of software is very quickly, and new and better tools are being developed all the time. Although it is incredibly important for developers to learn to fully leverage the tried-and-true tools that are robust and well-developed, they also have to be able to stay current with fresh ideas and cutting-edge technologies. Often times, a new project will call for capabilities that the developer or the organization that they are a part of have never implemented before, requiring them to search for and try out different tools that are available to accomplish the task. This involves reading a lot of API documentation to get an idea of how a certain tool can be used; however, the true test is to actually install and experiment with the software, with the aim of determining if and how it will work with the

specific context and dataset. This discovery process may be time-consuming, but it is a valuable investment to know that in the end, the best possible solution will have been found.

In this project, there were certain tools identified to be used from the beginning: Apache Spark, Apache Zeppelin, Python, and Pyspark's SQL/DataFrame library. This reason for the choice of these products was a combination of the author's prior familiarity with Python, the desire to be consistent with what DreamStart labs uses, and motive to gain practical experience with these tools as well-regarded big data analysis technologies. However, what had yet to be determined was which Python API to use to create the desired graphs. There is an abundance of packages available to a Python programmer, each of which provide different ways of going about making data visualizations. Matplotlib, Bokeh, Basemap, Cartopy, Seaborn, Pandas, Geopandas, and Plotly are all APIs which were explored at some point during the course of this project. Many of these were not only researched, but also installed and experimented with. Within this realm, a particular challenge was discovering the logistics of creating map visualizations. There are many different strategies to create map visualizations, depending upon the options provided by that the API that is being used. At times, it involves downloading a shapefile of the geographic area to be represented from the internet, and then displaying it based on the latitude and longitude information contained in the dataset along with a chosen map projection type. Other times, the API will have geographic data already built in, and it is simply a matter of calling a function—though this allows less capability for fine-tuned control. All of this goes to show that there are many different ways of approaching data visualization implementation, even within a limited scope. Additionally, it takes time and experimentation to learn at least the basics of how to use an API, whether or not that it ends up being a part of the final product. And, as was occasionally learned the hard way in this project, it is important to

determine ahead of time whether or not the goal can be accomplished using a certain API before fully committing to it. Learning and implementing new tools comes down to a research, trial and error process, and developers have to be realistically aware of the time commitment that is required.

Lost Investments and Starting Over.

The unfortunate part of entering into the unknown waters of a new software project is that often, through trial and error, something is discovered not to work, and it necessitates a change of course or even starting over. The loss of a time (and/or money) investment is often a reality for industry projects, even when organizations do what they can do minimize and mitigate that risk. Especially when entering into a new realm, the expectation that everything will go right the first time is almost never met. However, every failure is a lesson learned, and the hope in these situations is to make the most of that learning and move forward in the right direction. In the efforts to create data visualizations for this project, this loss of time investment and the discarding of work was experienced more than few times. Yet, eventually, it all led up to the finding of a working solution. For example, there was one instance in which at least half of a day was invested into troubleshooting a problematic installation of the Basemap library for Python (it needed to be built from the source, and the process did not go smoothly). However, it was later decided that Basemap was not the best tool, and all of the effort spent installing and building the package was for naught. Setbacks like this are an unfortunate reality of software projects, both in academics and probably even more so in industry.

At various times throughout development, this project appeared as if stuck in a trial-and-error cycle. It began with trying out a new tool or strategy that seemed promising, then getting stuck on some detail or limitation of the API, and finally finding a fix or workaround for that

issue, only to discover a new complication that resulted from solving the previous one. Take, for example, this sequence of events: as previously described, at one point in the project, the Pyspark interpreter in Apache Zeppelin was being used as a platform for processing data and coding visualizations. During research, the Python package called Bokeh was discovered, which is a visualization library that contains many helpful constructs for embedding interactivity into graphs, and it appeared to be an optimal tool for creating a savings group map in Zeppelin. However, upon experimentation, it was discovered that interactivity in Bokeh requires writing custom Python callback functions, which are an impossibility in the Zeppelin notebook. A work-around that Bokeh provides for this situation is to replace the Python callbacks with embedded custom-JavaScript (JS) callbacks; however, this also proved to be unsuccessful. It was then decided to abandon both the customJS and the Zeppelin notebook, and instead create a Bokeh server application, in which Python callbacks would be permitted.

Nonetheless, a new complication arose: namely, that it would be impossible to use the Pyspark interpreter to run the Bokeh server, and thus impossible to host a SparkContext in the application. To solve this new issue, an Apache Livy client was installed, which has a programmatic API for exporting data processing jobs to a remote Spark client. (In the case of this project, where the application and Spark both ran locally, in practice that meant the Livy server simply looped the jobs back to the same machine.) After all of this, the application ran successfully, but as it turned out, far too slowly to be practical. In the end, the use of Spark was abandoned altogether in favor of the DataFrame API from Python's Pandas, which proved to be much faster in the context of this project.

This long, complicated, up-and-down story serves to show that the trial and error process can be messy. The cycle of challenges faced here demonstrates that it is important to understand

how different tools can and cannot work together, and to try to identify what limitations might be faced before deciding to head down a certain path. The hours spent on work that did not become part of the final product were significant, but they did provide a taste of the reality of navigating an unprecedented industry project, and they developed skills of problem-solving and perseverance.

Lesson 3: Dealing with Dirty Data

As the title of Hernández and Stolfo's 1998 article simply and straightforwardly states, "real-world data is dirty." The concept of dirty (or inaccurate) data is a topic of great importance in the data management world. Because of the amount of weight that is given to data-driven insights, the integrity and reliability of data is of utmost importance in order to guarantee that the information it provides is trustworthy and accurate. Especially in the era of big data, where information is collected with ever-growing speed, size, and variation, ensuring the reliability of that information is an increasingly challenging and monumental task. If a dataset is not clean, creating visualizations in order to understand that dataset can result in graphs that are unhelpfully skewed, hinderingly sparse, or that communicate misleading information. Even worse, dirty data can be debilitating to an organization's operations if left unaddressed (Wheatley, 2004). These concerns have given rise to entire and complex fields of professional study surrounding the governance of data: there are full-time jobs for people who manage an organization's data integrity, companies which specialize in the business of creating data-cleaning software, etc. Thus, an important consideration for anyone who works with real-world data is to consider the ways in which it might be corrupted and determine the best way to address those things to increase the data's value.

Data can become dirty in many ways and for all kinds of reasons. For example, data can be confused when systems merge, redundancies can exist in the database that cause anomalies, there can be missing information in the dataset that makes a subset of the population go unaccounted for—and the list goes on. These inconsistencies can be caused by anything from data entry mistakes, to technological failures to malicious activity (Hernández and Stolfo, 1998). Thus, it follows that there are many different possibilities for what the appropriate response to bad data found in a database is. Though the job of database cleaning is an important one with many strategies and complexities, the focus and scope of this project is more interested in the various concerns of visualizing datasets that contain dirty raw data. These concerns include recognizing, analyzing, and attempting to reconcile the skews that appear in graphs as a result of incorrect or incomplete data.

In order to illustrate some of the types of data challenges that visualization creators must contend with, we will examine a few issues that were uncovered amidst working through this project of visualizing DreamSave data. To begin with, while making geographic visualizations, it was found that the latitude and longitude coordinates recorded for each savings group did not always match up with the country that the group was coded for. At first this appeared to be because of missing values; however, removing all groups that had a null value for longitude or latitude from the map did not eliminate the problem. It was soon discovered that the source of the discrepancy was that a significant number of the groups were recorded to be located at the coordinates (0, 0), which happens to be a little east of the Atlantic coast of Gabon, Africa. Clearly, these groups do not take place in the middle of the ocean—it much more likely is that for one reason or another, these values never got reassigned from their default value of zero. One possibility is that this was a result of insufficient GPS data to locate the group, but whatever the

case may be, it is misleading in a map visualization to display a data point where it should not really be located. Another example of potentially dirty data was found in the project coding for each savings group. The project code represents the nonprofit-led program that a group is associated with, if one exists. A null value for this attribute could mean a number of different things: maybe the group is not associated with a non-profit, the information was accidentally or purposefully left out by the app users, the project code was unknown to the users, or even that this group entity in the database does not truly exist, but rather is a fabricated group made to test the application. As it turned out, these groups with a null project code made up the far majority of the groups in question—and building a bar graph of group counts by project code resulted in an unsuitably tall "null" bar which made the other valid project codes hardly visible. Thirdly, a similar case to this one (in which null or missing values far outnumber the legitimate values) was found in attempting designate all the savings groups members to an age group as part of the demographic analysis. As the database does not store ages, this required making a calculation based off of the person's birthdate, which turned out to be missing for about 90% of the population. Having a missing value at such a rate exposes immediate problems. A visualization of savings group member ages will be severely limited if it is impossible to calculate age for the vast majority of the population.

Each of these instances of missing data lead to questions about how to approach visualization and analysis of the DreamSave dataset. On one hand, it is not practical to include inaccurate or outlier values in graphs, because the resulting image will be either incorrect or improperly scaled, and it will be impossible to generate any real insight from the data—which, as has been established, is where the value of data lies. On the other hand, simply excluding the corrupted data is not a perfect solution either, because it oversimplifies and begins to

misrepresent the underlying population. Because it is often impossible to know the exact cause behind corrupted data, there is no one way to clean a dataset that will be fair in all situations. Take, for example, the case of the missing birthdates. The reason behind these missing values is largely a guessing game for the data engineer—are they due to some unintentional error, or are people simply choosing not to enter their birthdates? If they chose not to enter a birthdate, could it be because they want to keep that information private, because they don't think it is important, or maybe because they simply do not know when it is? Yet another possibility is that it represents another test group, and the application tester simply did not bother to enter birthdates for all the placeholder group members they created. Any and all of these options are possible, but simply removing these values from the dataset before visualizing creates a not entirely accurate picture of what the savings group population looks like.

Thus, the issues of dirty data are complex, and must be handled with care. At least in the context of this project, it has been necessary to clean the data to some extent and exclude certain outliers in order for the visualizations to be readable. However, by beginning to ask questions about the causes of these outliers, it becomes clear that these data points should not be simply disregarded altogether. Therefore, as a general rule, this project chooses to reconcile these things by excluding the entities that need to be excluded in order for the graph to serve its purpose, but also incorporate some kind of disclaimer which describes the data that was removed. It is not a perfect solution, but it is reflective of the kinds of compromises that need to be made when dealing with real-world, dirty data.

Lesson 4: Working in Partnership with an Organization

One final aspect of industry projects that makes them a significantly different experience from school projects is the added component of organizational collaboration. In academic work,

projects are often either individual or small group endeavors. The amount of dependency on other people is limited, and the professor is responsible for setting up the students with all the information and resources they need in order to complete the assignment. In short, academic projects take place in a relatively controlled environment, and even though group projects do begin to emulate the experience of working in a team, the shortcomings of one group will generally not affect the success of another. When working in industry, however, the tie between the individual and the organization is much closer. Different teams will often collaborate with each other and rely on each other's work; lower-level employees depend on their superiors to make key decisions, and superiors depend on their employees to be productive. The added intricacy of existing within an organization often adds an extra layer of complexity to projects.

Through partnering with DreamStart Labs, this project created a small-scale experience of what it is like to work with an organization, and to have progress be affected by the larger organization. At times, this resulted in unexpected changes in direction; for example, it had initially been suggested that it would be necessary to develop a script which produces fabricated test data. Doing this would provide a larger quantity of data to work with when creating visualizations, as DreamSave had not yet been taken up by a large number of actual savings groups. However, after some time of playing with the idea and struggling to come up with an implementation strategy, DreamStart had already decided to begin their own data generation predictive model, and so it would no longer be necessary to perform this task. Thus, an organizational decision that was external to this project caused a change in direction, which is something that always comes with at least a small setback. Another example of project progress being affected by the larger organization was in the matter of getting access to DreamSave's data. Originally, it had been planned that DreamStart would do all the configuration of the

Cassandra, Spark and Zeppelin stack on their end, and beginning to work with the data would simply be a matter of connecting to their setup via the cloud. However, some technical difficulties and more pressing distractions caused this process to be delayed, followed by some other unforeseen complications. Ultimately, the delays compelled this strategy to be discarded completely, in favor of installing a local version of Spark on the personal working laptop. Amidst the waiting for solutions from the DreamSave team and the changing plans, months went by in which DreamSave's data was completely inaccessible for this project. Thus, having to depend upon an organization in order to make progress was at times a challenge, simply due to evolving circumstances.

Also, when discussing the complications of working in partnership with an organization of other people, it is essential to at least briefly mention the matter of communication. Communication, though simple in theory, is something that commonly becomes a challenge for organizations. This is especially true, as in the case of this project, when the majority of communication happens remotely. This manifested in the inconvenience of different schedules, having to wait for answers to messages, and being unsure of whether or not a certain struggle had ever been experienced by the team or warranted sending a message to ask for help. Also, when working remotely, it was much more difficult to become acquainted with the company systems, as it was usually not possible to observe and benefit from the experiences of others within the organization. Of course, in this technical era, working remotely is becoming more and more common in the field of software development, and there are quite a few quality tools available to teams as means of online messaging and meetings: DreamStart Labs, for example, makes use of primarily Slack and Zoom. These communication platforms are certainly helpful, but ultimately, they are not able to fully replace face-to-face interaction. For example, a technical difficulty

might take five minutes to solve when the expert is in the same room as the individual with the issue, but it takes significantly more time when attempting to troubleshoot via online chat or video call. All in all, the dependency upon other teams and the logistics of communication are a major component in industry projects, and as a software engineer, it is important to build up the skills to navigate these situations.

Conclusion

The four lessons identified here, and their various nuances, represent some of the common complicating factors that are a part of the process of data visualization, and more generally, the software development process in an industry setting. Placing the focus on the challenges in this way can have the effect of emphasizing the negative aspects of a project, and it is true: had these not occurred, a larger and more robust data visualization application could have been created. However, reporting the issues that arose is also a way of documenting everything that had to be overcome in order to end up with a working product, and demonstrates both valuable learning and eventual prevailing over obstacles, two very positive aspects of the project. It brings common issues to light so that, in future projects, inevitable complications can be planned for and met with an arsenal of strategies to address them.

In this project as a whole, the concepts of big data, data visualization, and various big data technologies have been theoretically examined. Additionally, a light has been cast onto the savings group model for development, identifying it as a powerful, revolutionary concept whose success could be augmented by the application of big data analysis. By engaging with the DreamSave application data to put this concept into practice, the project has given a picture of what it looks like to take the initial steps in synthesizing these fields. Although by no means is

the work here complete, the effort has provided a foundation for continued exploration of the future of big data and technology in the field of international development.

Identifying Areas for Further Research

There are many distinct topics that have been identified and surveyed over the course of this report, and though some important points are highlighted for each one, they all leave a significant amount of room for further in-depth research and analysis. For example, one could take a deeper dive into the complexities of big data analysis, NoSQL databases, dirty data, visualization techniques and tools, or big data processing engines. It would also be possible, as many have done, to further examine the intricacies of the savings group model, its varying implementations, and the impact that it has had on the field of development. However, the more unique area of study that this report brings to light is the union of big data technology with international development by means of collecting and analyzing savings group data. One could continue the effort by taking a more specific look at the variables involved in the implementation of savings groups, such as constitutional policies, group structure and formation, etc., and examine how this data can be visually compared to the success level of the group. Likewise, another interesting and important topic within this area of study would be how to address the limitations of big data coverage in developing areas. The relative lack of infrastructure in third-world countries creates a unique challenge when attempting to create inclusive technology, and as this project has demonstrated, can contribute to the incompleteness of datasets. However, taking into account the speed and force with which big data has grown and spread, it would be interesting to examine what the future may look for digital data in development. Anyone who wishes to continue with these ideas will have an abundance of possible questions to work off of.

References

Abramson, C., & Dohan, D. (2015). BEYOND TEXT: USING ARRAYS TO REPRESENT AND ANALYZE ETHNOGRAPHIC DATA. Sociological Methodology, 45, 272-319. Retrieved from http://www.jstor.org.pointloma.idm.oclc.org/stable/44289461

Albanna, B., & Heeks, R. (2018). Positive deviance, big data, and development: A systematic literature review. The Electronic Journal of Information Systems in Developing Countries, 85(1). doi: 10.1002/d2.12063

Allen, H. (2006). Village Savings and Loans Associations — sustainable and cost-effective rural finance. Small Enterprise Development, 17(1), 61–68. doi: 10.3362/0957-1329.2006.009

Ashe, J., & Neilan, K. J. (2014). *In Their Own Hands: How savings groups are revolutionizing development*. San Francisco: Barrett-Koehler Publishers, Inc.

Batistič, S., & van der Laken, P. (2019). History, Evolution and Future of Big Data and Analytics: A Bibliometric Analysis of Its Relationship to Performance in Organizations. British Journal of Management, 30(2), 229–251. doi: 10.1111/1467-8551.12340

Brath, R., & Jonker, D. (2015). Graph analysis and visualization: Discovering business opportunity in linked data. Retrieved from https://ebookcentral-proquest-com.pointloma.idm.oclc.org

Brown, M. (2013, September). Big Data Modelling with Cassandra. *Windy City Rails*. Talk presented at 2013 Windy City Rails Conference, Chicago, IL, USA. Retrieved from https://vimeo.com/76197228

Cloudera. (2019, September 4). Apache Spark. Retrieved October 15, 2019, from https://www.cloudera.com/products/open-source/apache-hadoop/apache-spark.html.

Damji, J. (2016, July 14). A Tale of Three Apache Spark APIs: RDDs vs DataFrames and Datasets. Retrieved October 16, 2019, from https://databricks.com/blog/2016/07/14/a-tale-of-three-apache-spark-apis-rdds-dataframes-and-datasets.html.

Datastax, Inc. (2019, September 24). Data modeling concepts. Retrieved 2019, from https://docs.datastax.com/en/archived/cql/3.3/cql/ddl/dataModelingApproach.html.

DreamStart Labs, Inc. (2019). DreamStart Labs - Technology that Empowers. Retrieved from http://www.dreamstartlabs.com/.

Duffy-Tumasz, A. (2009). Paying back comes first: Why repayment means more than business in rural Senegal. *Gender and Development*, 17(2), 243-254. Retrieved from http://www.jstor.org.pointloma.idm.oclc.org/stable/27809227

Falaki, H. (2015, March). *Spark Summit East*. *Spark Summit East*. New York, NY. Retrieved from https://databricks.com/session/visualizing-big-data-in-the-browser-using-spark

Firican, G. (2017, February 8). The 10 Vs of Big Data. Retrieved from https://tdwi.org/articles/2017/02/08/10-vs-of-big-data.aspx

Havers, M. (1996). Financial Sustainability in Savings and Credit Programmes. *Development in Practice*, 6(2), 144-150. Retrieved from http://www.jstor.org.pointloma.idm.oclc.org/stable/4028978

Hernández, M., A., & Stolfo, S. J. (1998). Real-world data is dirty: Data cleansing and the Merge/Purge problem. *Data Mining and Knowledge Discovery*, *2*(1), 9-37. doi: http://dx.doi.org.pointloma.idm.oclc.org/10.1023/A:1009761603038

Hilbert, M. (2015). Big Data for Development: A Review of Promises and Challenges. Development Policy Review, 34(1), 135–174. doi: 10.1111/dpr.12142

Holtz, Y., & Healy, C. (2018). From Data to Viz. Retrieved November 4, 2019, from

https://www.data-to-viz.com/index.html

Ksoll, C., Lilleør, H. B., Lønborg, J. H., & Rasmussen, O. D. (2016). Impact of Village Savings and Loan Associations: Evidence from a cluster randomized trial. *Journal of Development Economics*, 120, 70–85. doi: 10.1016/j.jdeveco.2015.12.003

Landset, S., Khoshgoftaar, T. M., Richter, A. N., & Hasanin, T. (2015). A survey of open source tools for machine learning with big data in the Hadoop ecosystem. Journal of Big Data, 2(1), 1-36. doi: http://dx.doi.org.pointloma.idm.oclc.org/10.1186/s40537-015-0032-1

Macwan, J. (2018, March 16). Apache Spark Pitfalls: The Limitations of the Big Data Processing Giant. Retrieved October 17, 2019, from https://www.smartdatacollective.com/apache-spark-pitfalls-limitations-big-data-processing-giant/.

Madison, M., Barnhill, M., Napier, C., & Godin, J. (2015). NoSQL database technologies. Journal of International Technology and Information Management, 24(1), 1-I. Retrieved from http://pointloma.idm.oclc.org/login?url=https://search-proquest-com.pointloma.idm.oclc.org/docview/1757727794?accountid=13223

Marr, B. (2016). Big data in practice: How 45 successful companies used big data analytics to deliver extraordinary results. Retrieved from https://ebookcentral-proquest-com.pointloma.idm.oclc.org

McDonald, C. (2018, October 17). Spark 101: What Is It, What It Does, and Why It Matters. Retrieved March 12, 2019, from https://mapr.com/blog/spark-101-what-it-what-it-does-and-why-it-matters/

Rosling, H., Rosling, O., & Rosling Ronnlund, A. (2018). Factfulness. New York, NY: Flatiron Books

Ruecker, S., Hodges, P., Lokhadwala, N., Ching, S.-Y., Windsor, J., Hudson, A., & Rodriquez, O. (2015). Why Experimental Interfaces Should Include an Application Programming Interface. *Scholarly and Research Communication*, *6*(2). doi: 10.22230/src.2015v6n2a220

Sahay, A. (2017). *Data visualization, volume ii: Uncovering the hidden pattern in data using basic and new quality tools*. Retrieved from https://ebookcentral-proquest-com.pointloma.idm.oclc.org

Schreiner, M., & Vonderlack, R.M. (2002). Women, Microfinance, and Savings: Lessons and Proposals. *Development in Practice*, 12(5), 602-612. Retrieved from http://www.jstor.org.pointloma.idm.oclc.org/stable/4029405

Simon, P. (2014). The visual organization: Data visualization, big data, and the quest for better decisions. Retrieved from https://ebookcentral-proquest-com.pointloma.idm.oclc.org

Sharma, A. (2019, September 5). Introduction to NoSQL. Retrieved from https://www.geeksforgeeks.org/introduction-to-nosql/.

Smallcombe, M. (2019, August 23). SQL vs. NoSQL: How Are They Different and What Are the Best SQL and NoSQL Database Systems? Retrieved September 27, 2019, from https://www.xplenty.com/blog/the-sql-vs-nosql-difference/.

The Apache Software Foundation. (n.d.). What is Cassandra? Retrieved October 8, 2019, from http://cassandra.apache.org/.

Tutorials Point. (n.d.). Cassandra Tutorial. Retrieved 2019, from https://www.tutorialspoint.com/cassandra/index.htm.

Urma, R. G. (Producer.) (2017). *Cambridge Spark Webinar: Getting Started with Spark and*

Zeppelin in Python. [Video Webinar]. Retrieved from https://www.youtube.com/watch?v=h787df9Za1w

Valkanova, N., Jorda, S., & Vande Moere, A. (2015). Public visualization displays of citizen data: Design, impact and implications. International Journal of Human-Computer Studies, 81, 4-16. doi: 10.1016/j.ijhcs.2015.02.005

Wheatley, M. (2004). Operation clean data; cleaning dirty data is not just a matter of mastering the technical challenges. it requires making sure your staff is working closely with the business every step of the way. *Cio*, *17*(18), 1-88. Retrieved from http://pointloma.idm.oclc.org/login?url=https://search-proquest-com.pointloma.idm.oclc.org/docview/205948184?accountid=13223

Wickrama, K., & Keith, P. (1994). Savings and Credit: Women's Informal Groups as Models for Change in Developing Countries. *The Journal of Developing Areas*, 28(3), 365-378. Retrieved from http://www.jstor.org/stable/4192358

Willems, K. (2017, March 28). Apache Spark in Python: Beginner's Guide. Retrieved October 16, 2019, from https://www.datacamp.com/community/tutorials/apache-spark-python.

Yau, N. (2018). Ask the Question, Visualize the Answer. Retrieved November 3, 2019, from https://flowingdata.com/2018/10/17/ask-the-question-visualize-the-answer/.

Zaharia, M., Xin, R. S., Wendell, P., Das, T., Armbrust, M., Dave, A., … Stoica, I. (2016). Apache Spark: A Unified Engine for Big Data Processing. *Communications of the ACM*, *59*(11), 56–65. doi: 10.1145/2934664