Developing Automated Applications for Clustering and Outlier Detection: Data Mining Implications for Auditing Practice

by

Paul Eric Byrnes

A Dissertation submitted to the

Graduate School-Newark

Rutgers, The State University of New Jersey

in partial fulfillment of the requirements

for the degree of

Doctor of Philosophy

Graduate Program in Management

written under the direction of

Professor Alexander Kogan

and approved by

____
Dr. Alexander Kogan

____
Dr. Miklos Vasarhelyi

____
Dr. Rajendra Srivastava

____
Dr. Kevin Moffitt

Newark, New Jersey

October, 2015

Abstract of the Dissertation

Developing Automated Applications for Clustering and Outlier Detection: Data Mining Implications for Auditing Practice

By Paul Eric Byrnes

Dissertation Chairman: Prof. Alexander Kogan

Occupational fraud is viewed as a growing, global problem, and solutions are thus needed. Furthermore, since passage of Statement on Auditing Standards (SAS) 99, auditors have been held to a higher standard relative to audit quality. More specifically, auditors are now required to consider the risks of material misstatement due to fraud throughout the entire audit process. Interestingly, clustering has emerged as one method for addressing this challenge.

Unfortunately, a set of difficulties exists in implementing data mining in practice, such as complexities relative to data pre-processing, algorithm selection, and model evaluation schemes. Given this, the traditionally trained auditor is ill-equipped to effectively perform clustering in the context of the financial statement audit. Given the likelihood that clustering will become ubiquitous in the auditing and accounting domains of the future, accounting professionals should be positioned to effectively use data mining in fulfillment of their responsibilities. One possibility for achieving this involves substantial *automation of the clustering routine. In this way, many of the historically manual decision points within the process can be eliminated, thus making it a more user friendly task. In so doing, practitioners could then focus on problem investigation and resolution, instead of being burdened with technical nuances of clustering operations.*

In this dissertation, efforts are made to progressively automate clustering and outlier detection. This is done via auditing credit card customer data. First, cluster analysis is performed to generate an initial set of partitions. Next, each group is evaluated using various mechanisms to note whether nested clusters exist. Following this, a method for identifying irregularities is proposed and implemented. Overall, results demonstrate clustering and outlier detection can provide utility in the auditing of organizational assets. In conclusion, findings are synthesized and two distinct applications are created. These are provided as implementable artifacts as well as proofs of concept demonstrating feasibility of automating clustering and outlier detection routines. It is hoped auditors see value potential in this type of software, and ultimately find such programs to offer both ease of use and perceived usefulness when investigating fraud in audit engagements.

ACKNOWLEDGMENTS

I have many people to credit for facilitating my continued development as an individual and scholar as well as being instrumental in my completing this dissertation and associated Ph.D. program. First, I am thankful and forever grateful for the efforts of Dr. Alexander Kogan, Dr. Miklos Vasarhelyi, Dr. Bill McCarthy, and Dr. John Erkkila. Their insights, support, and guidance were essential in helping me to succeed as a Ph.D. student at Rutgers University.

I also sincerely acknowledge and thank my remaining dissertation committee members, Dr. Kevin Moffitt and Dr. Rajendra Srivastava, for their valuable efforts and guidance throughout the dissertation process. I truly benefitted from their assistance, advice, and participation relative to my research pursuits.

I am additionally thankful for all of my colleagues and friends at Rutgers University. Without question, the supportive and collaborative relationships and atmosphere contributed significantly to my development, advancement, and success. In particular, I thank Barbara Jensen, Abdullah Al-Awadhi, Steve Kozlowski, and Deniz Applebaum for their consistent willingness to offer assistance, advice, insight, and encouragement.

Finally, I am grateful to God and my family and friends outside of the Rutgers University context for helping me to maintain perspective and stay the course, not only in my academic and career endeavors, but, more important, in everyday life.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF ILLUSTRATIONS

CHAPTER 1: INTRODUCTION

1.1 Introduction and Motivation

Occupational fraud is a problem for which solutions are needed. In fact, the estimated impact of this phenomenon has increased in recent times. More specifically, between 2009 and 2013, estimated global losses attributable to occupational fraud rose from about $2.9 to $3.7 trillion (ACFE, 2010; ACFE, 2014). This increasing pattern also applies in the United States, prompting a sense of urgency, particularly when considering that efforts have been underway for well over a decade in the accounting profession relative to mitigating the incidence of fraud.

For example, until 2002, auditors were not specifically required to consider the risk of fraud in financial statement audits. However, events such as the egregious corporate scandals at Enron, WorldCom, and Tyco prompted a significant shift in stakeholder expectations of auditors and management. Among other things, the enactment of SAS 99 in December 2002 immediately held auditors to a higher standard. More specifically, it requires auditors to maintain professional skepticism relative to the risk of material misstatement attributable to fraud (Harding, 2006). Consequently, auditors now have the obligation to actively embed fraud detection and investigation within the entire financial statement audit process, from planning the engagement through completion of the audit.

Since 2002, additional supplemental audit guidance has been formulated to provide more detailed assistance concerning how auditors might fulfill their duties in terms of considering the existence of fraud in financial statement audits. A common thread in such guidance entails the use of advanced data analytics, including but not limited to data mining methods. However, while it is encouraging to see audit standards promoting adoption and usage of such analysis tools, getting the audit profession to actively move forward with this agenda appears somewhat problematic.

While certain areas such as computer science, information technology, and even marketing might be considered progressive in embracing sophisticated data analysis technologies, the same cannot be said of the accounting profession. Instead, the tendency has been to cling to traditional methods and processes to the extent feasible, and there is a definite rationale for this. First, people often are naturally resistant to change, and this phenomenon can be ingrained in many organizational cultures. In managing change within these contexts, the change agent (e.g., partner in an accounting firm) serves as an instrumental force in encouraging others to embrace new approaches and techniques (Van de Ven, 1986; Kanter, 1983). In doing so, several issues are important, including achieving buy-in from affected individuals, maintaining employee input and involvement relative to the proposed change initiative, and ensuring that sufficient training is provided. Interacting with this, the technology itself is a force influencing whether or not change will be embraced. For example, the Technology Acceptance Model (TAM) helps explain the extent to which a new technological development is likely to be adopted. The two primary constructs in the TAM consist of perceived ease of use and perceived usefulness (Davis, 1989; Davis, 1985), and, while additional factors have been shown to be important, the TAM has nevertheless offered explanatory value in predicting the extent to which technology will be embraced (e.g., Tam and Ho, 2007; Wang and Benbasat, 2005). Incidentally, the effective use of data mining historically entails many considerations and complexities regarding data preprocessing, algorithm selection, parameter optimization, and model evaluation. Consequently, the vast majority of accounting practitioners would perceive ease of use as extremely low in this setting. Consequently, this problem must be corrected if the profession is expected to both adopt and use data mining tools. To assist in starting the conversation about and initially confronting this, the implemented processes in this paper are substantially automated in the R programming environment, and all associated code is provided in the appendices. In this way, complexity concerns are at least mitigated, and readers will have actionable information they can immediately and readily adopt or adapt for use in clustering and/or outlier detection initiatives.

Second, the accounting profession is perpetually preoccupied with and vigilant about litigation risk, and this position no doubt arises from legislation and standards as they currently exist. For example, if full population testing of events and transactions is attainable via automated technologies, external auditors must immediately be concerned about increased liability potential in cases where material misstatements are not discovered. In reality, this is completely dysfunctional. If auditors perform their engagements in good faith, with due diligence and professional care consistently applied, and maintain thorough working papers documenting all pertinent audit work, litigation risk should not be of any greater concern in cases where full population testing is conducted. If technology can cost-effectively improve audit quality, incentives should be in place to encourage its adoption and usage. Therefore, to maintain pace with the ever-changing business and technological landscapes, legislation and audit standards must continuously evolve in a corresponding manner.

Incidentally, prior research has relied upon clustering for fraud detection in a wide variety of settings. For example, it has been applied to identify fraud relative to money laundering (Liu et al., 2011; Khac & Kechadi, 2010; Zhang et al., 2003), credit card issues (Jyotindra & Ashok, 2011; Wu et al., 2010; Hao et al., 2010; Panigrahi et al., 2009; Tasoulis et al., 2008), insurance (Thiprungsri & Vasarhelyi, 2011; Ghani & Kumar, 2011; Xiaoyun & Danyue, 2010; Jurek & Zakrzewska, 2008; Zhang et al., 2006), and financial statement activities (Glancy & Yadav, 2011; Deng & Mei, 2009; Virdhagriswaran & Dakin, 2006). Without question, clustering has demonstrated effectiveness in locating irregularities, including but not limited to fraud. Consequently, it is not surprising that current audit guidance recommends the use of clustering by auditors in the fraud discovery process. Moving forward, efforts are needed to facilitate the actual adoption and usage of this valuable tool.

In this paper, the ultimate objective is to develop a comprehensive and seamless clustering and outlier detection program or set of applications that auditors (and others) may use or adapt for use in better meeting stakeholder expectations. This will be accomplished in a systematic fashion in the context of profiling the credit card customers of a large banking institution based upon a relevant subset of characteristics provided by the entity. In achieving this, it is hoped that useful findings are generated that the company can use in managing customers and improving operations. More important, it is intended that this research will successfully communicate the potential benefits and advantages of data mining for accountants and auditors seeking to fulfill their professional responsibilities in the evolving real-time global economy.

CHAPTER 2: CUSTOMER BASE CLUSTERING

2.1 Background

An area of data mining offers one potential set of solutions for customer profiling and outlier detection as introduced above. Fayyad et al. (1996) note that clustering is a well-established approach for finding meaningful patterns in data. Furthermore, clustering has been effectively utilized in addressing an extensive array of business issues, including customer segmentation (Garla et al., 2012; Tan et al., 2006). Therefore, clustering is certainly a suitable methodology to employ in the current study, but several considerations must be made prior to deciding on a specific strategy.

In a generic sense, cluster analysis places data into groupings that are beneficial, such that each object is comparable to items in the same cluster and different from objects assigned to other groups (Tan et al., 2006). Numerous methods exist for achieving this outcome, and no single algorithm is regarded as universally superior. In fact, the most appropriate method might often be a function of properties of the data being analyzed (Alpaydin, 2010). In addition, each algorithm employs a set of user-defined parameters, and these are often challenging to optimize. For example, K-means requires the user to specify both the number of clusters and initial seed value. While there might frequently exist some intuition concerning a range for the probable number of groups, the same cannot be said about the initial seed field. In fact, initial seed is something for which the user likely has no *a priori* information. Also, the available spectrum of values for this variable is excessive. For example, Weka allows users to select an initial seed of any non-negative integer between zero and 999,999,999, although each value will certainly not produce a distinct model. Nevertheless, a brute force approach to determining the optimal seed is infeasible, particularly when taken in conjunction with the number of clusters parameter. Using Weka as an example, with number of clusters ranging from three to ten, eight billion distinct K-means models would need to be created to exhaust all possible combinations of user specified parameter settings. In addition to issues of parameterization, each clustering method has a set of advantages and disadvantages, and these must be weighed in some fashion prior to finalizing algorithm selection.

As Garla et al. (2012) point out, K-means is one of the most frequently used clustering techniques. This is primarily attributable to its relative simplicity, ability to handle very large data sets, and overall efficiency. It offers linear time and space complexity such that substantial volumes of data can be easily processed (Tan et al., 2006). On the other hand, this approach can produce empty clusters and may have difficulty in handling outliers. Furthermore, because it uses the mean as the measure of central tendency, it assumes that the data follows a Gaussian (or normal) distribution. Nevertheless, K-means remains an extremely popular technique, and has demonstrated success in various applications and settings (e.g., Tan et al., 2006; Abdul-Nazeer and Sebastian, 2009; Garla et al., 2012; Mahendiran et al., 2012).

Other approaches have also been shown to perform well in data analysis routines, such as K-medoid and hierarchical methods. For example, K-medoid (or partitioning around medoids) is an algorithm that functions comparably to K-means, except it incorporates the median as the measure of central tendency. Consequently, it is more stable than K-means, and is applicable to a wide variety of data distributions.

Hierarchical methods include agglomerative and divisive approaches, although the former is a significantly more common approach (Tan et al., 2006). In agglomerative clustering, the algorithm begins with each object as a distinct cluster, and iteratively merges the closest pair of clusters until one global cluster containing all points is established. This process creates what is referred to as a tree, and this tree is ultimately "cut" at the appropriate number of clusters based upon some criterion or set of criteria. Complete Link (MAX or CLIQUE), and Ward's Method are two distinct and popular agglomerative techniques. By contrast, in divisive clustering, the opposite approach is taken. Specifically, the algorithm begins with a single group containing all objects, and iteratively splits until each point is a separate cluster . Once again, the tree is then sliced at the desired number of clusters in accordance with evaluation results. Strengths of hierarchical methods include the ability to handle differing cluster sizes as well as local decision-making at each iteration concerning how clusters are merged or split. Weaknesses include exponential time and space complexity and the fact that merging or splitting decisions are final (Tan et al., 2006).

Beyond K-means, K-medoid, and the hierarchical methods, other algorithms are available such as expectation maximization (EM) and DB Scan. However, they also present a series of challenges, including but not limited to significant time complexities. For example, while EM is actually viewed as a generalized version of K-means, it tends to be extremely inefficient on large data sets. In addition, the user must specify the number of clusters to be obtained as well as other settings. Conversely, EM can effectively operate with a wide array of distributions as well as clusters of varying sizes and shapes (Tan et al., 2006).

Generally speaking, it will be useful to consider an array of complementary algorithms when making a decision about which method to use for a given data set. As previously mentioned, no algorithm is strictly superior, each method carries a set of strengths and weaknesses, and the most appropriate algorithm is often a function of the data to be evaluated. In this study, K-means, Expectation Maximization, K-medoid, Complete Link, and Ward's Method will all be explored in determining which approach is most suitable for the involved data.

2.2 Data

The raw data is provided by a banking institution, and contains many attributes pertaining to the organization’s credit card customers. Also, the data set has 149,959 unique records. A primary initial challenge involves data preprocessing, including dimensionality reduction, problematic record elimination or transformation, and various issues concerning discretization, feature selection/creation, and normalization.

2.2.1 Dimensionality Reduction

The curse of dimensionality stipulates that, as the number of attributes increases beyond a certain point, the ability of data mining algorithms to produce meaningful results diminishes (Alpaydin, 2010). Consequently, it is advisable to include only truly useful features, while also minimizing redundancy. In establishing relevant dimensions, individual creditworthiness indicators reported in prior research and other sources serve to offer important guidance (e.g., Khandani et al., 2010; CFPB, 2012; CSD, 2013; Shuai et al., 2013). An initial listing of ten dimensions is assembled based upon this exploration, but further examination via descriptive statistics ultimately necessitates elimination of certain variables. For example, value of profitability represents the income a given customer presumably contributes to the institution, and would be useful for profiling purposes. However, this dimension exclusively contains null entries within the current data set. Also, VIP_Code is an internal metric used by the bank in assessing favorability of credit card clients, but 99.8 percent of records possess a value of “0” for this attribute. Therefore, it cannot provide for adequate differentiation among customers. Upon conclusion of the dimensionality reduction task, four attributes are established for profiling purposes. These include AccountAge, CreditLimit, AdditionalAssets, and LatePayments, and are described in the discretization, feature selection/creation, and normalization subsection below.

2.2.2 Record Elimination

Because the objective is to comprehensively segment the customer base and perform outlier analysis in accordance with existing data, a conservative approach to record removal is taken. Objects are only deleted if they contain obvious errors or are not able to be appropriately preprocessed for other reasons. For example, LatePayments is created via division of actual late payments by account age in months. The resulting attribute thus serves to standardize the measure for all customers, regardless of account age. However, some records reflect new accounts and thus possess an account age of zero. These line items are discarded to avoid the division by zero problem in deriving LatePayments. Following record elimination, the final data set includes 149,893 records and four dimensions.

2.2.3 Discretization, Feature Creation/Selection and Normalization

Each attribute is examined to determine the appropriate method of preprocessing. In doing so, characteristics such as data type are carefully considered. In the end, three dimensions are normalized and one variable is created and subsequently normalized. The specific dimensions and associated preprocessing routines are discussed next.

The age of credit card accounts is an important metric within the United States (U.S.) credit scoring system. Specifically, length of credit history carries a weighting of 15% in the computation of credit scores (CSD, 2013), and this suggests that account age is sufficiently important to consider in profiling credit card customers. The actual AccountAge dimension is expressed in months and ranges from one to 439. In terms of preprocessing, normalization precludes numeric dimensions with larger inherent values from dominating attributes with innately smaller amounts (Han et al., 2001). For example, if height in inches and weight in pounds are collectively used and provided equal weighting in a clustering operation involving people, results would be driven by the weight dimension. This is because it occupies a wider range, and, for each record, would typically be of a significantly larger value relative to height. Fortunately, proper transformation can resolve this problem. In specific terms, Shalabi et al. (2006) offer a basis for normalizing values on a [0,1] scale as follows:

$$\boldsymbol{Normalized\ Value} = \frac{\boldsymbol{Actual\ Value - Minimum\ Value}}{\boldsymbol{Maximum\ Value - Minimum\ Value}}$$

In the above equation, actual value is a specific amount to be transformed, minimum value is the smallest amount in the range of the target dimension, and maximum value is the largest amount for the feature of interest. For example, in normalizing the AccountAge variable, minimum value is one and maximum value is 439. When the formula is applied to each cell of the AccountAge column, all associated amounts fall within the desired [0,1] scaling.

Customer creditworthiness and credit lines maintain a positive relationship (Khandani et al., 2010). As a client's creditworthiness improves, he/she tends to be eligible for and/or granted enhanced lines of credit. Therefore, credit limit is an important indicator of credit risk and customer favorability, and should be a pertinent attribute. The actual CreditLimit dimension is measured in Brazilian Reais or Reals (BRL), and resides on a continuum from 45 to 80,000. Given its numeric nature, it is normalized in the same manner as AccountAge, except that the minimum and maximum values are 45 and 80,000, respectively.

Having credit variety is viewed as important to maintaining a solid credit standing. Specifically, 10% of U.S. credit score calculations incorporate the types of credit being used (Morgan, 2011). In the provided data, AdditionalAssets pertains to the number of bank products a customer has in addition to a credit card. For example, if a client held a credit card, savings account, mortgage, and auto loan, then the number of additional assets applicable to this individual would be three. This is a reasonable proxy for types of credit, and thus AdditionalAssets should provide incremental value for profiling purposes. As with the two previously discussed numeric variables, AdditionalAssets is transformed using the formula from Shalabi et al. (2006). In doing so, the minimum value is zero and the maximum value is eight, such that normalized amounts ultimately lie in the closed interval between zero and one.

Payment history is the most significant single determinant of U.S. credit scores. In particular, it carries a weighting of 35% in this calculation, and late payments is the most prominent element used in computing payment history (CSB, 2013). In highlighting its importance, a single late payment can reduce an individual's credit score by as many as 100 points (MSUFCU, 2015). Without question, this is an instrumental component of creditworthiness, and is therefore included. In achieving this, LatePayments is created from other elements in the raw data so as to provide for a standardized representation. Although the number of late payments is listed for each record, it is not inherently useful in this state. Other things equal, one would expect to note a positive relationship between account age and number of late payments. Therefore, simply normalizing the number of late payments and subsequently evaluating the data would produce misleading outcomes, and would tend to “punish” customers with older credit card accounts. To effectively incorporate the late payments metric, it is first modified so that values are made comparable among all records. In doing so, the LatePayments dimension is created through dividing number of late payments by account age in months, thus yielding a late payments per month measure for each record. Next, the data is examined to determine whether further transformation is necessary. In the U.S., it is typically the case that clients receive monthly credit card billing statements, and hence would be expected to make, at most, one payment per month. If this scenario is generalizable to the current context, the feature creation process should have fully transformed this variable. However, in examining the associated descriptive statistics, it is found to exist on a spectrum from zero to four. Consequently, normalization is done in a manner similar to the previous three dimensions. In completing this process, the minimum value is zero and the maximum value is four. With the data set now fully normalized, three important additional considerations are made prior to beginning actual data analysis. These are specifically addressed in the following three sections.

2.3 Dimension Redundancy

First, chosen dimensions must be compared to determine whether there are any redundancies to address. More specifically, when two or more features are highly related, all but one may be eliminated. Optimally, the retained variable will be that which contributes most significantly to model building. In testing for redundancy, a reference threshold of .75 is established[1], and pair wise correlations of all four dimensions are computed and compared with the benchmark value. In this way, any two dimensions having correlations of .75 or higher in absolute value terms are considered redundant. If this materializes, involved variables are further examined to determine which should be retained.

To test for redundancy, pair-wise correlations of all dimensions are computed. The results appear in Table 3 below, and do not seem to suggest problems. Specifically, absolute values of all correlations are well below the threshold amount. Consequently, the data set remains intact at this point.

[1] The .75 threshold is a rough heuristic. It is not based on prior literature, and, admittedly, this represents a potential limitation.

Table 1: Redundancy testing - Pair wise correlations of dimensions

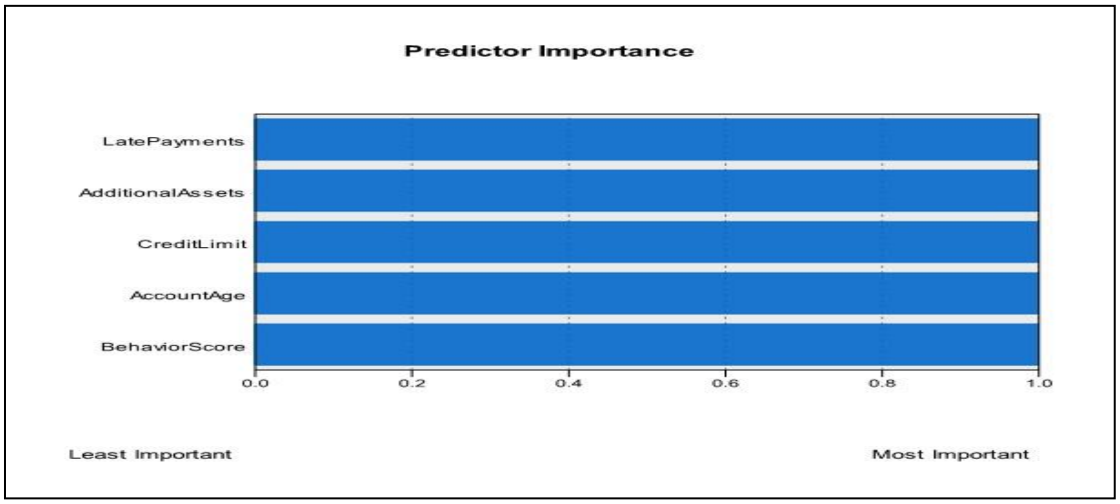
Pairwise correlation matrix				
Dimensions	AccountAge	CreditLimit	AdditionalAssets	LatePayments
AccountAge	1.0000000	0.4362117	0.0848748	-0.5727938
CreditLimit	0.4362117	1.0000000	0.1641602	-0.2917644
AdditionalAssets	0.0848748	0.1641602	1.0000000	-0.0241305
LatePayments	-0.5727938	-0.2917644	-0.0241305	1.0000000

However, correlations not indicative of superfluousness nevertheless must still be addressed relative to any endogeneity concerns. This can theoretically be alleviated in multiple ways, but one mechanism for effectively achieving this entails principal components analysis (PCA) (Tan et al., 2006). In particular, PCA generates a new set of variables that are linear combinations of the original dimensions. Also, these newly formed attributes maintain zero correlations with one another. Beyond this, PCA also serves as a dimensionality reduction technique, because it will typically be the case that the number of principal components needed for clustering will be less than the quantity of original dimensions. Moving forward, PCA is performed on all dimensions to be clustered, and the resulting principal components are used for evaluation.

2.4 Dimension Irrelevance

Second, it is important to identify and eliminate all irrelevant dimensions, and some data mining applications have the capability to determine which attributes actually contribute to solution outcomes (Tan et al., 2006). For example, IBM SPSS Modeler is able to assess the importance level for each dimension used in model construction. Although important features in this study are manually selected through research efforts in the initial data preprocessing step, it is still worthwhile to confirm the extent to which each chosen dimension supports model building. If a feature is identified as insignificant, it is irrelevant and discarded. In dealing with this issue, variables are examined in SPSS to conclude whether any lack utility. Intuitively, irrelevance is not expected to arise, given that an objective, literature-oriented approach is employed in feature selection and creation. Outcomes are presented in Table 2, and confirm that all dimensions contribute meaningfully to the model creation process. Therefore, all are retained for analysis routines.

Table 2: Irrelevance testing - SPSS analysis of dimension contributions



2.5 Model Evaluation Considerations

Third, a strategy is needed for model evaluation. Tan et al. (2006) stipulate that such assessment activities can be accomplished using supervised, relative, and unsupervised approaches. With supervised evaluation, cluster results are compared to an external standard, such as class label information. Given that no such guidance exists for the data in this study, a supervised method is infeasible.

2.5.1 Relative Evaluation

With the relative approach, cluster analysis is achieved by comparing and contrasting results based upon a subset of criteria. For example, in the K-means context, this form of evaluation might include incorporation of the sum of squared errors (SSE), but complementary mechanisms such as “knee” or “elbow” analysis would also be simultaneously performed. More specifically, an array of model results is plotted as a line whereby number of clusters and SSE are placed on the x and y axes, respectively. The graph is then reviewed to note where diminishing marginal returns occur concerning error reduction. This is identified as the region(s) where the slope of a line segment between two adjacent number of cluster values becomes less negative, thus producing an "elbow" or "knee" effect. Because SSE will naturally tend to decline as the number of clusters increases, elbow analysis serves as an important constraining mechanism. However, because it is not purely objective, a relative evaluation scheme such as this should be used in conjunction with other techniques. Whatever the case, because emphasis is on substantially automating the clustering and outlier detection processes, there is lack of preference for relative methods that are more likely to require a person in-the-process. Instead, a fully objective criterion or set of criteria is sought.

2.5.2 Unsupervised Evaluation

In the unsupervised domain, cluster analysis is conducted without the use of externally supplied information. This essentially involves application of an objective function, and specific examples include maximizing silhouette coefficient, maximizing the Krzanowski-Lai index, minimizing the Calinski-Harabasz metric, or minimizing the G3 index. In reflecting upon this, other potential supplemental methods emerge. For instance, percentage change is a measure frequently incorporated in the economics literature. Examples include but are not limited to elasticity measures such as price elasticity of demand, income elasticity of demand, and cross elasticity of demand, and they all rely fundamentally on the notion of percentage change in expressing the elasticity for various phenomena (Ekelund et al., 2006). This percentage change metric might also be suitable as an unsupervised evaluation technique in clustering. In particular, the objective in this context would presumably entail maximizing absolute percentage reduction in error in moving from n to n+1 clusters. Following reflection, it is apparent that unsupervised evaluation measures will better facilitate the automation of model selection. However, several options are available, and it is not currently known which metric, if any, would be a suitable proxy for model accuracy.

2.5.2.1 Cross-Validation Experiment for Unsupervised Metric Selection

In particular, in order for a measure to be useful for selection purposes, it should operate so as to choose the model that maximizes test accuracy. However, this is problematic in the current context because clustering is generally an unsupervised learning activity whereby label information is not available *a priori* for confirmation purposes. Given this predicament, a 10-fold cross-validation experiment is conducted using three labeled (two-class) data sets adapted from the UCI repository in order to explore the extent to which particular cluster quality indices are satisfactory for model selection. In achieving this, the data is clustered using a subset of algorithms. In addition, four recognized cluster quality indices (i.e., Silhouette Coefficient, Krzanowski-Lai, Calinski-Harabasz, and G3) are all evaluated. In conducting the experiment, some considerations must be made.

First, an approach is needed for deciding on the number of clusters for which models should be produced. To avoid biasing results, this should not incorporate use of any indices to be evaluated (e.g., G3). Given this situation, percentage change in error reduction is used as a basis. More specifically, cutoffs are established at 10%, 15%, and 20%, and models are constructed in accordance with these three thresholds. For example, the process initially proceeds by looking at percentage decline in error in moving from 1 to 2 clusters.

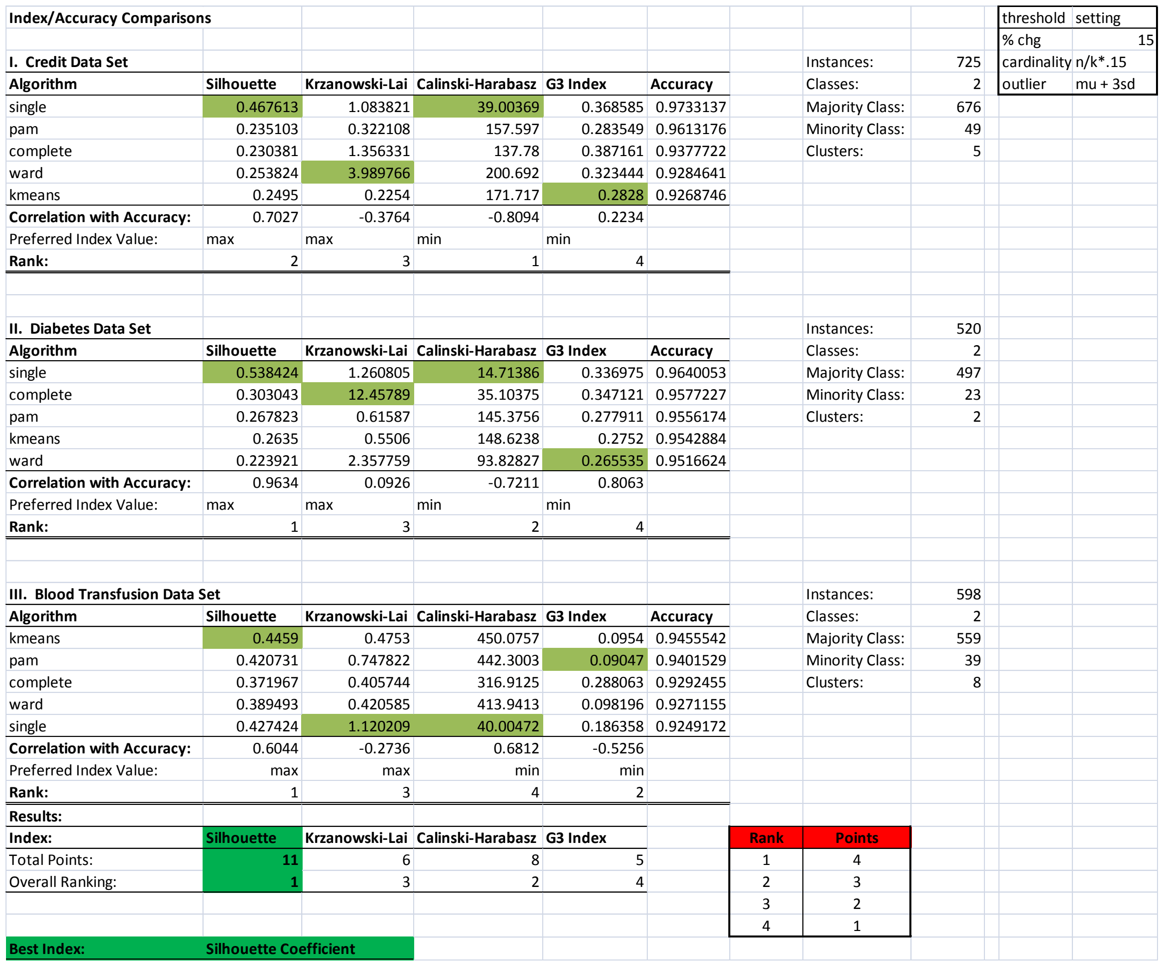
If this at least equals the cutoff value, then the procedure continues by examining percentage decline in moving from 2 to 3 clusters. The process terminates when percentage change initially falls below the cutoff, and the number of clusters chosen corresponds to the point where percentage change in error reduction last was above or equal to the cutoff value.

Second, because this paper is interested in fraud discovery (or more appropriately, irregularity detection), I am investigating a two-class problem. In particular, each object in this context is viewed as either normal (regular) or abnormal (irregular). However, this does not imply a two-cluster scenario automatically exists. In fact, any number of clusters might be applicable to this two-class problem. Consequently, a system is needed for deciding how objects in clusters should be assigned to classes in the experimental setting, and this involves two general considerations. First, when a cluster has an extremely small membership, all corresponding objects are viewed as irregular or abnormal. To determine what constitutes an extremely small membership, the notion of minimum cardinality must be established, and entails some percentage of n/k, where n is the number of records or objects in the data set and k denotes the number of clusters. In the cross-validation experiment, this percentage is set at four different levels including 5, 10, 15, and 20. For example, one specific cardinality threshold is n / k * .05. In this instance, if n is 10,000 and k is 10, then the cardinality threshold is 50. In this case, if membership for a given cluster is at or below 50, then all objects in that partition are assigned to the abnormal class. Second, each cluster exceeding the cardinality threshold is seen as likely containing both normal and abnormal objects. In fact, points furthest from (closest to) the associated cluster representative are more likely to be irregular (regular). In deciding where to assign an object, an outlier threshold must be constructed. In this experiment, this threshold is set as average euclidean distance plus three standard deviations. Thus, for each cluster satisfying the minimum cardinality condition, any object having a euclidean distance from the cluster representative that is greater than the outlier threshold is assigned to the abnormal class. Remaining objects are assigned to the normal class.

The 10-fold cross-validation experiment is formulated such that each data set is randomly split into 10 subsets. Then, for a given data set and array of cutoffs, each subset is used exactly nine times for training and one time for testing so that 10 iterations are performed in a single trial. In each iteration, the training set is used for model building and initial assignment. Then, the test set is classified based upon the training model results. Following this, test accuracy is computed. After the ten iterations, final test accuracy is calculated as the average of the ten test accuracies. This entire process is repeated for all data sets and combination of thresholds. This generates 36 sets of outcomes (3 data sets x 3 percentage change cutoffs x 4 cardinality thresholds).

Primary interest entails ultimate comparison of the four cluster quality indices with test accuracy information so that two questions may be answered: 1) Is there at least one index that serves as an adequate proxy for model accuracy?, and 2) If so, which index is relatively superior? In attempting to answer these questions, relevant values are documented and a ranking system is implemented. Specifically, for each iteration, index values for each algorithm are recorded. Then, test accuracies are arranged in descending order. Initially, extent to which the preferred value for each index corresponds with the highest test accuracy is noted. Then, secondary interest involves determining whether a meaningful relationship exists between a given set of values for an index and associated test accuracies. For instance, because a lower G3 index value is more favorable, if this index is indicative of test accuracy, there should be a significant negative correlation between the vectors of index values and test accuracies. Obviously, the closer this correlation is to -1, the better. In ranking each evaluation measure, the extent to which a metric successfully selects the correct method in terms of test accuracy is given primary weighting. In the event of a tie, correlations serve to resolve matters. For a given data set and associated array of parameter settings, the best performing index is given a score of 4, and the worst performing index is assigned a score of 1. Also, the intermediate rankings are appropriately scored as 2 and 3. Next, score sub-totals for all three data sets are summed. One set of outcomes is presented in Table 3 below.

Table 3: 10-fold cross-validation results example



Index/Accuracy Comparisons

threshold	setting
% chg	15
cardinality	n/k*.15
outlier	mu + 3sd

I. Credit Data Set

Algorithm	Silhouette	Krzanowski-Lai	Calinski-Harabasz	G3 Index	Accuracy
single	0.467613	1.083821	39.00369	0.368585	0.9733137
pam	0.235103	0.322108	157.597	0.283549	0.9613176
complete	0.230381	1.356331	137.78	0.387161	0.9377722
ward	0.253824	3.989766	200.692	0.323444	0.9284641
kmeans	0.2495	0.2254	171.717	0.2828	0.9268746
Correlation with Accuracy:	0.7027	-0.3764	-0.8094	0.2234	
Preferred Index Value:	max	max	min	min	
Rank:	2	3	1	4	

Instances:	725
Classes:	2
Majority Class:	676
Minority Class:	49
Clusters:	5

II. Diabetes Data Set

Algorithm	Silhouette	Krzanowski-Lai	Calinski-Harabasz	G3 Index	Accuracy
single	0.538424	1.260805	14.71386	0.336975	0.9640053
complete	0.303043	12.45789	35.10375	0.347121	0.9577227
pam	0.267823	0.61587	145.3756	0.277911	0.9556174
kmeans	0.2635	0.5506	148.6238	0.2752	0.9542884
ward	0.223921	2.357759	93.82827	0.265535	0.9516624
Correlation with Accuracy:	0.9634	0.0926	-0.7211	0.8063	
Preferred Index Value:	max	max	min	min	
Rank:	1	3	2	4	

Instances:	520
Classes:	2
Majority Class:	497
Minority Class:	23
Clusters:	2

III. Blood Transfusion Data Set

Algorithm	Silhouette	Krzanowski-Lai	Calinski-Harabasz	G3 Index	Accuracy
kmeans	0.4459	0.4753	450.0757	0.0954	0.9455542
pam	0.420731	0.747822	442.3003	0.09047	0.9401529
complete	0.371967	0.405744	316.9125	0.288063	0.9292455
ward	0.389493	0.420585	413.9413	0.098196	0.9271155
single	0.427424	1.120209	40.00472	0.186358	0.9249172
Correlation with Accuracy:	0.6044	-0.2736	0.6812	-0.5256	
Preferred Index Value:	max	max	min	min	
Rank:	1	3	4	2	

Instances:	598
Classes:	2
Majority Class:	559
Minority Class:	39
Clusters:	8

Results:

Index:	Silhouette	Krzanowski-Lai	Calinski-Harabasz	G3 Index
Total Points:	11	6	8	5
Overall Ranking:	1	3	2	4

Rank	Points
1	4
2	3
3	2
4	1

Best Index: Silhouette Coefficient

As can be seen, the silhouette coefficient performs most adequately in this particular set of trials. More important, it consistently selects the best model in terms of test accuracy results. The experimental procedure is repeated until all combinations of parameter settings are evaluated. Then, individual index scores are summed to arrive at an aggregate index score for each measure. Given that there are 36 individual rankings conducted, a particular index could achieve a maximum aggregate score of 144 (36 x 4), but this would require it to have the top ranking in all 36 instances. Actually, the silhouette coefficient performed in a superior fashion in an overall sense, achieving an aggregate score of 133 (see Table 4).

Table 4: 10-fold CV final rankings

Aggregate Points	
Silhouette	**133**
Calinski-Harabasz	86
Krzanowski-Lai	74
G3	67

Furthermore, the silhouette corresponded with the best performing model in terms of test accuracy in 30 of the 36 experiments. Because of its convincing relative dominance as well as satisfactory absolute performance, the silhouette coefficient appears suitable for automated model selection. However, before proceeding, it is useful to first obtain a better understanding of this index.

2.5.2.2 Silhouette Coefficient - An Indicator of Cluster Quality

The silhouette coefficient is a measure of cluster cohesion and separation (Tan et al., 2006). Consequently, the metric can meaningfully contribute to determining the number of clusters present in a given data set. In addition, the above experiment has offered evidence of its appropriateness for algorithm selection. The coefficient can theoretically vary between -1 and 1, with a higher value indicative of better cohesion and separation. The general formula for the silhouette coefficient of the *i*th object is:

$$\boldsymbol{Silhouette\ Coefficient\ (s_i) = \frac{(b_i - a_i)}{max\ (a_i, b_i)}}$$

Where: *a_i* = average distance from *i*th object to all other objects in same cluster, and
b_i = average distance from *i*th object to all other objects in next closest cluster

Typically, it is anticipated that $b_i > a_i$, and, in this setting, the silhouette coefficient can be simplified and expressed as:

$$\boldsymbol{Silhouette\ Coefficient\ (s_i) = 1 - {}^{a_i}/_{b_i}}$$

When a_i is zero, the silhouette coefficient attains its maximum value of 1. While this will certainly not be approximated in practical applications, a larger silhouette coefficient nevertheless argues for relative model superiority in terms of both cohesion and separation. In particular, a_i is the indicator of cohesion while b_i is the measure of separation, such that lower (higher) values of a_i (b_i) are strictly preferred. Given that cohesion and separation are described as two key indicators of cluster quality (Tan et al., 2006), it is not surprising the silhouette index is found to be suitable for model selection.

2.6 Analysis[2]

2.6.1 Model Selection

In assessing models, automated cluster simulation routines are executed for the five previously discussed algorithms. In doing so, silhouette coefficient is the basis for model judgment in this phase such that the algorithm producing the largest value is ranked as being most preferred. Initial findings from this procedure are presented in Table 3, and suggest that the Complete-Link Hierarchical method be adopted for clustering.

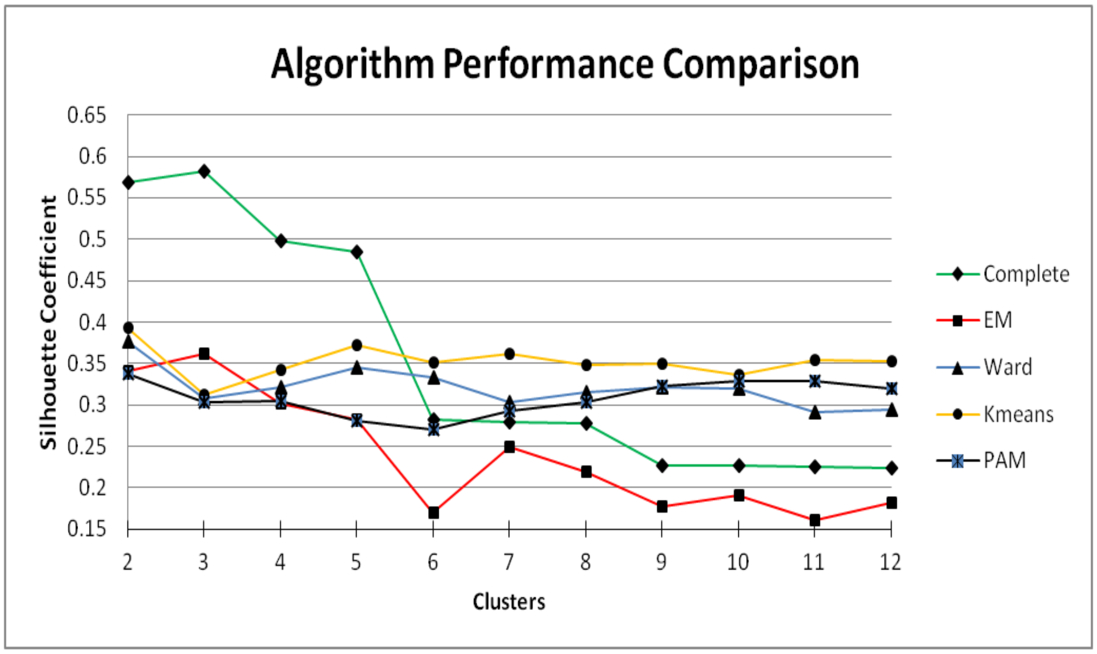
Table 5: Algorithm ranking

Algorithm Ranking	
Method	**SilhouetteCoefficient**
Complete	0.5817
K-Means	0.3936
Ward	0.3768
EM	0.3613
PAM	0.3383

To provide additional insight, the entire result set is plotted for each combination of algorithm and number of clusters. This is shown in Figure 1.

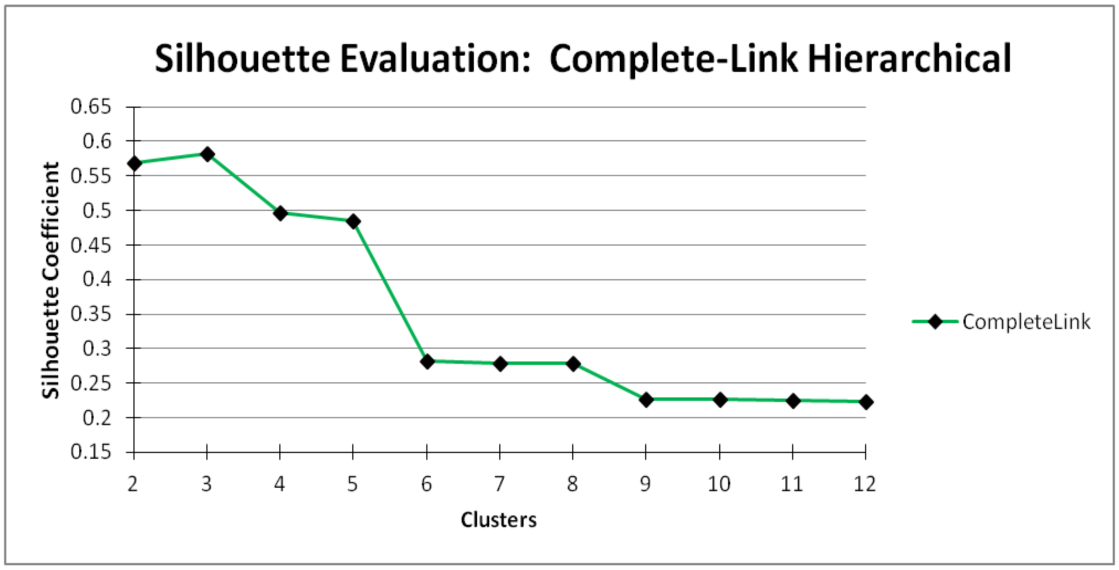
[2] R will not operate with long vectors (i.e., length > 2^31). Therefore, a random sample of 40,000 objects is used to compute silhouette values during the simulation routines. While this is perceived as a limitation, preliminary work with a variety of samples from the data yielded identical results in terms of ultimate model selection.

Figure 1: Algorithm performance plots - comprehensive



The complete-link hierarchical method is strictly superior to the other four algorithms in terms of silhouette coefficient for the range between two and five clusters, inclusive. In addition, it seems to achieve peak performance at three. For added clarity, an exclusive plot of silhouette coefficients for the complete-link method is considered next.

Figure 2: Algorithm performance plot - Complete-Link

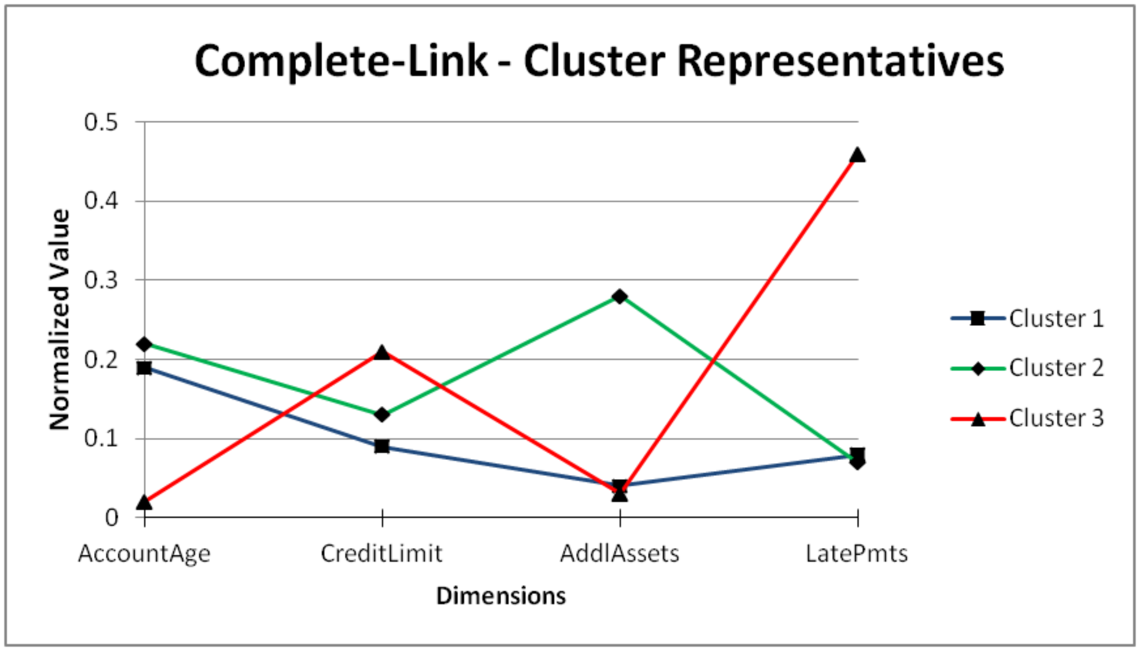


In fact, at three clusters, the maximum silhouette coefficient is attained (i.e., .5817). Even so, a two cluster solution is not substantially different, yielding a slightly lower silhouette of .5688. Nevertheless, because the present objective entails choosing the model with the highest silhouette index, a complete-link hierarchical three-cluster solution is selected for subsequent analyses.

2.6.2 Model Creation, Visualization, and Evaluation

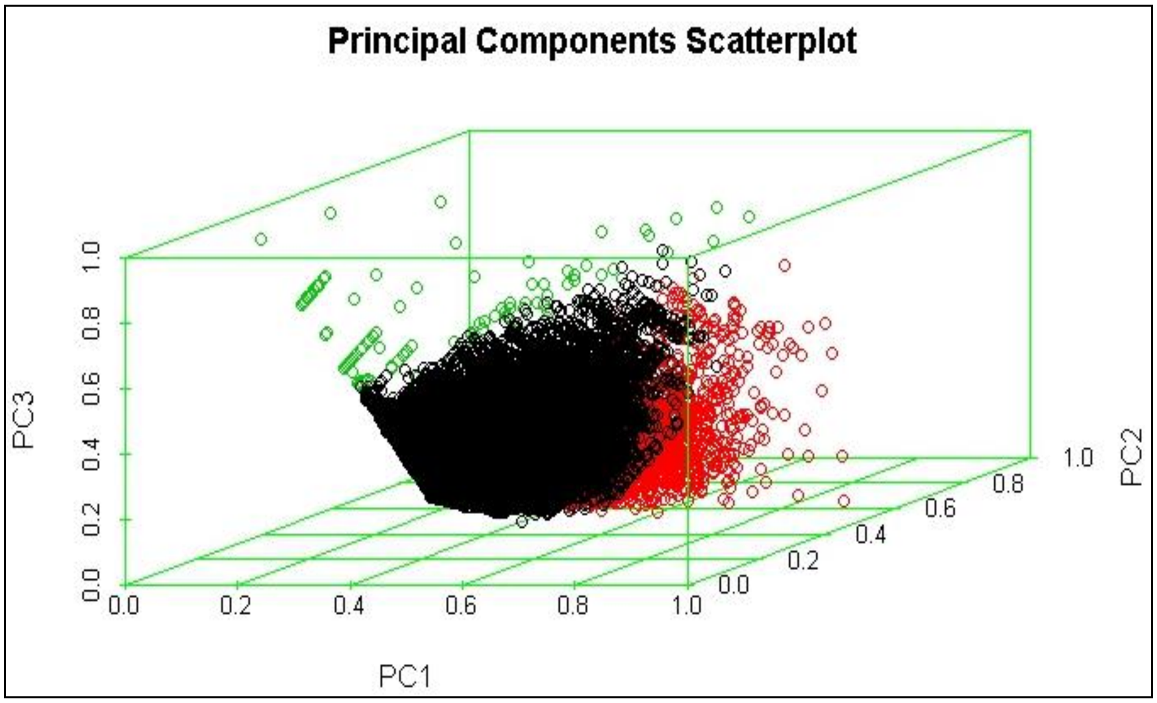
This model is generated and a series of visualizations follow in an effort to note distinguishing features of each cluster. In Figure 3 below, representative values for each profile are shown.

Figure 3: Cluster representative values for complete-link model



This graph depicts a meaningful set of groupings in that representative dimension values for a given profile are generally distinguishable from the others. In particular, cluster 2 is superior in terms of three dimensions, and thus represents the most creditworthy partition. In addition, cluster 3 is inferior with respect to account age, late payments, and additional assets. Interestingly, while this partition corresponds to the least creditworthy clients, its average credit limit is higher than the other segments. Finally, cluster 1 represents the intermediate category, with representative values typically falling between the extremes. To provide additional context, a three dimensional principal components scatter plot is generated whereby each object is depicted and a color coding scheme is employed to distinguish objects by cluster. This is presented in Figure 4.

Figure 4: 3D plot of normalized principal components



The above image appears at a 35 degree rotation and shows all objects in three dimensional space based upon normalized principal component values. Furthermore, each record is color coded to indicate cluster membership. Clearly, the vast majority of points are shown as black and correspond to the majority cluster. In fact, this group is classified as cluster one by the complete-link model and contains 141,858 of the 149,893 observations (94.6%). By contrast, the minority group contains only 147 records (.10%), and these are depicted as green in the above visualization. Incidentally, these are treated as cluster three in the complete-link model, and, given the extremely rare membership of this cluster, all associated records should be considered as potential outliers and investigated individually. The remaining 7,888 records appear as red points in Figure 4 above, and these correspond to cluster two.

Finally, statistical testing is performed on a representative sample of the data using a non-parametric Kruskal-Wallis test. However, given that there is small membership in one of the three clusters, results might be taken with a degree of caution. Traditionally speaking, the minimally adequate group size for statistical testing purposes is 30 (Singleton and Straits, 1999). However, the authors have also suggested minimum recommended group size might often be much larger than this (e.g., 100 to 200). Nevertheless, results are shown in Table 5, and offer additional incremental evidence arguing for a three-cluster solution.

Table 6: Kruskal-Wallis test - Complete-Link 3 cluster model

Field	Chi-Squared	df	Significance Level
AccountAge	156.89	2	**p<.001**
CreditLimit	212.83	2	**p<.001**
AdditionalAssets	6,833.20	2	**p<.001**
LatePayments	92.70	2	**p<.001**

*Based upon a representative sample of 30,000 records.[3]

More specifically, all variables are shown to be highly significant, with each dimension having a p-value below .001. This offers supplementary evidence indicating each cluster is distinct from the others and thus describes a unique

[3] Because the smallest cluster contains only 147 objects, a representative sample of 30,000 ensures that about 30 records from this group should be captured in the sample (i.e., 30,000/149,893 x 147 = 29.42). Incidentally, tests on the full population were also performed and all results found to be highly significant as well.

customer type. At this point, the 3-cluster complete-link hierarchical solution is accepted as the final stage-one model.

Note that the analysis process up to this point is substantially automated in the R programming environment. Consequently, significant processing and evaluation efficiencies have been achieved, and many of the complexities of clustering are at least mitigated. It is hoped that this will ultimately assist in encouraging auditors to move more actively toward adopting data mining technologies in their quest to continue providing information useful for stakeholder decision-making. For convenience, the R code pertaining to this chapter is reproduced in Appendix A.

To better facilitate understanding of existing customer group differences at this juncture, representative values are de-normalized and corresponding results are displayed in Table 7.

Table 7: De-normalized representative values for 3-cluster model

	Cluster		
Dimension	1	2	3
AccountAge	83	98	9
CreditLimit	$7,462	$10,451	$16,741
AdditionalAssets	0.27	2.27	0.27
LatePayments	6.95%	6.66%	42.36%
Instances:	141,858	7,888	147
Percent of Total:	94.64%	5.26%	0.10%

As can be seen, each profile contains a different customer type. For example, cluster 2 represents the most mature and creditworthy clients. In particular, mean account age is over 8 years, and, of the total number of payments made by customers in this group, only 6.7% are late, on average. However, this partition comprises just over five percent of the data set. On the other hand, cluster 3 describes the least mature and creditworthy individuals. In particular, the mean account age is only nine months, and, on average, nearly half of all payments submitted are late. Also, given the noted problems and issues, average credit limit appears exorbitant, especially when compared with the remaining segments. Fortunately, this group contains only 147 individuals (.10%). Without question, these customer accounts all deserve closer investigation and scrutiny. Cluster 1 basically falls between the extremes, and thus corresponds to the intermediate portion of the customer base in terms of creditworthiness. Also, this profile comprises the vast majority of credit card accounts. In summary, each customer group possesses a unique set of characteristics, and this suggests that each partition should be approached and managed differently.

Clustering results such as those in Table 7 can offer immediate utility to auditors in events such as risk assessments and going concern engagements. For example, imagine that a comprehensive set of attributes and standardized procedures are established for describing the customer base. The attributes might be industry specific, and industry benchmarks could be established for comparison purposes. The customer base could be clustered periodically (e.g., weekly, monthly, quarterly, annually, etc.) so the auditor was able to perform trend analysis relative to this information. To facilitate efficiency, the pertinent customer and industry benchmark information could be reflected as numeric scores. In a manner analogous to ratio analysis, the auditor could plot the customer base and benchmark scoring information over a desired time horizon, thus being positioned to monitor and respond to changes in the organization's client structure. For instance, if the customer base score exhibited significant decline and/or fell below the associated benchmark value, this would indicate substantially increasing risk, and the auditor would be positioned to incorporate this into the risk assessment, thus modifying the associated audit plan and corresponding audit scope and tests. In the going concern context, customer base information would be useful for decision-making regarding the probability an organization will remain solvent. Whatever the case, customers are critical to organizational success, and auditing would certainly benefit from using and disseminating information generated from pertinent customer data. More important, clustering could be useful in both detecting and mitigating the incidence of fraud as well as assisting auditors in satisfying audit requirements relative to the consideration of fraud in financial statement audits.

CHAPTER 3: INVESTIGATING AND CLUSTERING THE CLUSTERS

3.1 Background

Another potentially important consideration relative to clustering strategy involves the actual partitions in a given model. Tan et al. (2006) point out that a model could actually contain nested clusters. This implies that segments be further examined prior to finalization, and a stream of research has pursued this issue. In one study, a two-phase approach is implemented (Garla et al., 2012). In the first stage, K-means is used to obtain initial groupings. In the second phase, primary profiles are evaluated using a probabilistic method. In conclusion, it is found that a two-stage process provides for improved overall results. In another paper, a self-organizing map (SOM) network is employed to initially establish region clusters (Mo et al., 2010). These are investigated and multiple segments are ultimately discovered within each area. In comparing this two-step method to three alternative approaches, the two-stage procedure is significantly better than two of the three alternatives, and not significantly different from the remaining method. However, the two-stage approach is found to be most efficient, and therefore viewed as the preferred option. The above findings suggest that evaluation of initial clusters is warranted, and this leads to a research question:

RQ1: Can an improved set of customer profiles be achieved through a two-stage clustering, or is a traditional single-stage method sufficient?

The above question will be explored in this chapter. Upon conclusion, a finalized solution will be proposed and further examined.

3.2 Preliminary Descriptive Statistics

The data set is now segregated into three subsets, one pertaining to each primary cluster identified at the conclusion of chapter one. Preliminarily, because credit card fee discount information is available, associated descriptive statistics are computed for each subset to obtain a general sense of whether clusters differ on this dimension and the bank is currently managing its credit card customers in a systematic fashion and in accordance with the partitions identified thus far. Results appear in Table 8, and reveal several points of interest.

Table 8: Descriptive statistics - Fee discount percentages by cluster

Cluster	Mean	Median	Mode	Min	Max
Cluster 1	64%	60%	60%	0%	100%
Cluster 2	71%	70%	60%	4%	100%
Cluster 3	72%	76%	84%	38%	100%

Note: only non-negative amounts are included in this analysis.

Initially, discount differences between clusters are apparent, but these disparities are not consistently logical. For example, while the discount range of the least creditworthy cluster (i.e., 3) is relatively narrow, minimum discount is much higher than that for the other two profiles. Of potentially greater concern, the least favorable group enjoys the highest mean and median discounts at 72 and 76 percent, respectively. Furthermore, the most frequent discount in this partition is 84%, and this is 24 percentage points larger than that for the remaining groups. By contrast, the most creditworthy segment (i.e., cluster 2) has lower mean, median, and mode discount percentages than the least creditworthy partition. On the other hand, the intermediate group maintains discount statistics that, in general, are systematically below those of the most creditworthy cluster. Basically, the greatest points of concern in terms of the discount issue pertain to cluster three, and, fortunately, there are few customers in this category. Nevertheless, observations collectively suggest that, from the standpoint of fee discounts, the groups are not being treated in conformity with relevant indicators of creditworthiness. These findings as well as the potential for nested clusters suggest that a more comprehensive profiling of the customer base is warranted. Hopefully, among other things, final outcomes will facilitate the eventual development of a formal policy that representatives will subsequently implement in dispensing customer services.

3.3 Nested Cluster Evaluations

In continuing analyses, each data subset is considered separately to determine whether nested clusters exist within the primary groupings identified in the previous chapter. However, because cluster three has a very small membership, it will not be explored concerning the nested cluster issue. In reality, all objects in this group are viewed as anomalous, and should thus be thoroughly investigated on an individual basis. Conversely, cluster one contains the vast majority of objects while cluster 2 is not considered to be of small membership status. Consequently, they are subjected to deeper review. As in the previous chapter, the silhouette index is used as a basis for model selection for the cluster 1 and 2 data subsets. Once again, this process is substantially automated in R, and the associated R-script used for evaluation and model building is contained in Appendix B.

3.3.1 Cluster 1 Model Selection

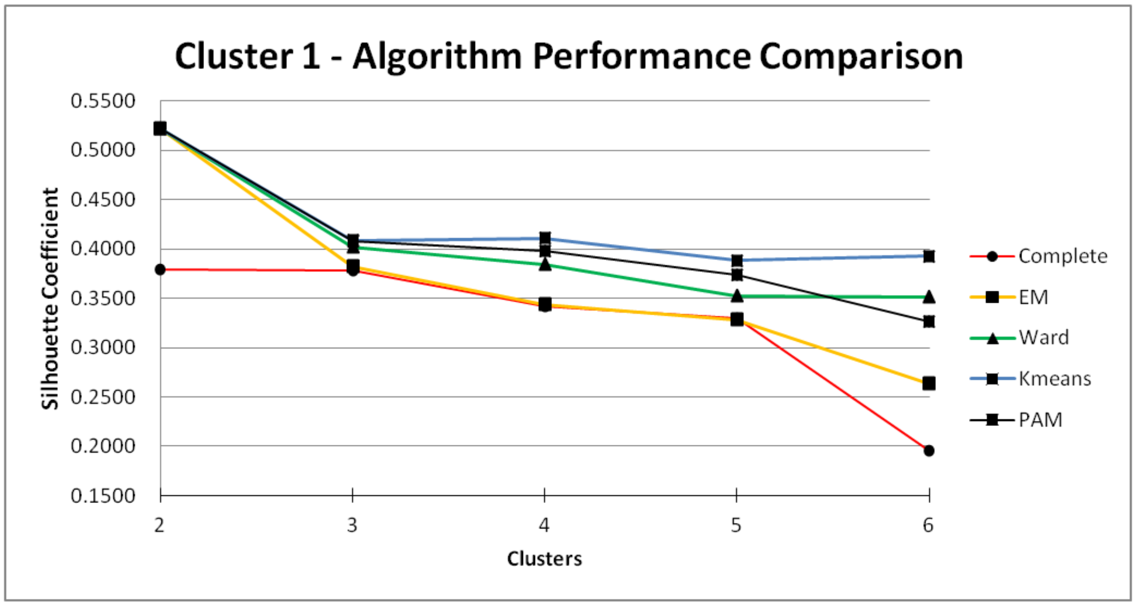
Initially, a simulation routine is again executed whereby silhouette coefficients are computed for a variety of clustering models. However, one difference exists relative to the previous chapter. Specifically, the number of potential sub-clusters considered is restricted to a range of two to six. Resulting algorithm rankings for cluster 1 are generated and presented in Table 9.

Table 9: Cluster 1 - Algorithm Rankings

Algorithm Ranking	
Method	**SilhouetteCoefficient**
K-Means	0.522
PAM	0.522
Ward	0.522
EM	0.522
Complete	0.379

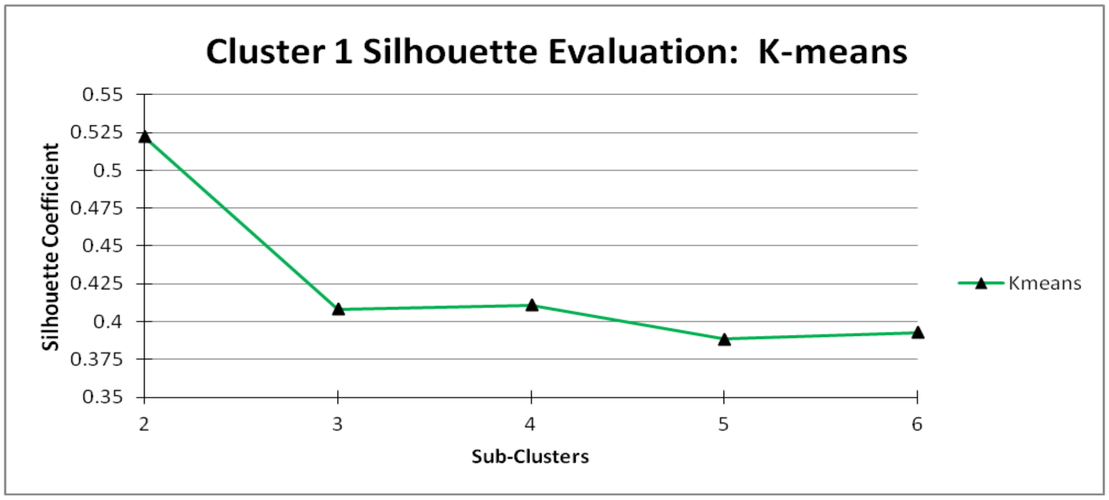
K-means, PAM, Ward, and EM are identical in terms of peak silhouette value (i.e. .522), initially suggesting that any of these algorithms could be chosen. To offer additional insight, silhouette values are plotted for all combinations of algoirthms and cluster settings. The associated graph appears in Figure 5 below.

Figure 5: Algorithm performance plots - comprehensive



In the above graph, the highest silhouette clearly occurs at two sub-clusters. Furthermore, as number of clusters increases, there is a general trend of decline in index values. This suggests that, at most, two nested partitions might exist in the cluster 1 data. However, because four methods have the same silhouette at two partitions, an additional observation is incorporated relative to algorithm selection. Specifically, it is noted that, at any other setting for number of clusters, K-means exclusively maintains the largest silhouette value. Based upon this, K-means is chosen as the algorithm for subsequent model construction. Specifically, a K-means two-cluster solution is used in determining whether any nested clusters are present in this data. Prior to execution, silhouette values for the chosen algorithm are first plotted, and follow in Figure 6.

Figure 6: Algorithm performance plot - K-means

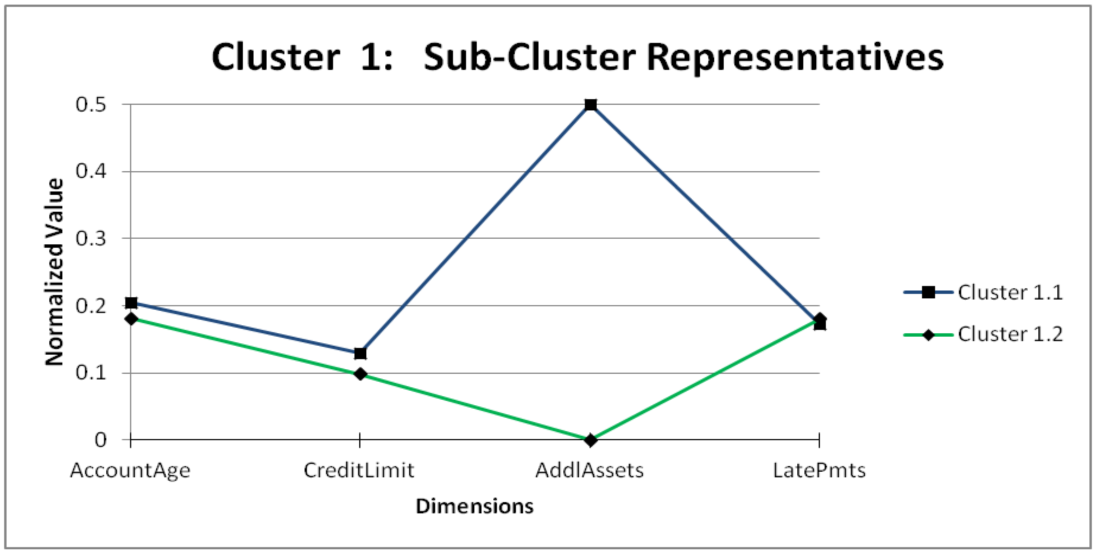


From this perspective, the two-cluster silhouette value is again noted as vastly superior to that of all competing models.

3.3.2 Cluster 1 Model Creation, Visualization, and Evaluation

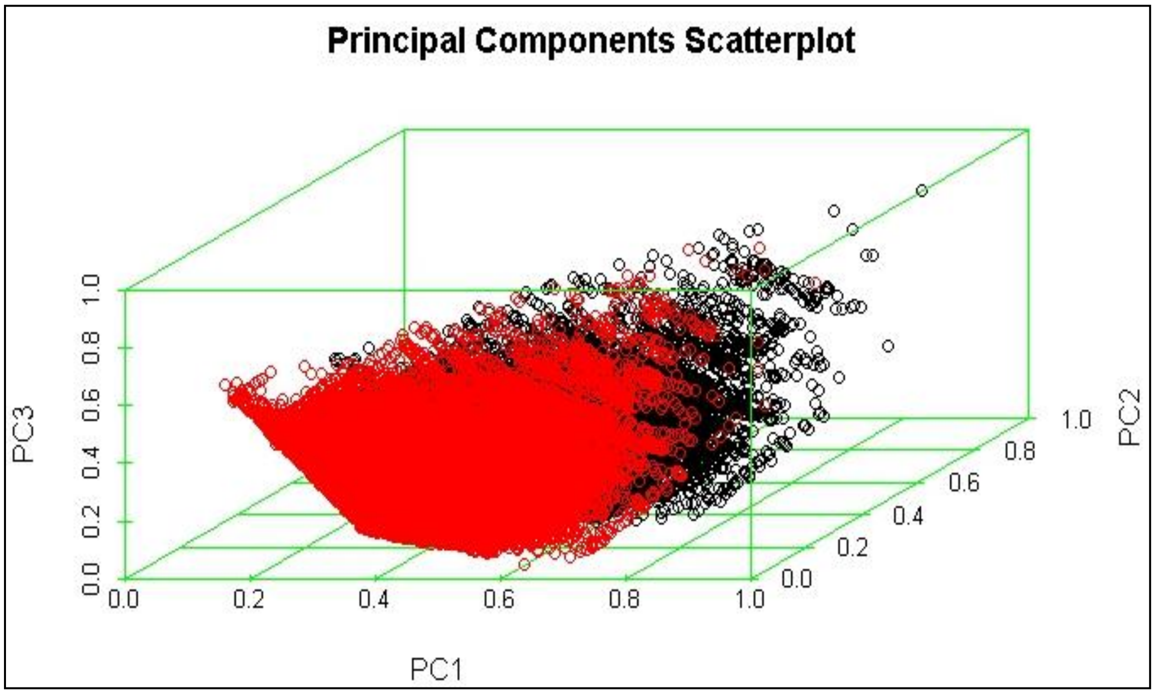
Moving forward, a K-means two sub-cluster model is developed and further explored. An image of cluster representatives follows in Figure 7.

Figure 7: Cluster representative values for K-means, 2 cluster model



Outcomes do not unambiguously argue that each of the two created sub-clusters represents a unique customer type. More specifically, the partitions are primarily distinguished via additional assets, and the remaining three dimensions are comparable in terms of normalized representative values. Even so, cluster 1.1 is technically superior in that it encapsulates more mature customers who maintain higher credit limits, possess more additional assets and make fewer late payments, on average, relative to cluster 1.2. Whether the two clusters are unique enough to justify segregation remains an open question, and, to provide additional context, a three dimensional principal components scatter plot is generated such that each object is depicted and a color coding scheme is used to compartmentalize objects by cluster. This is presented in the following image.

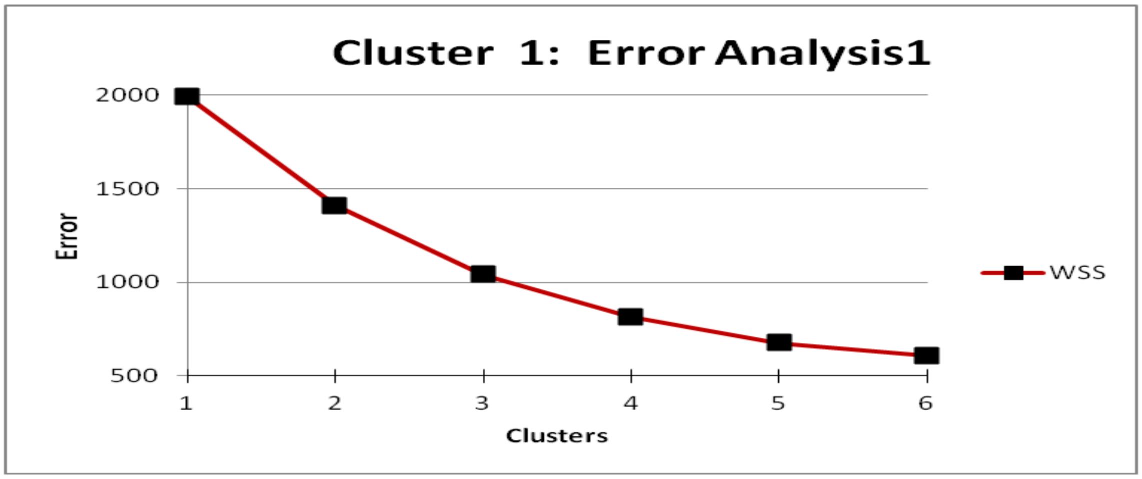
Figure 8: 3D plot of normalized principal components for cluster 1 data



In this view, objects are shown in three dimensional space in terms of normalized principal components. The model initially appears much more balanced relative to results from the prior chapter. Nevertheless, the majority cluster (subcluster 1.2) contains 103,015 of the 141,858 observations (72.6%), and this consists of objects appearing in red. Conversely, the remaining segment (subcluster 1.1) contains only 38,843 records (27.4%). In examining the plot, some level of separation is detected, but it fails to yield any convincing visual evidence that two distinct clusters are present.

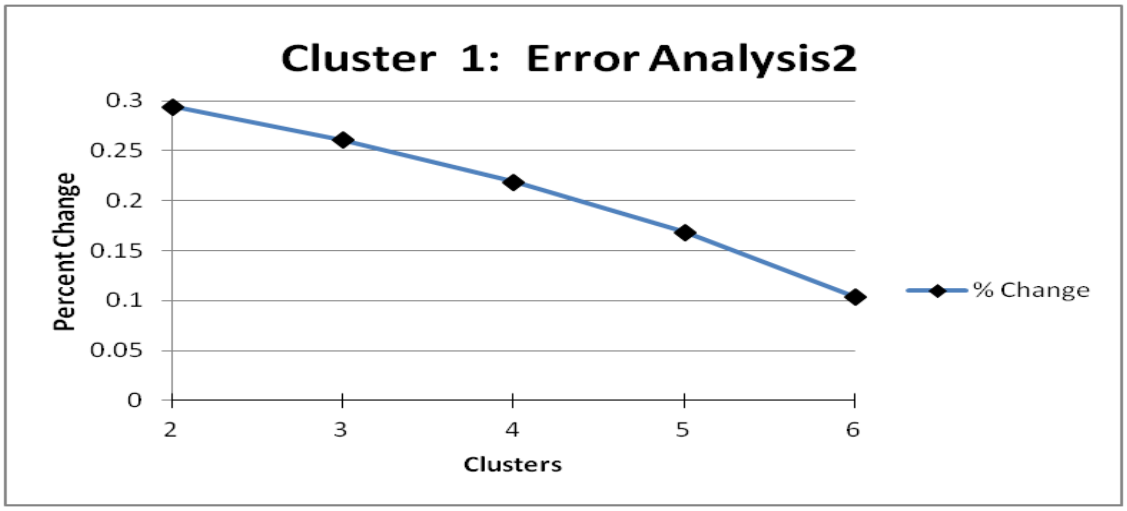
To gain additional insight, a two-part error analysis routine is performed. First, "elbow" analysis is conducted. In achieving this, error is computed for models ranging from one to six clusters, inclusive. Then, a graph is created with number of clusters plotted in ascending order on the x-axis and related error amounts on the y-axis, with adjacent error points connected via a series of line segments. Finally, the plot is examined to determine whether diminishing marginal returns occur in moving from n to n+1 clusters. As explained in the previous chapter, this occurs where the slope of a line segment between any two adjacent number of cluster values becomes less negative relative to the previous line segment. The graph follows in Figure 9.

Figure 9: Cluster 1 - Sub-cluster error analysis1



The individual segments collectively produce a line with a curvilinear structure, offering no clear argument that noteworthy diminishing marginal returns in error reduction is exhibited and indicating that nested clusters probably do not exist. Second, another graph is produced whereby number of clusters is shown in ascending order on the x-axis and percentage change in error reduction is plotted on the y-axis. This is produced in Figure 10, and essentially mirrors the previous image.

Figure 10: Cluster 1 - Sub-cluster error analysis 2



The highest percentage change occurs at two clusters, suggesting that, if sub-clusters exist, then the maximum number is two. However, the slope of the line is again highly curvilnear, and, in an overall sense, does not clearly campaign for the existence of nested clusters. Furthermore, when combined with the previous image, aggregate error-related evidence argues that the cluster 1 data should not be split.

A final piece of evidence is collected through two-group Kruskal-Wallis non-parametric tests of all dimensions. In this case, because a sufficient number of records reside in each sub-cluster, a representative sample is selected for testing purposes. Specifically, the total number of records in the population of cluster 1 is 141,858, with the majority (minority) partition containing 103,015 (38,843) records. Given that prior literature suggests minimum group size should be somewhere between about 30 and 200 instances (Singleton and Straits, 1999), the target sample size in this setting is 368 so as to ensure the minority sub-cluster is represented in a satisfactory manner[4]. The results appear in Table 10, and demonstrate that, with a .01 significance threshold, the groups are not significantly different in terms of three of the four involved attributes.

Table 10: Cluster 1 - Kruskal-Wallis results (n=368)

Field	Chi-Squared	df	Significance Level
AccountAge	1.06	1	**p = .3023**
CreditLimit	5.71	1	**p = .01684**
AdditionalAssets	369.00	1	**p < .001**
LatePayments	0.29	1	**p = .5921**

Furthermore, if this threshold is increased to .05, two of the dimension representatives continue to remain insignificant. In comparing this table to the graph of cluster representatives in Figure 7 above, commonalities become apparent. For instance, additional asset representative amounts are very different for the two sub-clusters, and the Kruskal-Wallis test is highly significant

[4] Because sub-cluster sizes are sufficient, emphasis is on obtaining about 100 cases from the minority cluster in the statistical sample (38,843/141,858 x 368 = 101).

for this attribute (p < .001). Conversely, late payment representatives do not appear unlike for the two groups, and, not surprisingly, the Kruskal-Wallis test produces highly insignificant results, indicating that the two clusters are not statistically different relative to this variable. Moving forward, because prior visual evidence and statistical testing generally fail to support the presence of nested partitions, cluster 1 is not split into sub-groups. Instead, it is retained as it existed at the conclusion of the previous chapter. Next, cluster 2 is evaluated in a similar manner to determine whether it contains multiple partitions.

3.3.3 Cluster 2 Model Selection

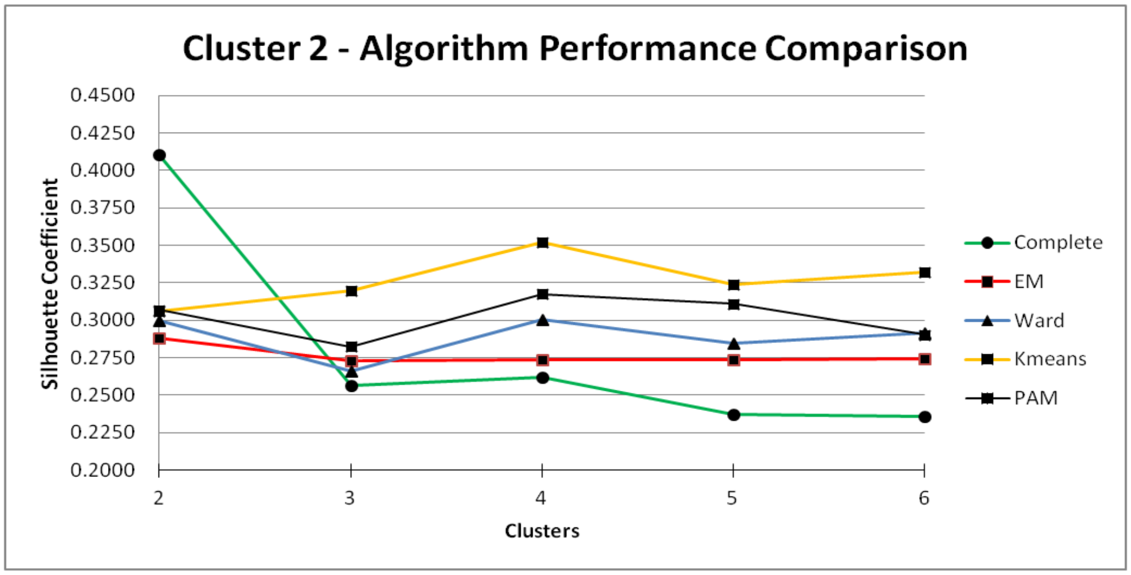
Initially, silhouette coefficients are generated for the five algorithms with the number of sub-clusters again ranging from two to six, inclusive. Outcomes are presented in Table 11.

Table 11: Cluster 2 - Algorithm Rankings

Algorithm Ranking	
Method	**SilhouetteCoefficient**
Complete	0.411
K-Means	0.352
PAM	0.318
Ward	0.311
EM	0.2879

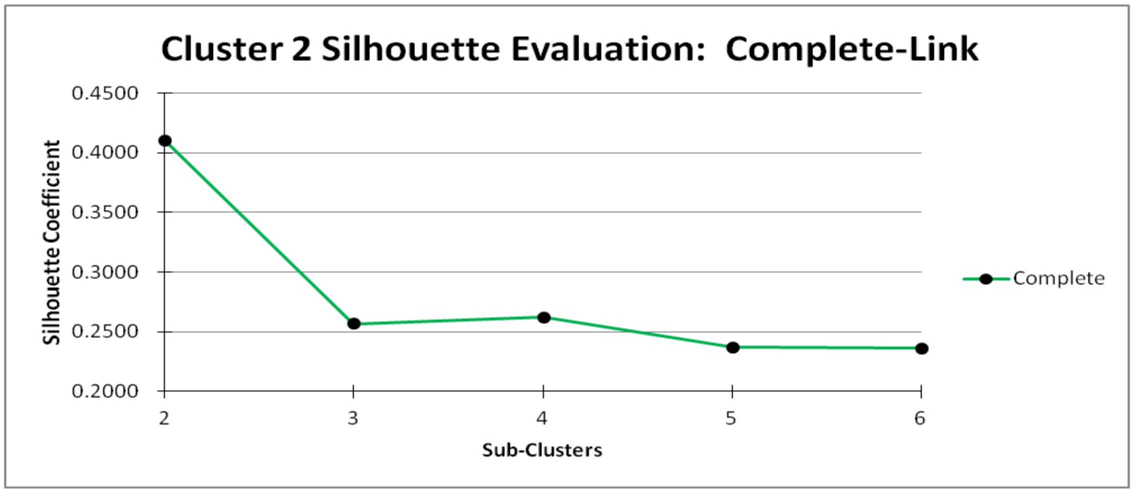
In this case, Complete-Link achieves the highest silhouette value (i.e. .411), but, to gain more specific insight, silhouettes are depicted for all combinations of algoirthms and cluster settings. The associated graph appears in Figure 11.

Figure 11: Algorithm performance plots - comprehensive



In the above, the silhouette is maximized at two sub-clusters, and all other models are substantially lower in terms of the comparison index. This suggests that, at most, two sub-groups might exist in the cluster 2 data, and that the Complete-Link method should be used in making decisions about the nested cluster issue. To obtain improved intra-algorithm contrast, silhouette values for the chosen method are plotted, and follow in Figure 12 below.

Figure 12: Algorithm performance plot - Complete-Link

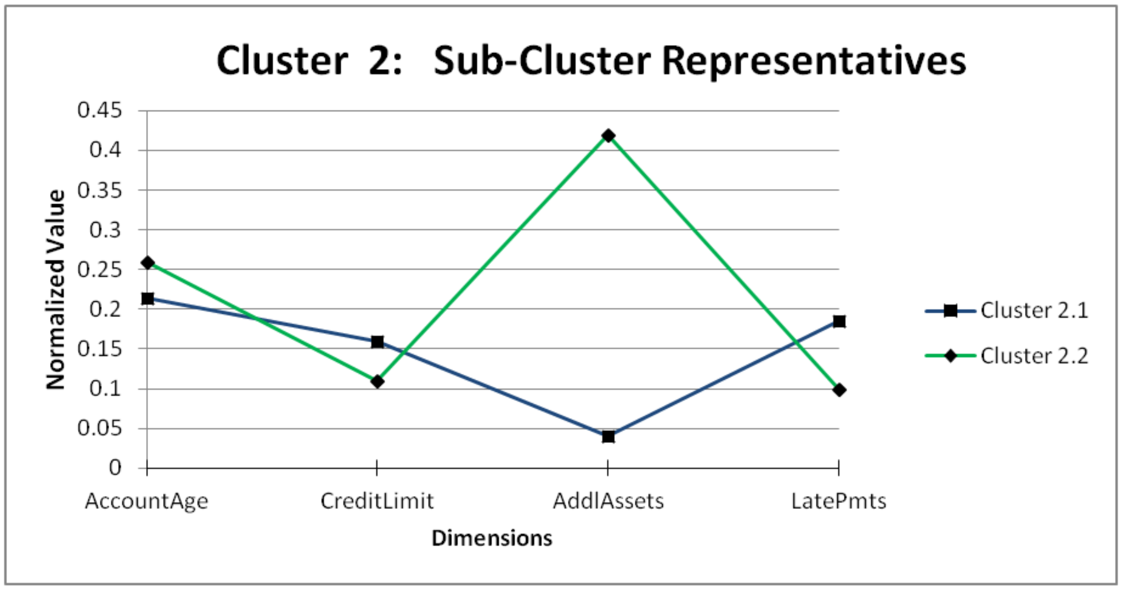


In this context, the Complete-Link two-cluster silhouette value is seen as vastly superior to all competing models.

3.3.4 Cluster 2 Model Creation, Visualization, and Evaluation

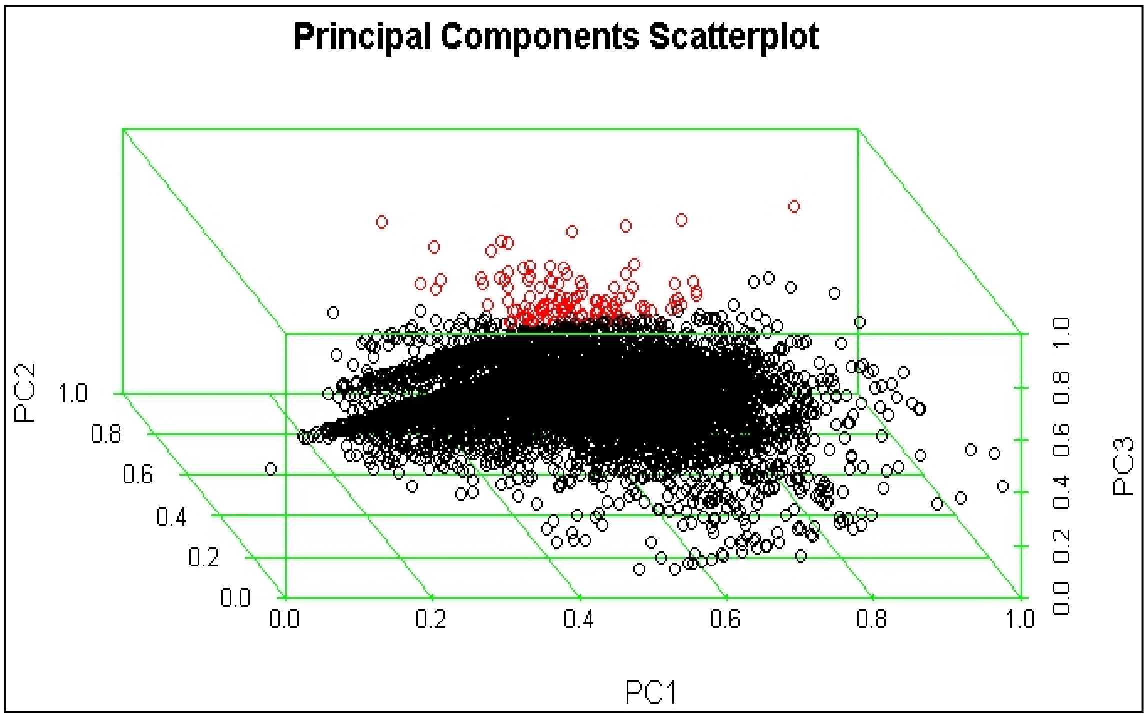
Moving forward, a Complete-Link two sub-cluster model is developed and further considered. The plot of cluster representatives follows.

Figure 13: Cluster representative values for Complete-Link, 2 cluster model



Initial visual information suggests each of the two segments created from cluster 2 represents a distinct customer type. More specifically, while the groups are primarily distinguished via additional assets, the other three dimensions also exhibit differences in terms of normalized representative values. In particular, cluster 2.2 is more favorable in that it encapsulates more established customers who possess substantially higher levels of additional assets and make fewer late payments, on average, than the other group. Interestingly, although cluster 2.2 represents clients with higher creditworthiness, these customers actually maintain lower mean credit limits. To gain additional perspective, a three dimensional principal components scatter plot is generated such that each object is shown and a color coding scheme is again employed to distinguish objects by cluster. This is presented in Figure 14.

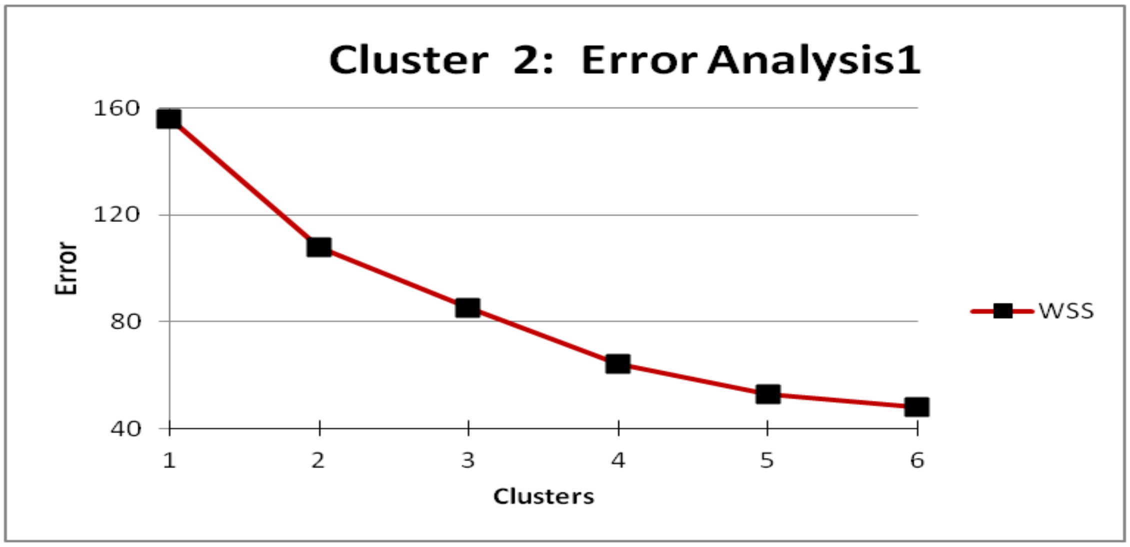
Figure 14: 3D plot of normalized principal components for cluster 2 data



The vast majority of points are shown in black and thus correspond to the majority cluster. In fact, this group is classified as cluster one (subcluster 2.1) by the complete-link model and contains 7,768 of the 7,888 observations (98.5%). By contrast, the minority group (subcluster 2.2) contains only 120 records (1.5%), and these are depicted as red in the above visualization. Overall, the plot offers some positive evidence for a two partition model.

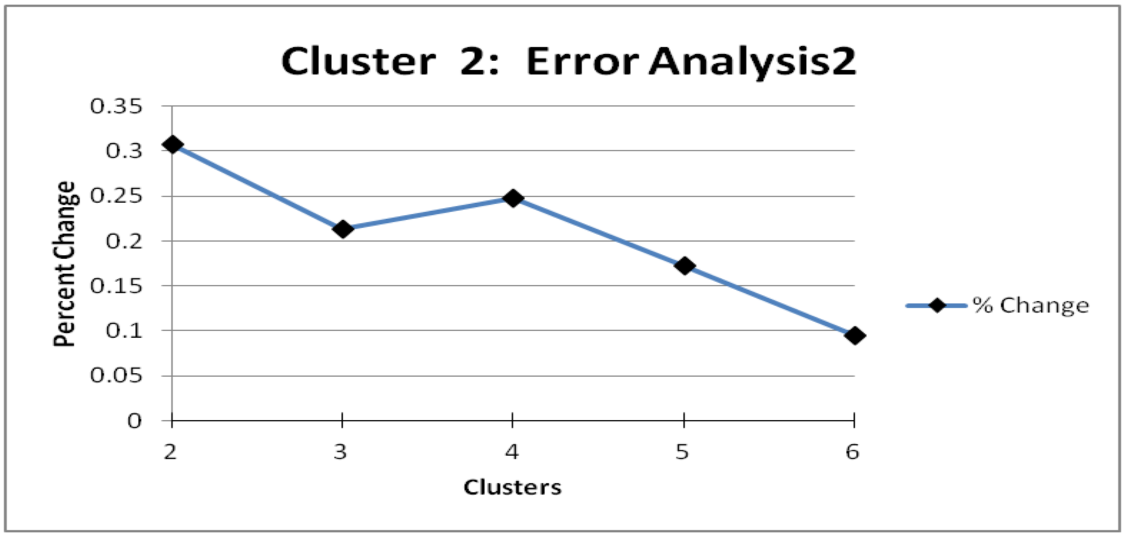
For additional perspective, a two-part error evaluation process is again implemented. First, "elbow" analysis is conducted, and the associated graph follows in Figure 15.

Figure 15: Cluster 2 - Sub-cluster error analysis 1



In this case, a slight elbow is detected at 2 clusters, suggesting that diminishing marginal returns occur in error reduction when moving from a 2 to 3 group solution. This provides some incremental evidence two partitions might be present in the data. Second, another graph is created showing percentage change in error reduction as the number of clusters increases. This is produced in Figure 16, and corresponds with the previous image.

Figure 16: Cluster 2 - Sub-cluster error analysis 2



Specifically, the highest percentage change occurs at two clusters. In addition, with the exception of a 4 group model, percentage change in error reduction declines rather sharply and consistently as number of clusters increases. Collectively, this suggests that, if multiple segments exist in cluster 2, then the maximum number is two. At this point, accumulated information argues for a 2 cluster model. However, prior to making a final decision, statistical testing is performed.

In particular, two-group Kruskal-Wallis non-parametric tests are again conducted for all dimensions. In this case, the data population consists of 7,888 records, with the majority (minority) cluster having 7,768 (120) objects. Because the minority group is extremely small and currently falls within the recommended group size range, initial testing is done for the population. Results follow in Table 10, and are significant for all dimensions.

Table 12: Cluster 2 - Kruskal-Wallis results (n=7,888)

Field	Chi-Squared	df	Significance Level
AccountAge	16.5200	1	**p < .001**
CreditLimit	7.2700	1	**p < .01**
AdditionalAssets	655.4600	1	**p < .001**
LatePayments	29.7000	1	**p < .001**

In comparing this table with cluster representatives in Figure 13, commonalities become apparent. For instance, additional asset representative amounts are very different for the two partitions, and the Kruskal-Wallis test is highly significant for this attribute (p < .001). Conversely, credit limit representative values appear more comparable for the two groups, and, not surprisingly, the Kruskal-Wallis test produces less significant results (p < .01). As a final measure, statistical tests are performed on a sample of the data. In doing so, a prerequisite is that number of objects in the sample from the minority cluster must fall within the recommended range for group size (i.e., about 30 to 200). Consequently, a 50 percent representative sample is taken such that about 60 records from the minority partition are anticipated to be in the sample. Outcomes are presented in Table 13, and, although lower in terms of significance, they nevertheless support a two sub-cluster model.

Table 13: Cluster 2 - Kruskal-Wallis results (n=3,944)

Field	Chi-Squared	df	Significance Level
AccountAge	10.0700	1	**p < .01**
CreditLimit	3.9278	1	**p = .04749**
AdditionalAssets	330.6800	1	**p < .001**
LatePayments	15.1510	1	**p < .001**

Incorporating a significance threshold of .01, three of the four dimensions are found to be significantly different in terms of representative values. In addition, if this threshold is expanded to .05, then statistically significant results are obtained for all attributes. In the aggregate, information argues for the existence of two partitions within the cluster 2 data. As such, this new model is retained, and the dimension values are de-normalized and presented in Table 14.

Table 14: Cluster 2 - De-normalized results for 2 sub-cluster model

	Cluster	
Dimension	**2.1**	**2.2**
AccountAge	98	118
CreditLimit	$10,499	$7,319
AdditionalAssets	2.24	4.49
LatePayments	6.71%	3.68%
Instances:	7,768	120
Percent of Total:	98.48%	1.52%

From this view, each segment indeed depicts a unique customer type. More specifically, cluster 2.2 contains clients with higher creditworthiness. These individuals have more established accounts, maintain a higher level of addditional assets, and make fewer late payments as a percent of total, on average, relative to cluster 2.1. Interestingly, the less creditworthy sector has a higher average credit limit, and this appears counterintuitive.

As previously mentioned, cluster 3 contains a very small membership. Therefore, it is not further examined relative to the nested cluster issue. Next, results from this and the previous chapter are combined into a single model.

3.4 Results

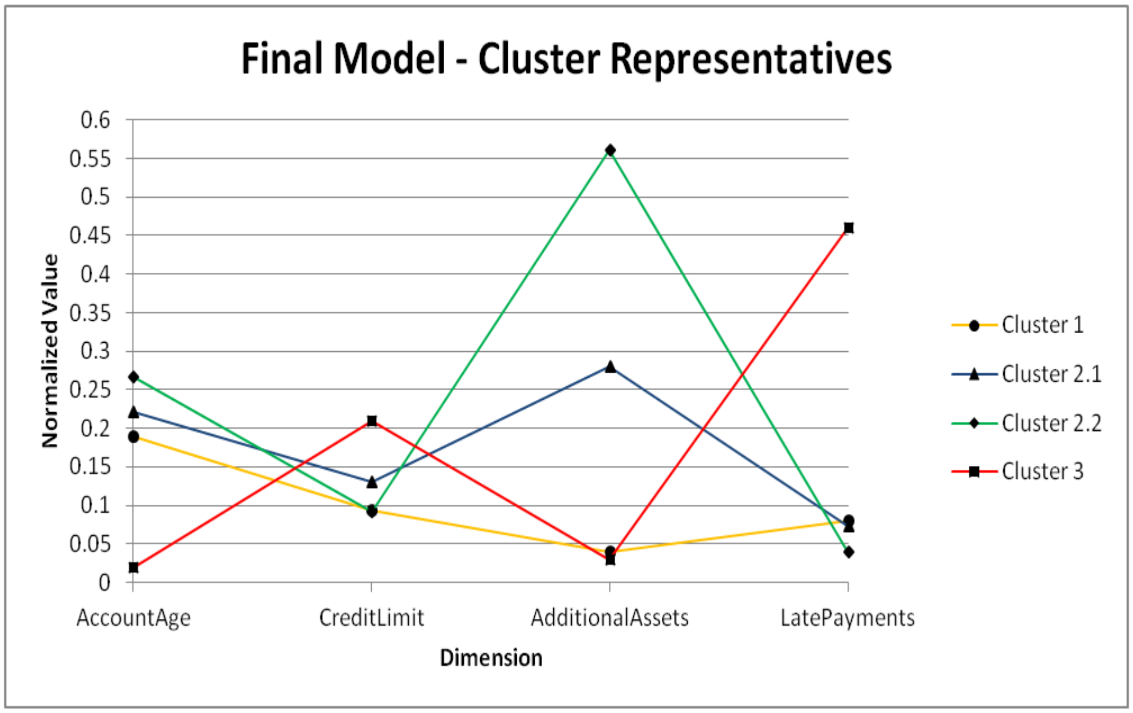
This final, synthesized model consists of four clusters, and the de-normalized representative values are shown in Table 15.

Table 15: De-normalized results for final synthesized 4-cluster model

	Cluster			
Dimension	**1**	**2.1**	**2.2**	**3**
AccountAge	83	98	118	9
CreditLimit	$7,462	$10,499	$7,319	$16,741
AdditionalAssets	0.27	2.24	4.49	0.27
LatePayments	6.95%	6.71%	3.68%	42.36%
Instances:	141,858	7,768	120	147
Percent of Total:	94.64%	5.18%	0.08%	0.10%

Several interesting observations are noted here. First, cluster 3 is clearly the least creditworthy group as it contains the least mature accounts, the lowest level of additional assets, and the highest incidence of late payments. Even so, these clients have been rewarded with the highest credit limits, on average. Fortunately, this group comprises a very small percentage of the customer base. Conversely, cluster 2.2 contains the most creditworthy individuals. They have the most established accounts, largest number of additional assets, and smallest incidence of late payments. Nevertheless, this partition has been granted the lowest average credit limit. Incidentally, clusters 1 and 2.1 reside intermediate to the other two segments. However, cluster 2.1 is more favorable in terms of creditworthiness, with more mature accounts, higher credit limits and level of additional assets, and a slightly lower propensity for making late payments, relative to cluster 1. Interestingly, the credit limits for these two partitions correspond logically, unlike those for clusters 2.2 and 3. The final model is graphed in terms of normalized cluster representatives in Figure 17.

Figure 17: Final clustering model



In this format, it is not extremely difficult to identify and rank the profiles. For example, cluster 2.2 pertains to clients with the most established accounts, greatest number of additional assets, and fewest late payments. Therefore, this group is the most creditworthy in terms of the depicted dimensions. However, this assumes that each attribute is given equal weighting, and, based upon information presented earlier relative to the importance of variables in computing credit scores, this is not the case in practice.

To clarify and finalize matters, a rudimentary scoring system is developed to incorporate individual dimension weights, thus facilitating a more technically accurate ranking of profiles. In achieving this, prior literature is considered in attempts to establish an appropriate weighting for each dimension. The system is implemented such that a higher score is indicative of greater creditworthiness (or lower risk), and this requires that one existing dimension is slightly reformulated. Specifically, because late payments is measured such that a lower amount is more favorable, the associated representative value for each profile is subtracted from one, and the resulting difference is used as the dimension value for calculating overall score. To generate a given score, each of the four representative amounts for a single cluster is multiplied by its corresponding weight. These products are then summed within each cluster to arrive at overall score. This process is repeated until all scores have been computed. In simple mathematical terms, the formula is as follows:

$$OverallScore = \sum_{i=1}^{4} D_i \, w_i$$

Where, D_i = value of the ith dimension
w_i = weight of the ith dimension

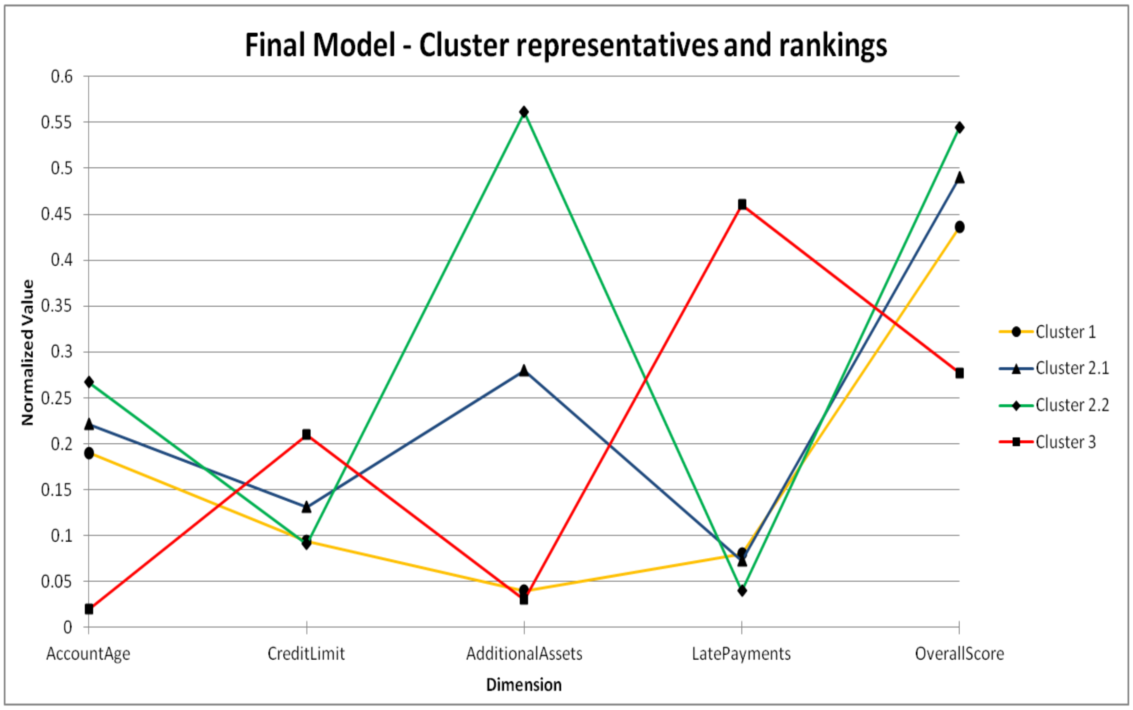
For reference purposes, individual weights are shown in table 16 below.

Table 16: Dimension Weights

Dimension Weights	
Dimension	**Weighting**
AccountAge	0.2
CreditLimit	0.25
AdditionalAssets	0.15
LatePayments	0.4

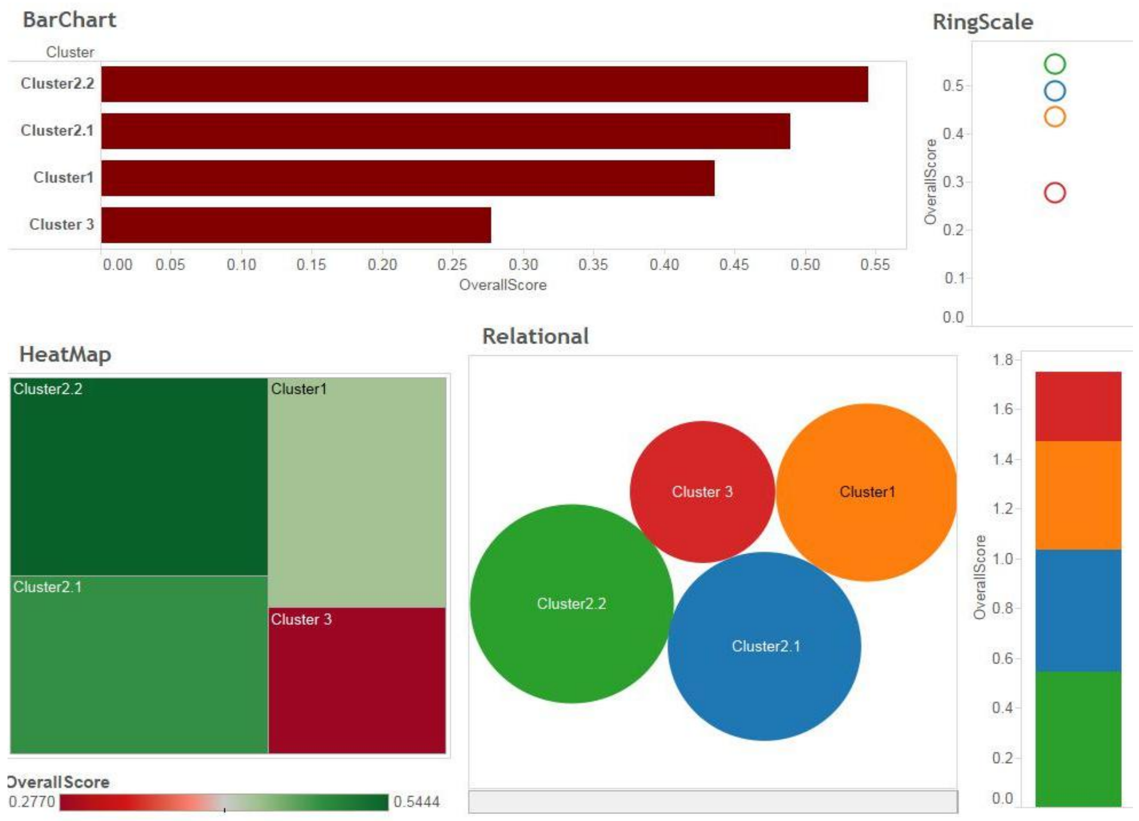
With conversion efforts complete, outcomes are plotted to include the final normalized numeric values as well as overall score information. The graph follows in Figure 18.

Figure 18: Representative values and overall scores by cluster for final clustering model



Although this depiction might be scrutinized on multiple grounds, it is nevertheless an objective approach to enhance customer profile identification and ranking tasks. Incidentally, the scoring system did not result in changing the rankings of the original partitions in terms of creditworthiness. For instance, cluster 2.2 was previously viewed as representing the most favorable client type, and this observation persists when incorporating the scoring method. In addition, cluster 3 was initially identified as containing the least creditworthy customers, and this remains true when using the score attribute as an exclusive basis for judgment. Essentially, the OverallScore dimension simplifies the process of noting where each cluster resides in terms of relative creditworthiness level. Furthermore, it considers the individual dimension weights, so that, presumably, a more accurate perspective of customer profiles is established. For added clarity, an alternate set of overall score visualizations is provided in the following image. Once again, a larger value is indicative of higher creditworthiness.

Figure 19: Dashboard view of customer profile overall scores



The above dashboard readily facilitates prioritization of segments for the employed attributes of creditworthiness. For example, it is clear that cluster 2.2 (cluster 3) relates to the most (least) favorable grouping.

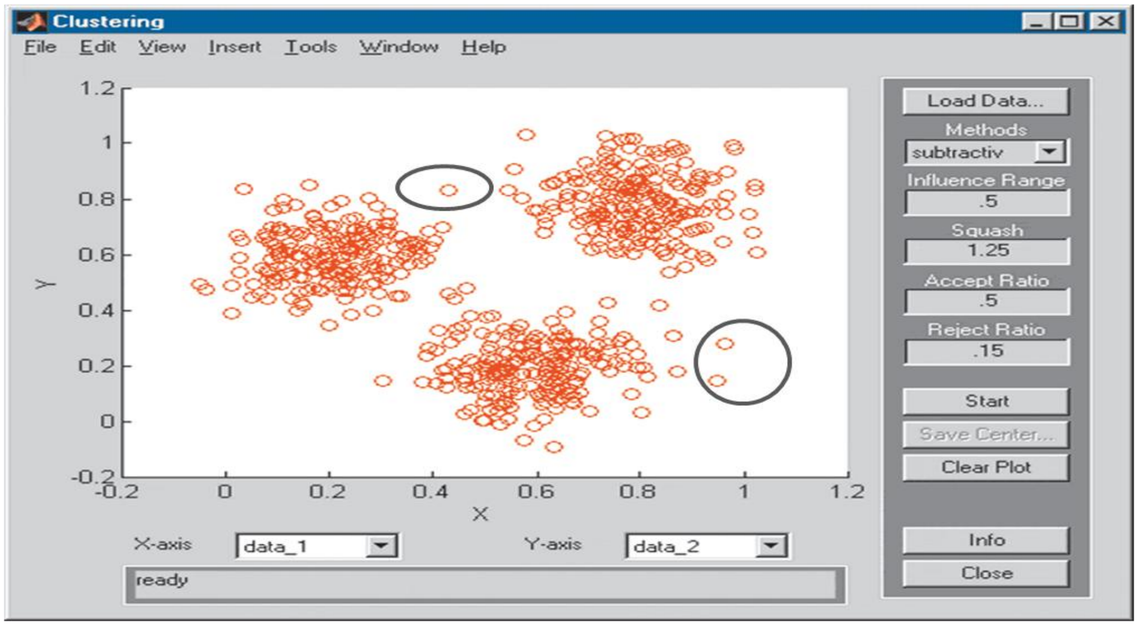
While there might be lingering limitations and questions concerning procedures used in scoring clusters within the customer profiling model, efforts thus far can at least be perceived as providing an objective basis upon which future work might build. Furthermore, the four unique credit card customer clusters can be referenced by bank personnel in establishing a formal policy wherein a distinct set of customer service protocols are tailored for each profile. In this way, customer representatives will subsequently have at their disposal a more systematic methodology for addressing the customer base that takes into account the creditworthiness of a customer prior to providing services including but not limited to determining satisfactory credit card fee discounts. In the auditing context, the established clusters facilitate the identification of potentially problematic records, thus assisting in the irregularity/fraud detection process. Moving forward, the next chapter explores the process by which established clusters are examined so as to prioritize objects for productive use of auditor resources in the fraud/irregularity discovery and investigation processes.

CHAPTER 4: OUTLIER DETECTION - LOCATING THE "NON-CONFORMING CLIENTS"

4.1 Background

Outlier detection is often an integral step in clustering. In highlighting this, Figure 20 provides a simplified, generic example of objects depicted in two-dimensional space.

Figure 20: Clusters in two-dimensional space



Methods subtractiv
Influence Range .5
Squash 1.25
Accept Ratio .5
Reject Ratio .15
X-axis data_1
Y-axis data_2
ready

In looking at the diagram, one likely notes three partitions. However, in each group, there are objects residing near the fringes. This raises questions suggesting these data points should be further examined. For instance, it may be that a subset of the records contain errors or are otherwise irregular items warranting formal investigation and resolution.

In this chapter, efforts are made to identify potential outliers within the relevant clusters finalized in the previous section. The reader should keep in mind that, while the method for outlier detection ultimately proposed, developed, and implemented in this chapter is used for identifying irregular customer records, it is applicable to ostensibly any outlier detection activity whereby data is able to be meaningfully represented in a numeric fashion.

Outliers have been historically described in a variety of ways. For example, Hawkins (1980) referred to an outlier as *"an observation which deviates so much from other observations as to arouse suspicions that it was generated by a different mechanism"*. Barnett and Lewis (1994) described an outlier as *"an observation which appears to be inconsistent with the remainder of that set of data"*. Irrespective of specific definition, an outlier, exception, or anomaly can be perceived as a point that is substantially different from other items in the set to which it presumably belongs. Outlier detection is a method for capturing those objects that are notably different from the others (Zimek et al., 2014).

4.2 Outlier Methods

The process of outlier detection obviously entails preliminary considerations. First, several methods have been addressed in the literature, including but not limited to classification, nearest neighbor, clustering, and statistical based anomaly detection methods (Chandola et al., 2009). Method selection is an important step, and will be at least partially driven by the data itself. For example, the presence of label information might suggest a classification-based approach is most suitable, whereas the absence of such information might indicate that a clustering or nearest neighbor-based solution is desirable. Given the nature of this study, outlier detection will obviously rely upon clustering-based techniques. For example, one assumption in clustering-based anomaly detection is that normal objects within a cluster are nearest the cluster representative while exceptions are furthest from this value. This intuition is adopted as one foundation for identifying whether data points qualify as anomalous.

Because the centroid is defined as the average value for a group of records within a cluster, the arithmetic average might be initially thought of as an appropriate reference point to which all other objects in a profile should be compared. However, the mean is a satisfactory measure of central tendency only when a normal distribution exists or is at least approximated. In many cases, this condition might not hold. In these scenarios, the median is a superior measure of central tendency. Furthermore, the median is as good as the mean when the data distribution is Gaussian or near-Gaussian, suggesting the median is the preferred metric in general. Given this, the median is employed as the measure of central tendency in this chapter. More specifically, the median will be computed for each dimension within each cluster, resulting in a median vector representative for each partition. The four median vectors then serve as the set of benchmark values for conducting outlier detection in the various clusters.

4.3 Outlier Measures

Second, the measure to be used for anomaly detection is an important consideration, because it influences performance and outcomes (Chandola et al., 2009). In addressing this, Zimek et al. (2014) propose the use of ensembles in outlier detection, whereby multiple methods are implemented in a type of majority voting context. Collectively, this indicates that, instead of relying upon a single metric, a meaningful combination of measures could enhance overall quality of results. In cluster-based outlier detection, proximity measures are often discussed (e.g., Tan et. al, 2005, Green and Rao, 1969). These entail an array of both distance and similarity measures. For instance, a subset of common distance measures are Manhattan, Minkowski, Euclidean, and Mahalanobis. Likewise, a few similarity measures include the Simple Matching Coefficient, Jaccard Coefficient, Cosine Similarity, and the Tanimoto Coefficient (Tan et al., 2006).

Prior research has found that two measures of a particular type will tend to be more highly correlated than two metrics from differing categories (Zimek et al., 2014). Furthermore, distance (similarity) measures have been found to be more suitable in high (low) density data (Tan et al., 2006). Given the various nuances, measure selection should obviously be done in a cautious and strategic manner.

4.3.1 Distance Measures for Outlier Detection

In reflecting upon distance measures highlighted above, three exhibit substantial comparability in terms of formulaic structure. For example, the general formula for Minkowski distance is:

$$\textit{Minkowski Distance} = \left(\sum_{i=1}^{n} |x_i - y_i|^p\right)^{\frac{1}{p}}$$

Where n = number of observations
p = parameter
x_i = the ith observation of x
y_i = the ith observation of y

In the above, p is a parameter and may theoretically be specified as any integer value above zero. However, when p = 1, Minkowski distance is reduced to Manhattan distance, and when p = 2, it becomes Euclidean distance. Because of these fundamental relationships, it is presumably not worthwhile to incorporate more than one of these three measures in a particular outlier detection ensemble.

In further exploring available distance measures, Chandola et al. (2009) find that Euclidean distance is popular in the context of outlier detection. In addition, no prior intuition exists concerning what an appropriate parameter setting for p would be in the case of Minkowski distance. Furthermore, when attempting to locate outliers in n-dimensional space, Euclidean distance would undoubtedly be more suitable than Manahattan. Therefore, Euclidean distance is initially selected as one candidate method for exception identification. The associated equation for Euclidean distance follows:

Euclidean distance = d(p,q) = d(q,p)

$$= \sqrt{(q_1 - p_1)^2 + (q_2 - p_2)^2 + \cdots + (q_n - p_n)^2}$$

$$= \sqrt{\sum_{i=1}^{n}(q_i - p_i)^2}$$

Where (in this study) ***d = distance between two vectors p and q***
p = a given customer record (vector)
q = median vector of a cluster (benchmark vector)

Interestingly, Mahalanobis distance is fundamentally distinct from the other three aforementioned distance measures in that it incorporates the covariance matrix of the data when computing distance information. In addition, it has been found to be quite successful in multivariate anomaly detection (Starkweather, 2013). However, a lingering question relates to the appropriate estimator for Mahalanobis distance. Holgersson and Karlsson (2012) investigated this phenomenon and found that, in very high-dimensional data where n (number of objects) and p (number of dimensions) were comparable, the traditional estimator performed poorly relative to a ridge-based estimator. Conversely, in cases where n was 3.33 times greater than p, both versions performed equally well. It was concluded that all estimators of Mahaloanobis distance performed adequately, except in cases of extremely high dimensionality wherein number of features was comparable to number of objects. In this chapter, there are only four dimensions involved in analysis. Furthermore, the smallest cluster contains 120 objects. Consequently, a traditional estimator of mahalanobis distance is clearly sufficient for outlier detection in this setting. The resulting formula is:

$$\textbf{\textit{Mahalanobis Distance}} = \sqrt{(\boldsymbol{x} - \boldsymbol{\mu})^{\boldsymbol{T}} \boldsymbol{S}^{-\boldsymbol{1}}} (\boldsymbol{x} - \boldsymbol{\mu})$$

Where** x **= an observation or object
μ = mean value (median in this study)
S = covariance matrix

At this point, two distance measures, Euclidean and Mahalanobis, are included in the emerging outlier detection scheme. Moving forward, other proximity measures are contemplated.

4.3.2 Similarity Measures for Outlier Detection

Similarity measures are able to assess the degree of likeness between any two objects. Within this domain, Tan et al. (2006) discuss the Simple Matching, Jaccard, Cosine Similarity, and Tanimoto coefficients. While these are often used in analysis of document data, a subset are useful elsewhere. More specifically, the Simple Matching and Jaccard coefficients both rely upon usage of binary data in determining the extent to which any two records are similar. Consequently, they are not applicable in the current study because non-binary, numeric dimensions are present. Conversely, Cosine Similarity and the Tanimoto Coefficient both entail the computation of vector lengths and dot products such that objects possessing non-binary, quantitative dimensions are able to be measured.

Cosine Similarity computes the cosine of the angle between two arrays x and y. In this paper, x is a record and y is the benchmark (median) vector. In calculating Cosine Similarity, a value of one stipulates the angle between two vectors is zero degrees, demonstrating the two objects are perfectly identical. At the other extreme, a value of zero means the angle between two vectors is 90 degrees, and, thus, the two objects are completely dissimilar (Tan et al., 2006). The associated calculation for Cosine Similarity follows:

$$\textit{Cosine Similarity} = \cos(x,y) = \frac{\boldsymbol{x} \cdot \boldsymbol{y}}{\|\boldsymbol{x}\|\ \|\boldsymbol{y}\|}$$

***Where** x = a vector x (record or object)*
y = a vector y (record or object); in this study, y is the median vector (benchmark vector)

The Tanimoto Coefficient is also referred to as the Extended Jaccard Coefficient. Like Cosine Similarity, it may be used in cases of non-binary data (Tan et al., 2006). Also, a value of one indicates complete similarity while zero dictates complete dissimilarity. The formula is:

$$\textit{Tanimoto Coefficient} = \frac{\boldsymbol{x} \cdot \boldsymbol{y}}{\|\boldsymbol{x}\|^2 + \|\boldsymbol{y}\|^2 - \boldsymbol{x} \cdot \boldsymbol{y}}$$

***Where** x = a vector x (record or object)*
y = a vector y (record or object); in this study, y is the median vector (benchmark vector)

While Cosine Similarity and the Tanimoto Coefficient have identical numerator terms, they differ considerably in denominator structure. Consequently, the two measures could each provide incremental value in the outlier detection process. Furthermore, because two distance measures are being incorporated, it seems advisable to employ an equal number of similarity measures so as to construct a balanced evaluation scheme. With the measurement selection phase complete, three additional considerations emerge.

4.4 Outlier Detection Method - Additonal Considerations

First, the distance measures are such that objects with larger values have higher probabilities of being anomalous. On the other hand, similarity measures result in assessments that are exactly opposite of this. To streamline and simplify outlier identification, all measures should operate in the same direction. In achieving this, similarity measures are converted to dissimilarity measures by simply subtracting the associated formulas from one (Tan et al., 2006).

Second, a threshold value will be helpful for suggesting a rough cutoff point relative to outlier status. Chandola et al. (2009) discuss two possibilities for this. The first is based upon the box plot (Tukey, 1977), and is referred to as the box plot rule. One part of this rule argues that any object greater than 1.5 times the inter-quartile range (75th percentile - 25th percentile) below the 25th percentile or above the 75th percentile is deemed anomalous. Acuna and Rodriguez (2011) elaborate on the second part of the box plot rule by discussing both mild and severe outliers. They mention that mild anomalies are those lying beyond 1.5 times but no more than 3 times the inter-quartile range below the 25th and above the 75th percentiles. An object residing more than 3 times the inter-quartile range beneath the 25th percentile or beyond the 75th percentile is treated as a severe outlier.

An alternative approach for threshold establishment suggests that any record beyond +/- 3 standard deviations from the mean qualifies as an exception. Initially, at least partially because of its non-parametric nature, this study incorporates the box plot approach. It should be emphasized that the benchmarks will serve as mere heuristics, thus helping to flag the set of most significant objects.

Third, a method is needed for ranking identified exceptions in an aggregated manner so the most egregious outliers may be investigated first. Issa (2013) found that a large number of anomalies (i.e., "exceptional exceptions") are often generated such that examination of all outliers is impractical. Furthermore, many exceptions could actually be false positives that unnecessarily consume investigatory resources. To address this problem in one particular context, a methodology based upon a predictive ordered logistic regression model is incorporated to review and prioritize control risk assessments so that investigation emphasis is placed upon the most problematic issues (Issa & Kogan, 2014). In another study, a suspicion score system is developed and implemented for prioritizing records relative to internal control rule violations in transactional data, thereby again confronting the issue of "exceptional exceptions" (Issa et al., 2015). Once again, this procedure facilitates the productive use of scarce resources in examining problematic records. With this in mind, and, given that four separate measures are being adopted in this paper, a scoring procedure is implemented for aggregating and ranking objects in each evaluated profile.

In particular, within each group, each of the four measures is normalized on a (0,1] scale in a manner comparable to how quantitative dimensions (e.g. AccountAge, CreditLimit, etc.) were preprocessed in chapter one. However, the formula used in this stage is simplified, and, most important, it ensures that a normalized measure for a given object will not have a value of zero unless it is identical to the median vector. While it might be that either form of normalization achieves substantially similar results, the selected alternative is more technically precise. This formula for normalizing distance and dissimilarity measures was applied in Scandizzo (2005) and follows:

$$\boldsymbol{Normalized\,Value} = \frac{\boldsymbol{Actual\,Value}}{\boldsymbol{Maximum\,Value}}$$

Following normalization, the outlier score for a given object is computed as the sum of the four normalized measurements for that record. With this approach, an outlier score can lie be between zero and four, inclusive. This anomaly detection method is fully automated within R, and is written such that individual weights can be readily manipulated for each measure in cases where differential weighting is warranted. By default, each metric is given equal weight, and, because it is not known in advance whether unequal weightings should be applied, default settings will be used in the following analysis section.

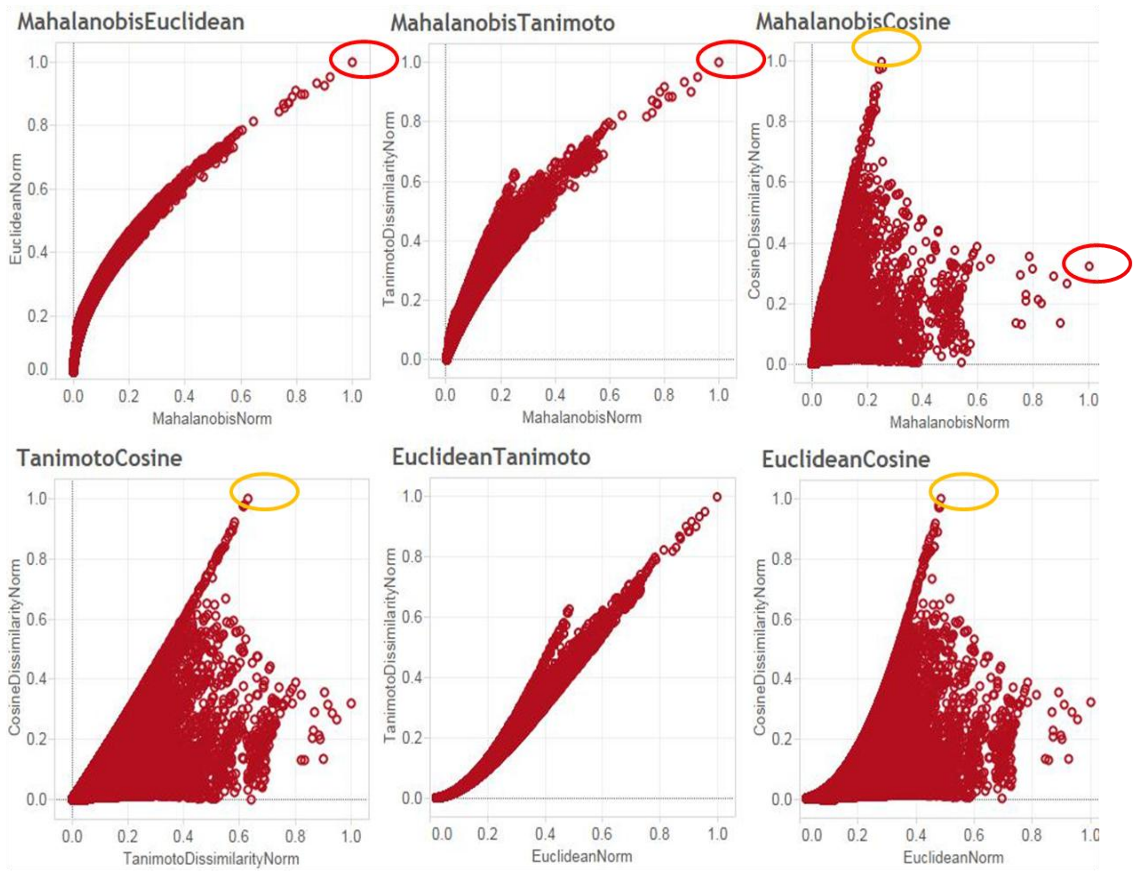
4.5 Analysis

Relying upon the aforementioned methodology, each cluster is separately explored to determine the extent to which exceptions might exist. For convenience, the code pertaining to this section is provided in Appendix C. In addition, the processing routine results in an output file that can be further analyzed in other environments such as Tableau, Clickview, Excel, etc.

4.5.1 Cluster 1 Evaluation

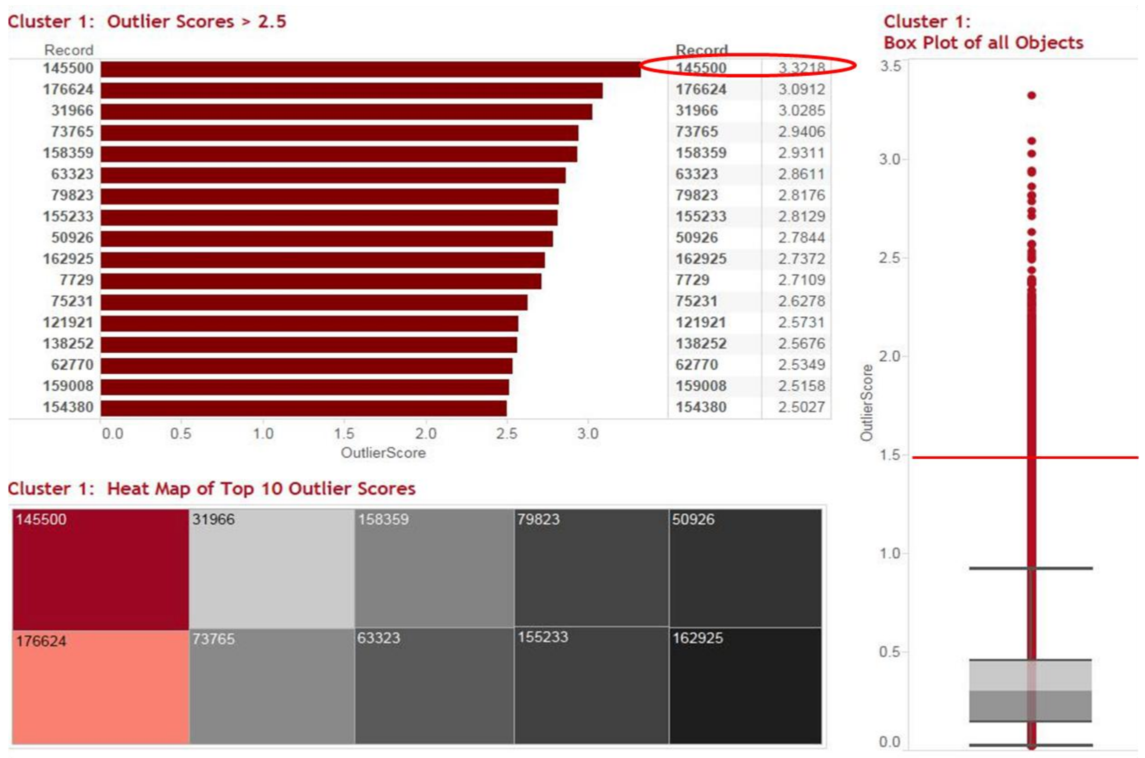
Initially, cluster 1 is visualized using Tableau, and an array of pair-wise complete scatter plots follows in Figure 21.

Figure 21: Outlier Plots - Cluster 1



In each of the above graphs, the representative value (i.e. median vector) exists at the origin. Therefore, objects appearing furthest from this region are most likely to be anomalous. Several items are well away from the origin, but not all measures yield consistent evidence. For example, record 145500 is highlighted in red in the upper set of images. In terms of Mahalanobis, Euclidean, and Tanimoto, it is identified as furthest from the median value. Conversely, it is relatively comparable to the median in terms of Cosine Similarity. In another example, record 183782 is highlighted in orange in three regions of the above collection of plots. Based upon Cosine Similarity, this object is most different from the cluster representative. On the other hand, it does not appear as exceptional according to the remaining three measures. Given the lack of consistency, it is useful to consider outlier score information created through aggregating the four measurements for each object. A dashboard relating to this follows in Figure 22.

Figure 22: Outlier Dashboard - Cluster 1



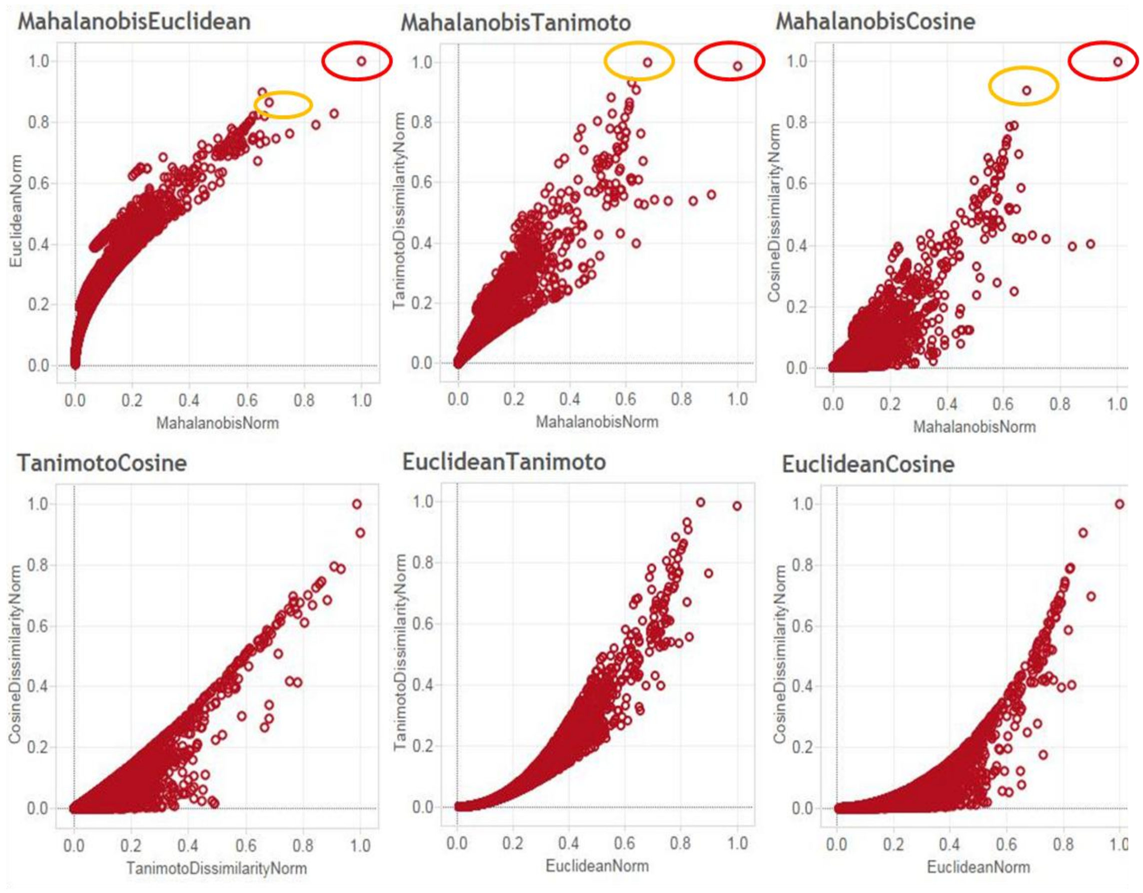
Record	
145500	3.3218
176624	3.0912
31966	3.0285
73765	2.9406
158359	2.9311
63323	2.8611
79823	2.8176
155233	2.8129
50926	2.7844
162925	2.7372
7729	2.7109
75231	2.6278
121921	2.5731
138252	2.5676
62770	2.5349
159008	2.5158
154380	2.5027

Because this cluster contains over 140,000 records, all views except for the box plot are limited to objects with outlier scores above 2.5. Also, recall that a point with a higher outlier score is more likely to be anomalous. Consequently, it is not alarming that record 145500 possesses the largest score (i.e., 3.3128), given it was identified as the most probable outlier by three of the measures. Interestingly, record 183782 does not appear in the restricted dashboard view, even though it was labeled as most dissimilar to the cluster representative according to Cosine Similarity. This occurred because the object received low measurements via the remaining three metrics. Of additional interest, only three records possess outlier scores above 3. In the box plot view, points above the red horizontal line correspond to those that are greater than three times the inter-quartile range above the 75th percentile. Such objects are viewed as potentially egregious outliers, and, as can be seen, a large number of exceptions are identified. Fortunately, outlier scores provide a ranking mechanism, such that objects with higher scores are more likely to be anomalous. Consequently, investigations should begin with the record having the highest outlier score, and proceed in a descending fashion.

4.5.2 Cluster 2.1 Evaluation

Moving forward, cluster 2.1 is examined in a similar manner, and a set of standard plots is generated in the following view.

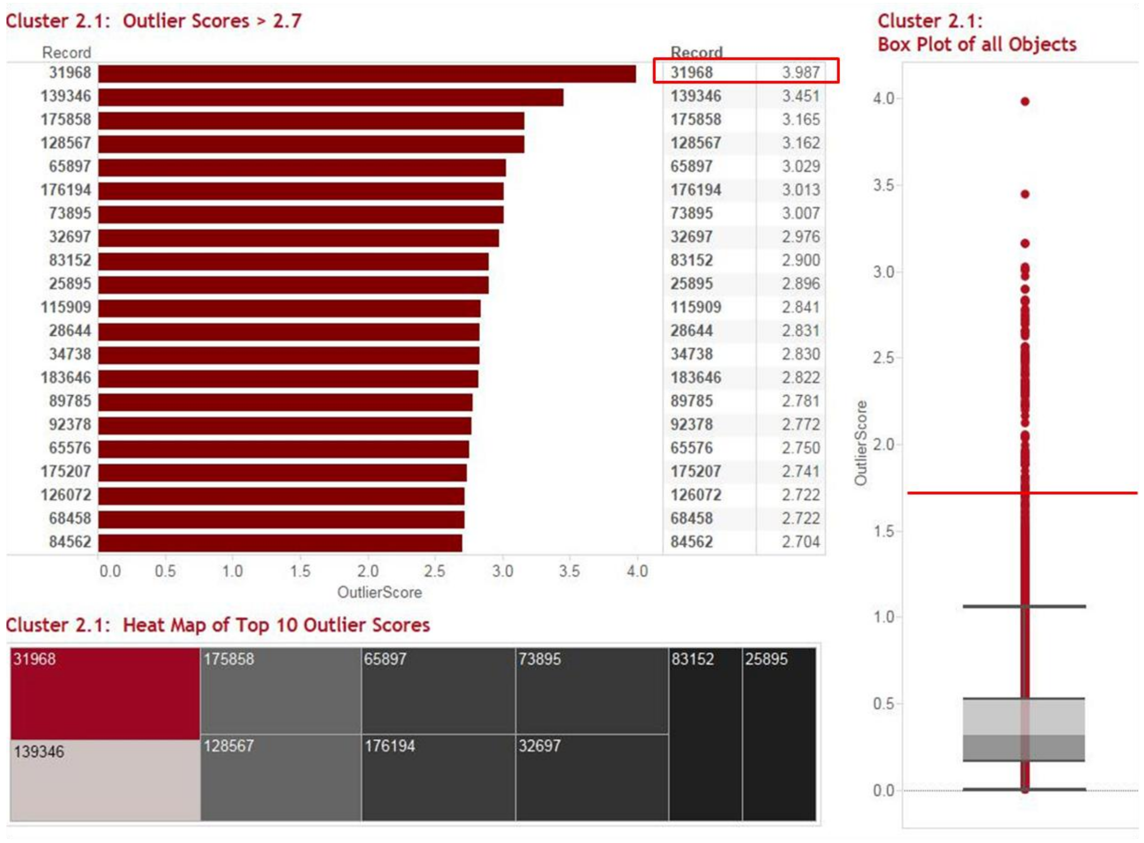
Figure 23: Outlier Plots - Cluster 2.1



In the above, one object clearly emerges as most noteworthy regardless of measure considered. Specifically, record 31968 is circled in red in the upper set of views, and is the most anomalous according to each of the four metrics. In fact, separation between this point and and the next most probable outlier appears substantial. For example, while record 31968, highlighted in orange above, is seen as second most different from the representative value in terms of the two similarity measures, it is less anomalous based upon the two distance measures.

Moving forward, an outlier dashboard is again constructed to provide an ordered presentation of findings. Incidentally, this cluster contains nearly 8,000 records. As such all but the box plot view is limited to data points with outlier scores above 2.7. The dashboard is reproduced in Figure 24, and allows for some interesting observations.

Figure 24: Outlier Dashboard - Cluster 2.1



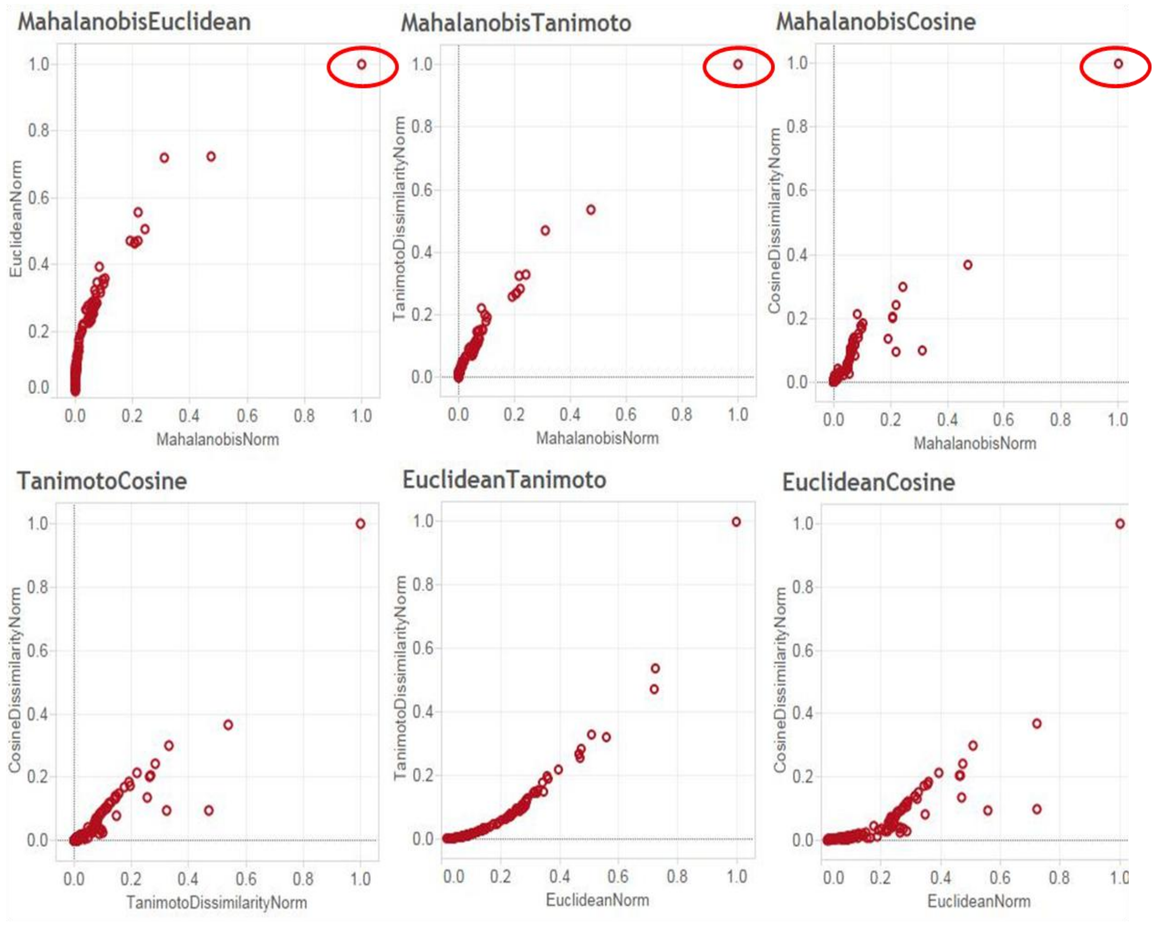
Record	
31968	3.987
139346	3.451
175858	3.165
128567	3.162
65897	3.029
176194	3.013
73895	3.007
32697	2.976
83152	2.900
25895	2.896
115909	2.841
28644	2.831
34738	2.830
183646	2.822
89785	2.781
92378	2.772
65576	2.750
175207	2.741
126072	2.722
68458	2.722
84562	2.704

First, record 31968 has an extremely high outlier rating of 3.987. This is about 16 percent higher than the second largest score, suggesting that the object stands out even among the set of potential anomalies and is therefore of particular interest. Second, the box plot view shows that objects with scores above about 1.7 are viewed as severe outliers. While this again appears to be a case of "exceptional exceptions" (Issa, 2013), only 106 records would be subjected to further review based upon this criterion. In addition, recall that the outlier score effectively ranks each object such that investigation ensues by first examining the record with the highest score and continuing in a descending manner.

4.5.3 Cluster 2.2 Evaluation

At this juncture, cluster 2.2 is visualized using Tableau, and an array of pair-wise complete scatter plots follows in Figure 25.

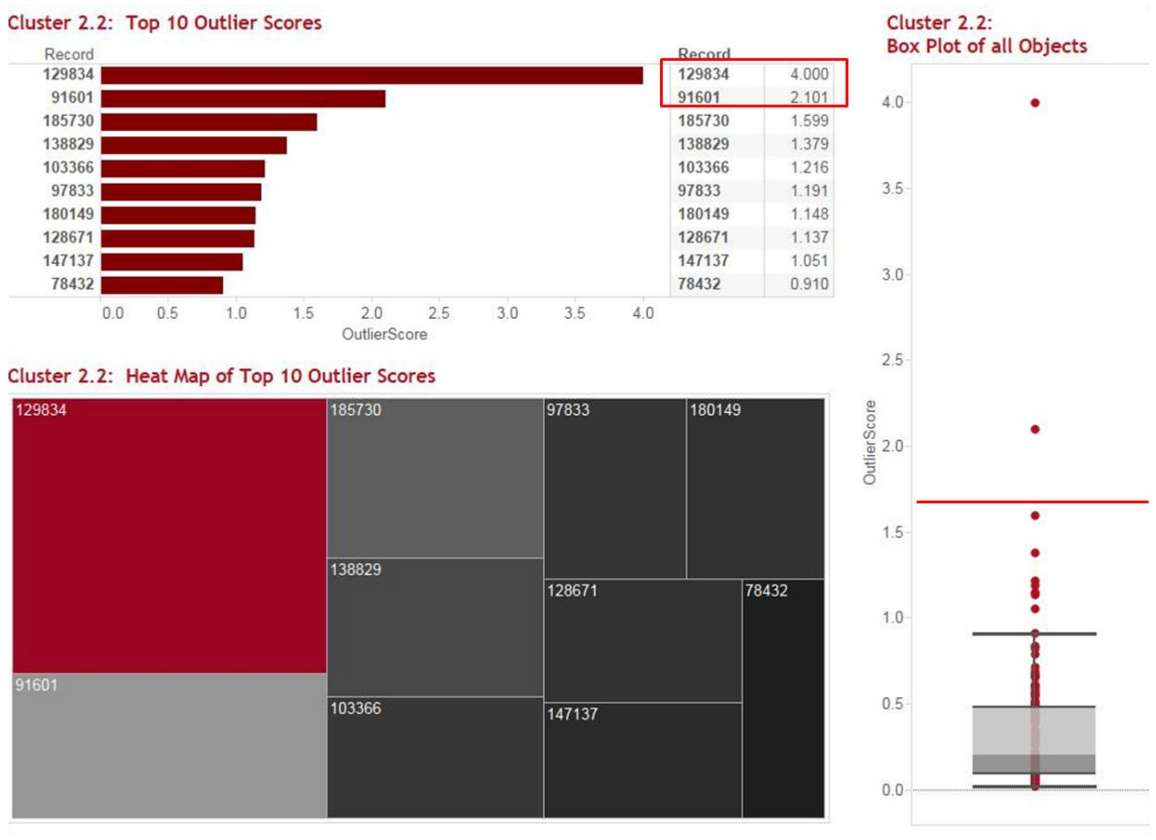
Figure 25: Outlier Plots - Cluster 2.2



This group only contains 120 points and is thus classified as a small membership cluster. Given this, an outlier detection exercise is not specifically required here. In practice, all objects with small membership status are considered problematic and should be targeted for further review. However, to maintain consistency, all clusters are visualized in this chapter. In the above figure, although all points are deemed anomalous, one is significantly different from the others. Specifically, record 129834 is circled in red in the upper set of images, is well separated from the other objects, and yields the highest assessment by all four measures.

At this point, the outlier dashboard view is incrementally beneficial to consider. This follows in Figure 26.

Figure 26: Outlier Dashboard - Cluster 2.2



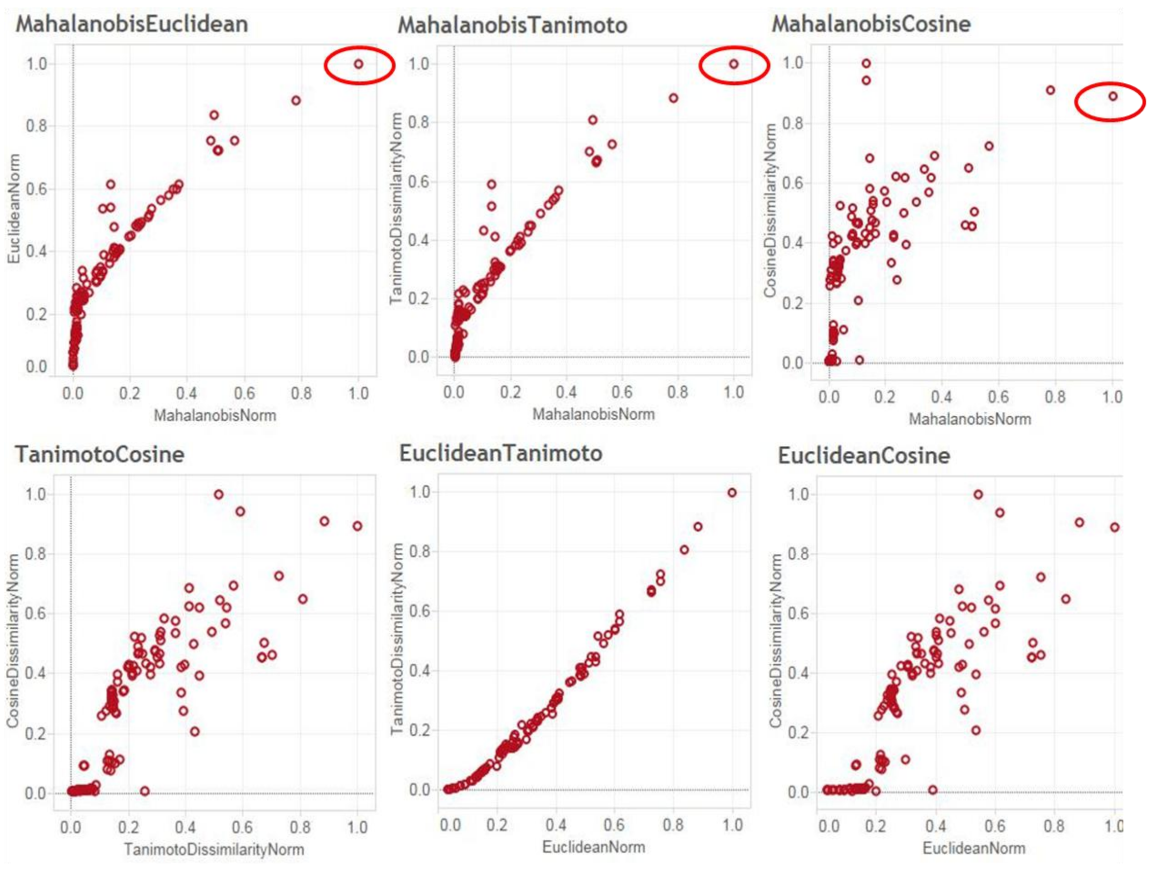
Record	
129834	4.000
91601	2.101
185730	1.599
138829	1.379
103366	1.216
97833	1.191
180149	1.148
128671	1.137
147137	1.051
78432	0.910

Record 129834 received the highest possible outlier score of 4. Furthermore, this value is nearly twice that of the second most anomalous object, suggesting it indeed is an outlier among outliers. Also, according to the box plot rule, only two records (i.e., 129834 and 91601) are stipulated as substantial outliers. However, this heuristic need not be referenced as this is a small membership cluster whereby all objects are viewed as outliers by definition. Nevertheless, it is interesting to appreciate the relationships between the clustered set of points.

4.5.4 Cluster 3 Evaluation

Finally, cluster 3 is examined in a similar manner, and a set of standard plots is generated.

Figure 27: Outlier Plots - Cluster 3

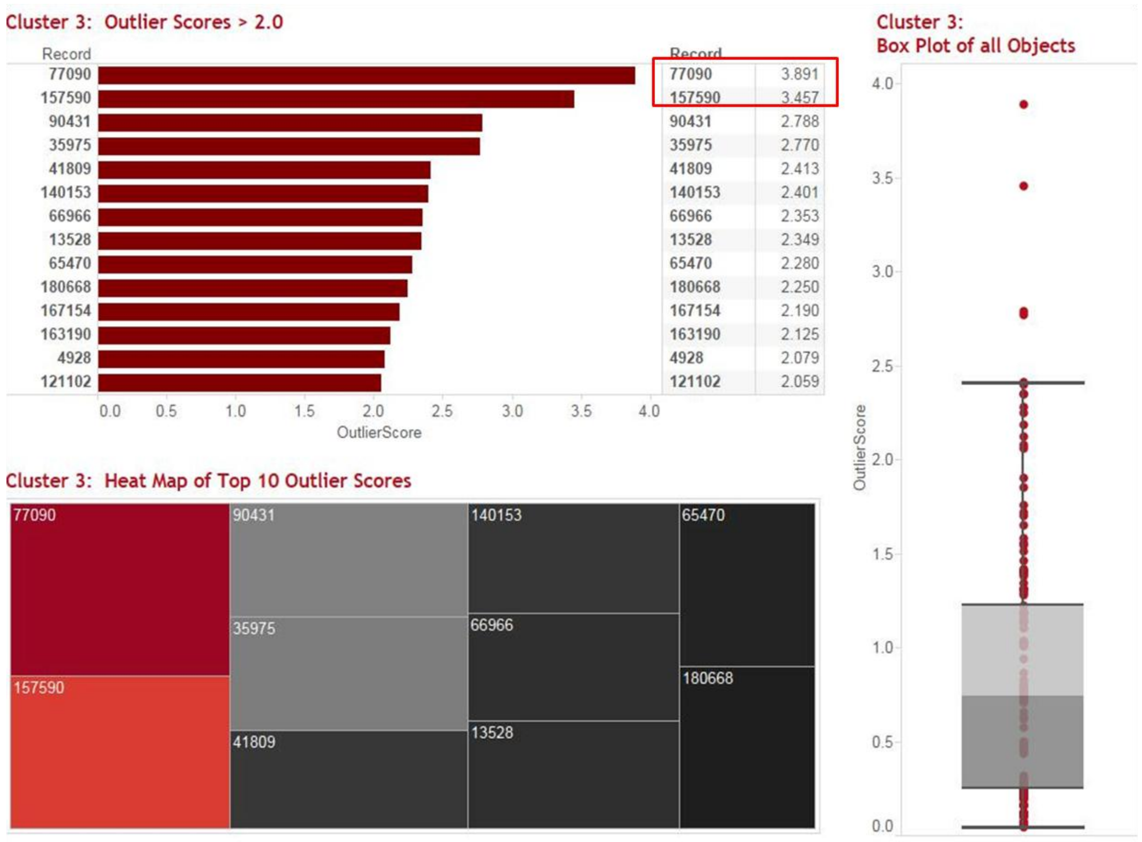


Note that cluster 3 only has 147 records, and, as such, is also a small membership cluster. Therefore, all records would normally be investigated.

Nevertheless, it is interesting to note that one object clearly emerges as the most prominent. In particular, record 77090 is circled in red in the upper set of views, and is the most anomalous according to three of the four metrics. In addition, it is seen as fourth most dissimilar according to Cosine Similarity.

Moving forward, an outlier dashboard is again constructed to provide for an ordered presentation of findings. This is reproduced in Figure 28.

Figure 28: Outlier Dashboard - Cluster 3



Record	
77090	3.891
157590	3.457
90431	2.788
35975	2.770
41809	2.413
140153	2.401
66966	2.353
13528	2.349
65470	2.280
180668	2.250
167154	2.190
163190	2.125
4928	2.079
121102	2.059

Record 77090 has the highest outlier score, and is thus considered the most anomalous object in this cluster. Also, according to the box plot rule, no records are identified as egregious outliers. However, once again, the heuristic is not necessary because this is a small membership cluster wherein all objects are subject to investigation.

The above outlier detection method demonstrates ability to not only identify potential exceptions, but also prioritize them in a manner so as to facilitate productive use of valuable and scarce investigatory resources. As mentioned previously, although the scheme is applied in an effort to locate customers who do not fit established profiles, it can be implemented in ostensibly any outlier detection activity for which the associated data is able to be represented numerically. For example, auditors could employ this method in locating anomalous transactions for the purposes of fraud discovery. Also, the approach could be used during the course of an audit for other activities including but not limited to audit planning, risk assessment, and analytical procedures. One example of such an application follows in the next section.

4.6 Outlier Detection Implementation Example - Ratio Analysis

Ratio analysis is a common analytical procedure in auditing. It typically entails evaluating four different types of measures, including liquidity, leverage, profitability, and activity ratios (Whittington and Pany, 2008). Historically, a univariate approach is adopted, whereby each ratio is considered in isolation and compared against the industry benchmark. While such a piecemeal technique has value, it fails to identify relationships exhibited via the combination or synthesis of all examined ratios. For example, in anomaly detection, objects identified as outliers in univariate space are often not found to be multivariate outliers (Starkweather, 2013). This suggests that a multivariate scheme is important to consider, and consists of treating the set of company ratios as an array to be compared with the industry benchmark vector of identical ratios (e.g., median, mean, etc.).

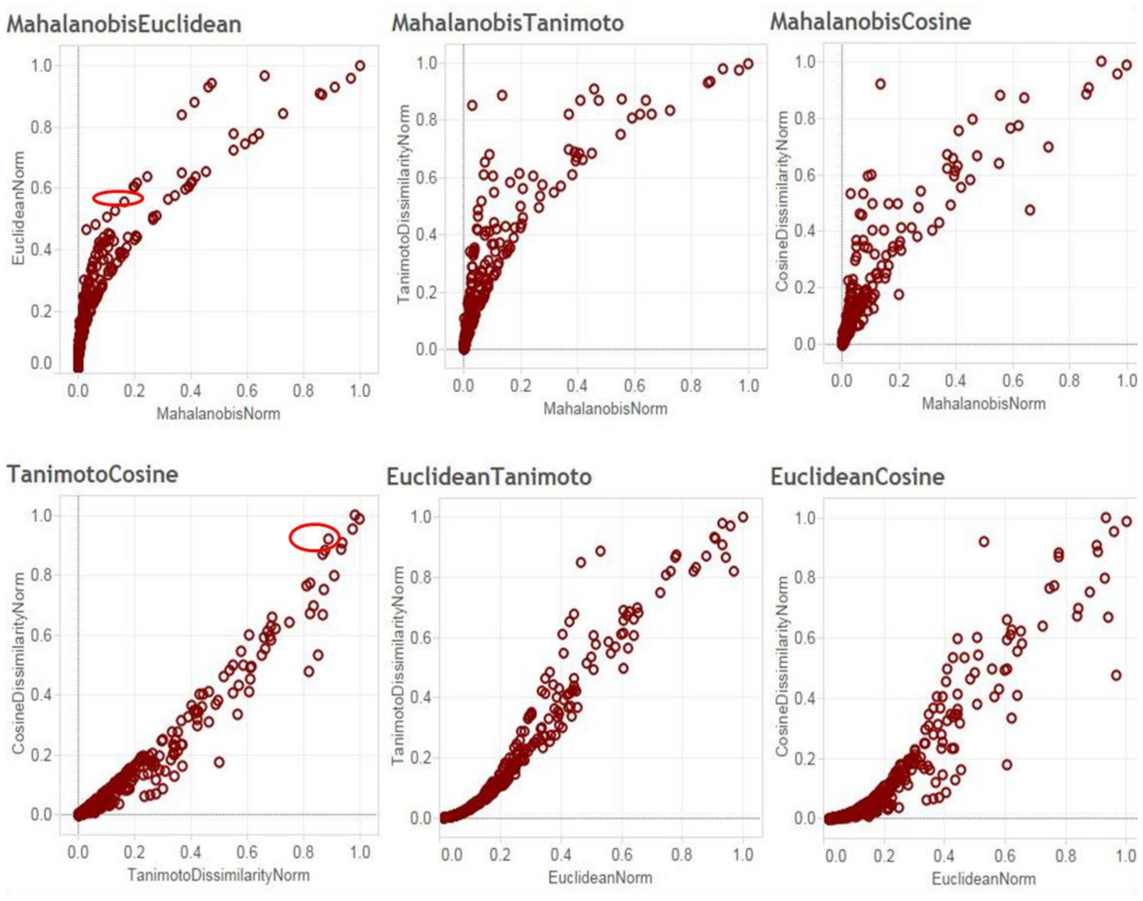
In achieving this, an assortment of measures of differing types might be considered. For instance, two popular distance-based metrics include Mahalanobis and Euclidean distance. Furthermore, two common similarity measures are Cosine Similarity and the Tanimoto Coefficient (Tan et al. 2005). In deciding what measure(s) to incorporate, two issues must be confronted. First, no metric is strictly superior, and ensembles of multiple approaches have been shown to be particularly effective (Zimek et al., 2014). In fact, each measure possesses a set of strengths and weaknesses, and these should be contemplated in advance of measure selection. Second, two metrics in a particular category tend to be more highly correlated than two measures of differing types (Zimek et al., 2014). Given these observations, an approach that employs a combination of measures might be explicitly considered.

In the analysis example that follows, the four aforementioned measures are used individually and in the aggregate to perform multivariate ratio analysis of firms in the retail industry. The compiled data is from Compustat, consists of entities with Standard Industrial Classification (SIC) codes between 5000 and 5999, and pertains to business operations during the 2013 fiscal year. Evaluated ratios are as follows: 1) current ratio, 2) long-term debt-to-equity ratio, 3) return on assets, and 4) inventory turnover. In preparing the data for analysis, ratios are standardized so that no single item will dominate outcomes. For example, consider the current ratio and inventory turnover. A typical current ratio might be about 2, whereas the usual inventory turnover value would be significantly higher (e.g., 7). Also, turnover would tend to occupy a larger range relative to the current ratio. If the amounts are not standardized, then turnover would dominate evaluation, thus biasing results.

In computing distance and similarity measurements, the median firm is used as the benchmark or comparison array. For example, in calculating a given measurement for a particular company, the organization's ratio vector and the median firm's ratio vector are the computational inputs. In addition, after calculating similarities, results are converted to dissimilarities. In this way, a larger value for any given measure is always indicative of greater absolute distance or difference from the median firm. Finally, for each entity, the four measurements are normalized on a (0,1] scale, and summed to arrive at a final outlier score (as previously described). In the multivariate ratio analysis task that follows, company code 1106838 assumes the role of auditee.

To maximize efficiency, data processing is fully automated in R, and the resulting output file is used for subsequent visualization in Tableau. A pair-wise complete set of normalized distance/dissimilarity plots for all objects follows in the next figure.

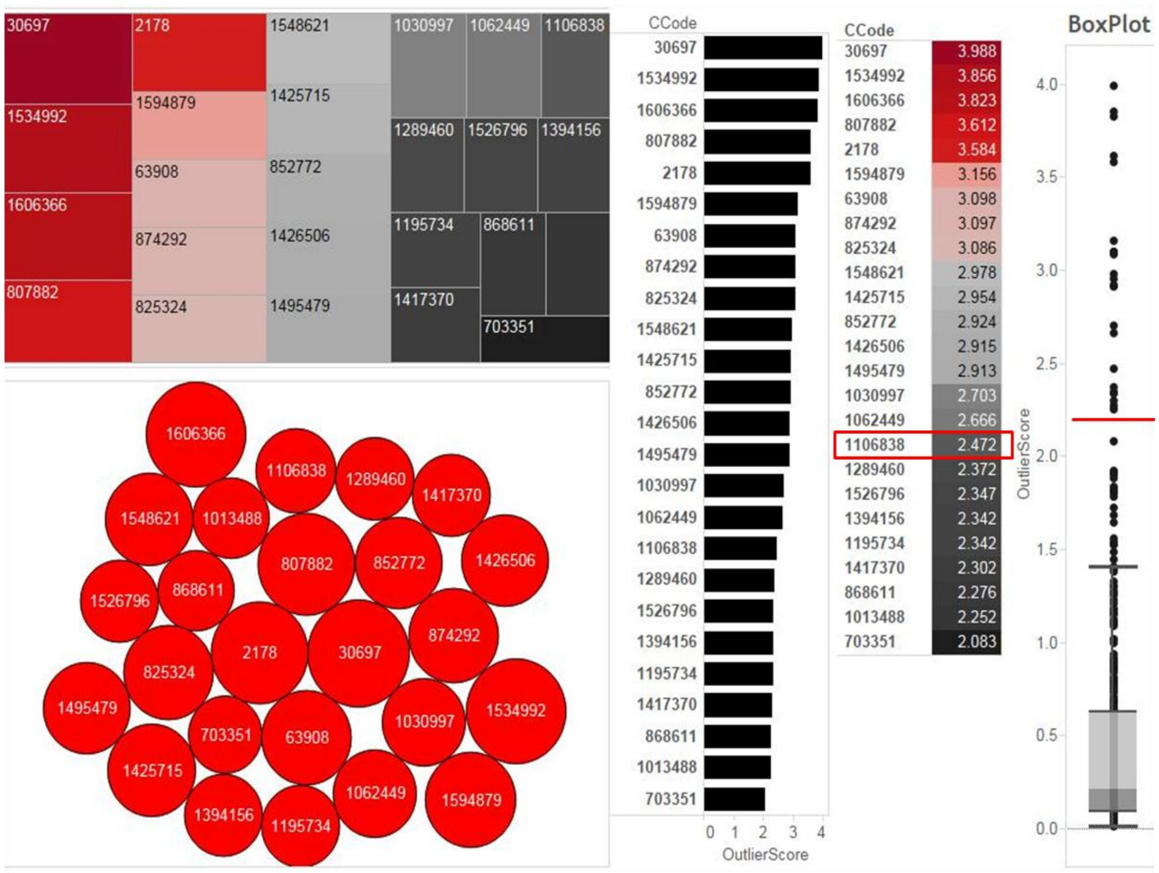
Figure 29: Plots - Multivariate Ratio Analysis



In each image, the median firm exists at the origin. Therefore, objects farther from this location are more distant/different from the industry benchmark. For example, the auditee is circled in the lower left graph, and is identified as substantially different from the median firm in terms of both Tanimoto and Cosine dissimilarities. This client is again highlighted in the upper left plot. In this case, the object is not as far from the origin when only considering distance measures, and certainly not perceived as anomalous when using Mahalonobis distance as the exclusive criterion. While this viewpoint offers preliminary value, an aggregated representation of all measures yields more specific insights.

In achieving this, each metric is given equal weighting. For each object, the four measurements are summed during the data processing step to yield a final outlier score. Because each measure is normalized on a (0,1] scale, the maximum score for an object is four. The outlier score dashboard follows in Figure 30.

Figure 30: Outlier Score Dashboard - Multivariate ratio analysis



CCode	
30697	3.988
1534992	3.856
1606366	3.823
807882	3.612
2178	3.584
1594879	3.156
63908	3.098
874292	3.097
825324	3.086
1548621	2.978
1425715	2.954
852772	2.924
1426506	2.915
1495479	2.913
1030997	2.703
1062449	2.666
1106838	2.472
1289460	2.372
1526796	2.347
1394156	2.342
1195734	2.342
1417370	2.302
868611	2.276
1013488	2.252
703351	2.083

Except for the box plot, image views are restricted to organizations with outlier scores of at least 2.0 so as to reduce clutter. A heat map is shown in the upper left section of the dashboard, and the size and color of each rectangle correspond to the relative magnitude of an outlier score. For instance, company 30697 has the shape with the largest area and most intense color scheme.

Consequently, it is viewed as having the highest probability of being anomalous. A comparable observation can be made from examining the lower left image. In this case, size alone is indicative of outlier status. Once again, company 30697 is depicted as the most irregular object, although this finding is not as easy to deduce.

Of the 25 entities specifically depicted, the auditee has the 17th highest outlier score (i.e., 2.472), suggesting that, while several other firms are more likely to be anomalous, the auditee is nevertheless worthy of further investigation. For example, the box plot image shows objects with outlier scores above about 1.4 fall beyond the "normal" range of the data distribution. More specifically, about 98% of points are expected to lie between the upper and lower horizontal lines of the box plot. Also, a red horizontal line is stationed at three times the inter-quartile range above the 75th percentile such that, objects above that line are seen as severe cases. This stipulates records with scores above about 2.2 qualify as potentially serious outliers. Incidentally, the auditee occupies this category.

To provide for contrast, ratios of the: 1) median firm, 2) entity with the lowest outlier score, 3) auditee, and 4) object with the highest outlier score are all presented in Table 17.

Table 17: Ratio Comparisons - Multivariate ratio analysis

Company	Current Ratio	LT Debt-to-Equity	ROA	Inventory Turnover	Outlier Score
Median	1.66	.29	.04	7	0.000
791519	1.56	.16	.05	7	0.016
1106838	.31	.07	**-.38**	26	2.472
30697	2.64	.73	.01	152	3.988

In the data set, overall median and mean outlier scores are only .214 and .526, respectively. As such, although there are 16 entities with outlier scores greater than 2.472, the auditee is still substantially different from the typical firm in this industry. Furthermore, the client occupies the lower end of the distribution, suggesting the organization has an unfavorable relative standing. In particular, its current ratio, long-term debt-to-equity ratio, and ROA values are all well below the associated industry benchmarks. Of particular concern, ROA is highly negative, demonstrating the company suffered a relatively high net loss during 2013. If the auditee is mature and well-established, this is a significant issue. Conversely, the client's inventory turnover ratio is nearly four times larger than the median firm, and this in isolation might be viewed in a positive light. Also, it could at least partially explain the abnormally low current ratio. However, when taken in a multivariate context, the company's performance is clearly substandard and suggestive of substantial risk. This would have a significant impact on risk assessment outcomes and subsequent formulation of the audit plan. By contrast, the object having the lowest outlier score is quite comparable to the median firm, indicating it is generally operating in alignment with what would have been expected of firms in this industry during 2013.

It should now be more apparent the outlier detection method proposed, developed, and implemented in this section is applicable to a variety of accounting and auditing activities. A primary pre-condition for its usage is that the associated data is able to be meaningfully represented in numeric terms. Given that the program code is provided in Appendix C, it can be readily adopted, adapted, and/or translated, thus facilitating ease of use. Collectively, it is hoped that this paper and the information provided within it will help to propel the accounting profession forward relative to the adoption of technology and automation in conducting future accounting and auditing tasks in a more productive manner.

CHAPTER 5: CLUSTERING AND OUTLIER DETECTION APPLICATIONS AND USAGE DISCUSSION

5.1 Introduction

To finalize this paper, procedures and approaches documented in the essays thus far are considered and synthesized to produce a seamless set of applications capable of performing clustering and outlier detection processes. The R code for this is contained in Appendices D and E. The program in appendix D is most useful on smaller data sets, generally accommodating up to about 20,000 records. Conversely, the application in Appendix E can ostensibly be used on both smaller and larger data sets. At this point, the programs may be viewed as initial prototypes offering proofs of concept relative to feasibility of automating clustering and outlier detection.

5.2 Background and Application Usage Information

Both programs operate with some assumptions relative to the data to be analyzed. Initially, it is assumed the first column of data is reserved for a unique identifier, and all subsequent columns are to be processed by the application. Also, the data to be evaluated must be represented on a truly numeric basis. For example, if an attribute to be clustered is of an interval or ratio nature, then it is already in a satisfactory state and can be used as is. In more technical terms, interval and ratio data both demonstrate the properties of addition and subtraction (Tan et al., 2006), and thus are truly numeric representations.

Conversely, categorical and ordinal data do not possess the properties of addition and subtraction, and are therefore not truly numeric. For example, if we consider two account numbers in a general ledger, 1000 and 2000, it is not sensible to speak of account number 2000 as being the sum of account number 1000 plus account number 1000. In fact, this is categorical data, and needs to be modified before use. One method for achieving this is asymmetric binarization (Tan et al., 2006). This entails constructing a unique binary string for each distinct value of a target dimension. For instance, if there are eight distinct account numbers in a data set to be analyzed, then a binary string of length eight is required to uniquely portray the attribute, thus effectively creating a new set of eight dimensions from the original account number variable. The first unique account number might be binarized as 10000000, the second as 01000000, the third as 00100000, and so on until each distinct value for account number is converted to a unique binary string.

In the case of ordinal attributes, data is able to be ranked or ordered (Tan et al., 2006), but could require modification prior to use. For example, academic grades A to E are considered ordinal data because they can be ranked in terms of favorability. More specifically, an "A" is better than "B", a "B" is superior to an "C", and so forth. One simple approach for transforming ordinal data entails establishing numbers that "mimic" the ordinal rankings in some logical manner. For instance, the letter grades A to E might be converted to the numbers 5 to 1, respectively. One issue with this type of transformation scheme is the inherent assumption that distance between any two adjacent ordinal values is identical (e.g., that distance between "A" and "B" is the same as that from "D" to "E"), and this will not always be true. Nevertheless, this form of ordinal transformation is discussed in the literature and done in practice (Tan et al., 2006). Prior to running the programs in Appendix D or E, the data set must be structured so that all but column 1 contains data to be analyzed and that all dimensions to be evaluated are represented in meaningful, numeric terms. As a final caveat, one should only include those attributes believed to be truly relevant for the clustering task.

The programs in appendices D and E also contain user specified parameters, although default settings are used where feasible to mitigate complexity. For example, simulation routines are configured to explore models having between 3 and 15 clusters, inclusive, but the user is able to adjust the range if desired. On the other hand, certain line items must be at least initially set by the user, such as path name for the data set to be imported for analysis and path names of destinations for processed information.

In the first program, some key parameters must be entered by the user. First, line 19 indicates the path where the data set to be analyzed resides. Incidentally, all path names are enclosed in quotes. Second, line 398 specifies location where the initial clustering model output file should be sent. Last, lines 452 to 604 indicate the destination path for sending final output files, which are segregated by cluster and contain distance information resulting from outlier detection. While there are several line items relative to output path settings, it is conceivable these would only need to be configured once by a given user. On the other hand, the data import path (i.e., line 19) might need to be modified frequently. This program also contains some parameters with default settings. Specifically, "mink" and "maxk" are pre-set at 3 and 15, respectively. Nevertheless, they are located on lines 95 and 96, and can be manipulated if desired. However, a condition is that "mink" cannot be set higher than "maxk".

The second application also requires the user to enter path information. In particular, the import path is specified on line 24, and the output path line items are on lines 178, 373 to 525, and 713 to 865. Once again, the import path command (i.e., line 23) might be expected to change often, while the output paths could be designed so as to remain substantially constant. In addtion, there are some optional settings to consider. First, a sample size can be selected. This parameter appears on line 104 (i.e., "sample" variable), and is set at 40,000 by default. Second, "minclust" and "maxclust" exist on lines 126 and 127, and they specify the range for number of clusters to be used in the simulation routines. As with the previous program, these are also set at 3 and 15 by default, and the user can adjust them if necessary.

Basically, the two programs may be executed in an identical manner. More specifically, assuming that parameters are set as desired and that all necessary R packages have been imported/added to the RStudio library, the user may simply open an instance of RStudio, left-click the "Compile Notebook" icon located near the top of the upper-left window, select the "Compile" button, and await results. When completed, the user will be presented with an electronic HTML document containing the R program code along with associated execution notations, tables, and visualizations. Incidentally, if a PDF or MS Word version of this document is preferred, this may be obtained by selecting the desired option from the drop-down list in the "Compile Notebook From Rscript" window prior to selecting the "Compile" button. Furthermore, processed output files in .csv format will be sent to the output path locations as specified by the user, and these can be used in performing addtional analyses in programs such as Excel, Tableau, Clickview, etc. as desired.

Two limitation issues in R are worth mentioning. First, all versions of RStudio are not compatible with all existing R packages. Consequently, a new version of RStudio may not recognize all packages executable in earlier editions. Given this, it cannot be guaranteed that the programs in Appendices D and E will immediately operate as intended on any version of R[5]. However, it should be the case that, at most, only minor modifications are necessary.

Second, R relies on random access memory (RAM) when processing data. In some cases, the amount of required RAM can be substantial, and, in such contexts a server environment is highly recommended. Also, even if a satisfactory amount of RAM is available, R might not be able to access it without additional configuration. For example, if attempting to maximize memory access in a system with 64GB of RAM, right-click the RStudio executable file, left-click "properties", and, in the "Target" field following the end of the path name, enter " --max-mem-size=64G" followed by " --max-vsize=64G". Similarly, if the system

[5] These programs were written using 64-bit R in conjunction with RStudio version 3.0.2. 32-bit versions of R have significant memory limitations and are not recommend here.

has 96GB of RAM, enter essentially the same information, except replace "64" with "96". Ensuring that sufficient RAM is available and is able to accessed is one key to improving the R experience.

CHAPTER 6: LIMITATIONS, FUTURE RESEARCH, AND CONCLUSIONS

6.1 Limitations and Future Research

As with most studies, this paper has limitations and presents opportunities for future research. First, the data set is viewed as somewhat lacking in terms of available dimensions. For instance, while information such as customer profit and revenue are presumably captured by the banking institution, the data set contains missing values for these items. Additional information could offer incremental value in comprehensively profiling the customer base. Therefore, if more exhaustive data could be obtained, it would be worth analyzing. While I believe the current results are meaningful, perhaps they could be enhanced via inclusion of other pertinent features.

Second, the formula implemented in establishing overall scores for clusters in chapter 3 is rudimentary and could be targeted for reengineering. For example, only three of the four dimensions were given weightings based upon the findings of prior research. The remaining feature was simply assigned influence based upon the available amount remaining following establishment of other weights. Furthermore, a certain amount of extrapolation is required to make the sum of all weights equal one. As such, a degree of subjectivity is inherent in this task. Also, each score is computed simply as the sum of the products of each dimension value and associated weighting. It would be instructive to determine the specific importance level of all variables and assign weights in a fully objective manner. It could also be that the scoring formula itself would benefit from further enhancements.

Third, data mining often involves analysis of extremely large data sets. In this context, the notion of statistical significance between groups in a large population is problematic and perhaps even meaningless. More specifically, the tendency in research is that, as the number of observations increases, the probability of finding significant outcomes also increases. Consequently, it is not surprising that clustering results will often be found to be significant if testing is conducted on the full population. Given this as well as the perceived utility of employing tests of statistical significance in this domain, the ability to extract and test representative samples is highly desirable. However, an immediate empirical question concerns what a preferred sample size is in the clustering context, particularly in the case of unbalanced groups. Future research might explore this issue in an effort to establish parameters concerning sampling from populations in cases of conducting statistical tests of clustering models.

Last, while user-defined weight parameters are incorporated for use in conducting outlier detection, intuition and/or information is lacking concerning what each weight should be for a each distance/dissimilarity measure. Consequently, equal weight was attached to each metric in this study. It is likely that appropriate weight settings would be a function of the data being analyzed, and perhaps heuristics could be developed through subsequent research. For example, because Mahalanobis Distance has been found to be particularly useful in multivariate outlier detection, the weight on this measure might be expanded as the number of dimensions increases. Conversely, it could be that Euclidean Distance becomes less suitable as the number of dimensions grows, suggesting its weight should be reduced in such cases. Also, as the density of data decreases, this might dictate that greater (lesser) weight be placed on the similarity (distance) measures. Determining specifics about what actual weightings should be in given situations produces a set of empirical questions justifying further study. At least partially because this is a newly proposed method in need of validation, it would be interesting to assess the approach on a variety of data specifically seeded with outliers.

One obvious outcome for this project relates to providing information for creating formal and objectively determined customer profiles that can be translated into a set of customer relationship management policies. Because each segment relates to a distinct client type in terms of creditworthiness, each group should be managed differently. For example, the most creditworthy customers might be viewed as an irreplaceable and extremely valuable asset. In this case, extensive efforts should be made in the areas of customer satisfaction and retention initiatives relative to individuals fitting this profile. Furthermore, this group should be rewarded for their responsible handling of credit as well as the contributions they make toward the long-run success of the institution. For example, among other things, they should be eligible for the highest credit card fee discounts.

The main purpose of this study entails initial formulation of automated clustering and outlier detection programs. It is hoped they can be used or adapted for use to provide valuable information to auditors and accountants in fulfilling their responsibilities to relevant stakeholder groups. In very specific terms, the programs are designed for use by auditors in the fraud discovery process. However, they should have utility in a variety of settings. Throughout this paper, emphasis was placed upon substantial automation of associated tasks. Upon completion of initial work, the program code was then reformulated, synthesized, and logically connected such that two separate applications eventually resulted. In fact, the initial program proved to be primarily useful for relatively small data sets (see Appendix D). Consequently, a second application was created to accommodate larger amounts of data (see Appendix E). My belief is that, both programs demonstrate proofs of concept that clustering and outlier detection are indeed automatable. Moving forward, perhaps additional program modifications and/or enhancements are worth considering. Furthermore, in striving for efficiency gains, the R-code might be translated into alternative, suitable languages (e.g., C, Python, Java, etc.). Certainly, many opportunities exist for building on the work conducted and artifacts developed in this paper. Ultimately, if the programs are found to be beneficial, cost-effective, and user-friendly, the likelihood of adoption by the accounting profession should be greatly enhanced.

Historically, data mining initiatives can be very cumbersome, time-consuming, and complicated. For instance, a significant amount of expertise is generally needed in the areas of data pre-processing, algorithm selection, and model evaluation. Furthermore, resistance to change can interact with machine learning complexities so as to inhibit adoption by the profession. The methods presented in this paper are highly automated and, to the extent feasible, made intuitive. Also, associated code is included in the appendices so as to encourage the accounting profession and others to move forward in experimenting with and using data mining technologies in efforts to improve productivity.

While this paper demonstrates a set of fully automated solutions, the programs could be adapted and enhanced to accommodate the use of other clustering algorithms and outlier detection routines, making it ostensibly applicable to a wider variety of data mining initiatives. For example, in the second and third chapters, five algorithms were considered for primary model building. However, the approach could theoretically accommodate other methods as well. For instance, one might seek to use DBScan in addition to the other algorithms. The outlier detection process described in chapter four may also be refined for use in other contexts. Specifically, because the model relies upon computation of well-established distance and similarity measures for multivariate use, it is applicable to any outlier detection context, as long as the involved data is able to be adequately represented in numeric terms. However, other recognized proximity measures could ostensibly be explored in addition to or as substitutes for the metrics used in this paper.

6.2 Conclusion

Customers are a key resource for any organization, and are instrumental in determining the extent to which business success is able to be achieved and maintained. Customer profiling can be perceived an important initial building block for understanding the customer base. When an entity is aware of its mix of customer types, it is better equipped to tailor business strategies, initiatives, and activities, and thus improve current and future company performance.

Organizational fraud is a growing problem for which solutions are desperately needed. Fortunately, data mining techniques can readily assist in addressing this significant problem. Through substantial automation of data mining tasks, auditors can more easily employ clustering and outlier detection in fulfilling their duties in the evolving real-time global economy. In so doing, they will be better positioned to add value to and advance the accounting profession both now and in the future.

REFERENCES

Abdul-Nazeer, K.A., and Sebastian, M.P. (2009). Improving the Accuracy and Efficiency of the K-means Clustering Algorithm. *Proceedings of the World Congress on Engineering. 1.* London, U.K.: WCE.

ACFE. (2010). *Report to the Nations on Occupational Fraud and Abuse: 2010 Global Fraud Study.* Retrieved from www.acfe.com: http://www.acfe.com/uploadedFiles/ACFE_Website/Content/documents/rttn-2010.pdf

ACFE. (2014). *Report to the Nations on Occupational Fraud and Abuse: 2014 Global Fraud Study.* Retrieved from www.acfe.com: http://www.acfe.com/rttn/docs/2014-report-to-nations.pdf

Acuna E., and Rodriguez, C. (2011). *A Meta Analysis Study of Outlier Detection Methods in Classification.*

Alpaydin, E. (2010). *Introduction to Machine Learning* (2nd ed.). Massachusetts: Massachusetts Institute of Technology.

Auditing Standards Board (ASB). (2002b). *AU Section 316: Consideration of Fraud in a Financial Statement Audit.* American Instititute of Certified Public Accountants (AICPA).

Auditing Standards Board (ASB). (2012). *AU-C Section 240: Consideration of Fraud in a Financial Statement Audit.* American Institute of Certified Public Accountants (AICPA).

Auditng Standards Board (ASB). (2002a). *Statement on Auditing Standards No. 99: Consideration of Fraud in a Financial Statement Audit.* American Institute of Certified Public Accountants (AICPA).

Barnett V., and Lewis, T. (1994). *Outliers in Statistical Data.* John Wiley.

Chandola, V., Banerjee, A., and Kumar, V. (2009). Anomaly Detection: A Survey. *ACM Computing Surveys, 41*(3), 15-58.

Consumer Financial Protection Bureau (CFPB). (2012). *Key Dimensions and Processes in the U.S. Credit Reporting System.* Retrieved from consumerfinance.gov: http://files.consumerfinance.gov/f/201212_cfpb_credit-reporting-white-paper.pdf

Credit Score Decoder (CSD). (2013). Retrieved from creditscoredecoder.com: http://creditscoredecoder.com/credit-score-calculated/

Davis, F. D. (1985). Technology Acceptance Model. *Proceedings of 2002 CSCW Conference*, (pp. 107-114).

Davis, F. D. (1989). Perceived Usefulness, Perceived Ease of Use, and User Acceptance of Information Technology. *MIS Quarterly, 13*(3), 318-339.

Deng, Q., and Mei, G. (2009). Combining Self-Organizing Map and K-means Clustering for Detecting Fraudulent Financial Statements. *IEEE International Conference on Granular Computing, 2009*, (pp. 126-131).

Ekelund, R. B., Ressler, R. W., and Tollison, R. D. (2006). *Economics: Private Markets and Public Choice* (7th ed.). Boston, MA: Pearson Education, Inc.

Fayyad, U.M., Piatetsky-Shapiro, G., Smyth, P., and Uthurusamy, R. (1996). *Advances in Knowledge Discovery and Data Mining.* AAAI Press/MIT Press.

Garla, S., Chakraborty, G., and Gaeth, G. (2012). *Comparison of K-means, Normal Mixtures, and Probabilistic D-Clustering for B2B Segmentation Using Customers' Perceptions.* SAS Global.

Ghani, R., and Kumar, M. (2011). Interactive Learning for Efficiently Detecting Errors in Insurance Claims. *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, (pp. 325-333). New York, NY, USA.

Glancy, F., and Yadav, S. (2011). A Computational Model for Financial Reporting Fraud Detection. *Decision Support Systems, 50*(3), 595-601.

Green, P., and Rao, V. (1969). A Note on Proximity Measures and Cluster Analysis. *Journal of Marketing Research, 6*(August), 359-364.

Han, J., and Kamber, M. (2001). *Data Mining: Concepts and Techniques.* San Francisco, CA: Morgan-Kaufmann Publishers.

Hao, M., Dayal, U., Sharma, R., Keim, A., and Janetzko, H. (2010). *Visual Analytics of Large Multi-Dimensional Data Using Variable Binned Scatter Plots.* Bibliothek der Universitat Konstanz.

Harding, W. (2006). Accounting Today. *Data Mining is Crucial for Detecting Fraud in Audits*, pp. 22-24.

Hawkins, D. (1980). *Identification of Outliers.* Chapman and Hall.

Holgersson, H.E.T., and Karlsson, P.S. (2012). Three Estimators of the Mahalanobis Distance in High-Dimensional Data. *Journal of Applied Statistics, 39*(12), 2713-2720.

Huang, J.Z., Ng, M.K., Rong, H., and Li, Z. (2005). Automated Variable Weighting in K-means Type Clustering. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *27, 5*, pp. 657-668.

Issa, H. (2013). *Exceptional Exceptions.* Doctoral Dissertation, Rutgers University-Graduate School-Newark.

Issa, H., and Kogan, A. (2014). A Predictive Ordered Logistic Regression Model as a Tool for Quality Review of Control Risk Assessments. *Journal of Information Systems, 28*(2), 209-229.

Issa, H., Kogan, A., and Brown-Liburd, H. (2015). Identifying and Prioritizing Irregularities Using a Rule-Based Model with a Weighting System Derived from Experts' Knowledge. *Working Paper*.

Jurek, A., and Zakrzewska, D. (2008). Improving Naive Bayes Models of Insurance Risk by Unsupervised Classification. *International Multiconference on Computer Science and Information Technology (IMCSIT)*, (pp. 137-144).

Jyotindra, N., and Ashok, R. (2011). A Data Mining with Hybrid Approach Based Transaction Risk Score Generation Model (TRSGM) for Fraud Detection of Online Financial Transactions. *International Journal of Computer Applications, 16*(1), 18-25.

Kanter, R. M. (1983). *The Change Masters: Corporate Entrepreneurs at Work.* New York: Simon and Schuster.

Khac, N., and Kechadi, M. (2010). Application of Data Mining for Anti-Money Laundering Detection: A Case Study. *2010 IEEE International Conference on Data Mining Workshops (ICDMW)*, (pp. 577-584).

Khandani, A.E., Kim, A.J., and Lo, A.W. (2010). *Consumer Credit Risk Models via Machine Learning Algorithms.* draft of May 9, 2010.

Liu, R., Qian, X., Mao, S., and Zhu, S. (2011). Research on Anti-Money Laundering Based on Core Decision Tree Algorithm. *Control and Decision Conference (CCDC)*, (pp. 4322-4325).

Mahendiran, A., Saravanan, N., Venkata-Subramanian, N., and Sairam, N. (2012). Implementation of K-means Clustering in Cloud Computing Environment. *Research Journal of Applied Sciences, Engineering, and Technology, 4*(10), 1391-1394.

Michigan State University Federal Credit Union (MSUFCU). (2015). What is a Credit Score?

Mo, J., Kiang, M.Y., Zou, P., and Li, Y. (2010). A Two-Stage Clustering Approach for Multi-Region Segmentation. *Expert Systems with Applications, 37*, 7120-7131.

Morgan, K. (2011). Retrieved April 23, 2014, from voices.yahoo.com: http://voices.yahoo.com/how-improve-credit-score-8656533.html

Panigrahi, S., Kundu, A., Sural, S., and Majumdar, A. (2009). Credit Card Fraud Detection: A Fusion Approach Using Dempster-Shafer Theory and Bayesian Learning. *Information Fusion, 10*(4), 354-363.

Scandizzo, S. (2005). Risk Mapping and Key Risk Indicators in Operational Risk Management. *Economic Notes by Banca Monte dei Paschi di Siena SpA, 34*(2), 231-256.

Shalabi, L.A., Shaaban, Z., and Kassabeh, B. (2006). Data Mining: A Preprocessing Engine. *Journal of Computer Science, 2*(9), 735-739.

Shuai, L., Lai, H., Xu, C., and Zhou, Z. (2013). The Discrimination Method and Empirical Research of Individual Credit Risk Based on Bilateral Clustering. *Modern Economy, 4*, 461-465.

Singleton, R. A., and Straits, B. C. (1999). *Approaches to Social Research* (3rd ed.). New York: Oxford University Press.

Starkweather, J. (2013). *Multivariate Outlier Detection with Mahalanobis Distance.* Retrieved October 2014, from http://www.unt.edu/rss/class/Jon/Benchmarks/Moutlier_JDS_July2013.pdf

Tam, K. Y., and Ho, S. Y. (2007). A Smart Card Based Internet Micropayment Infrastructure: Technical Development and User Adoption. *Journal of Organizational Computing and Electronic Commerce, 17*(2), 145-173.

Tan, P., Steinbach, M., and Kumar, V. (2006). *Introduction to Data Mining.* Boston, MA: Pearson Education, Inc.

Tasoulis, D., Adams, N., Weston, and Hand, D. (2008). Mining Information from Plastic Card Transaction Streams. *Proceedings in Computational Statistics: 18th Symposium*, (pp. 315-322).

Thiprungsri, S., and Vasarhelyi, M. (2011). Cluster Analysis for Anomaly Detection in Accounting Data: An Audit Approach. *The International Journal of Digital Accounting Research, 11*.

Tukey, J. (1977). *Exploratory Data Analysis.* Addison-Wesley.

Van de Ven, A. H. (1986). Central Problems in the Management of Innovation. *Management Science, 32*, 590-607.

Virdhagriswaran, S., and Dakin, G. (2006). Camouflaged Fraud Detection in Domains with Complex Relationships. *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, (pp. 941-947).

Wang, W., and Benbasat, I. (2005). Trust in and Adoption of Online Recommendation Agents. *Journal of the Association for Information Systems, 6*(3), 72-101.

Wang, X., and Danyue, L. (2010). Hybrid Outlier Mining Algorithm-Based Evaluation of Client Moral Risk in Insurance Company. *2010, The 2nd IEEE International Conference on Information Management and Engineering (ICIME)*, (pp. 585-589).

Whittington, O. R., and Pany, K. (2008). *Principles of Auditing & Other Assurance Services* (16th ed.). New York, NY: McGraw Hill/Irwin.

Wu, J., Xiong, H., and Chen, J. (2010). COG: Local Decomposition for Rare Class Analysis. *Data Mining and Knowledge Discovery, 20*(2), 191-220.

Zhang, S., Chen, F., Wu, X., and Zhang, C. (2006). Identifying Bridging Rules Between Conceptual Clusters. *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, (pp. 815-820).

Zhang, Z., Salerno, J., and Yu, P. (2003). Applying Data Mining in Investigating Money Laundering Crimes. *Proceedings of the 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, (pp. 747-752).

Zimek, A., Campello, R., and Sander, J. (2014). *Ensembles for Unsupervised Outlier Detection: Challenges and Research Questions.* SIGKDD Explorations.

APPENDIX A: RSCRIPT - CHAPTER 2 CLUSTERING

```
#Chapter 1 - Stage One Cluster Analysis and Testing
#Stage One - Part I
#Data preprocessing
#Load necessary packages
library(cluster)
library(mclust)
library(stats)
library(plyr)
library(Rcmdr)
library(gplots)
library(scatterplot3d)
library(chemometrics)
library(fields)
library(fastcluster)
library(EMCluster)

#Increase memory limit
memory.size(max=TRUE)
memory.limit(size=999999)

# Read main data set CustomerDataNormalizedCSV  in R and read number of
rows
CustomerData <- read.csv("K:/Itau/Credit
Card/Paul/Dissertation/PrimaryData/CreditCardClustering1.csv")
View(CustomerData)
y <- nrow(CustomerData)

#Create a data frame of CustomerData
x1 <- CustomerData[, 1:5]
View(x1)

#-----------------------------------------------------------------------------------------
#Normalize four dimensions to be clustered
#Assumes dimensions to be normalized are in columns 2, 3, 4, and 5 of data
#Normalize AccountAge column
aa1 <- x1[[2]]
maxaa <- max(aa1)
minaa <- min(aa1)
aan <- vector(mode="numeric", length=y)
i<-1
while(i<y+1) {
  aan[i] <- ((aa1[i] - minaa)/(maxaa-minaa))
  i <- i+1
}

#Check/verify max & min normalized AccountAge column values (max=1, min=0)
View(max(aan))
View(min(aan))

#Add normalized AccountAge column to data frame x1
x1["AccountAgeN"] <- aan

#Normalize CreditLimit column
cl1 <- x1[[3]]
maxcl <- max(cl1)
mincl <- min(cl1)
cln <- vector(mode="numeric", length=y)
i<-1
while(i<y+1) {
  cln[i] <- ((cl1[i] - mincl)/(maxcl-mincl))
  i <- i+1
}

#Check/verify max & min normalized CreditLmit column values (max=1, min=0)
View(max(cln))
View(min(cln))

#Add normalized CreditLimit column to data frame x1
x1["CreditLimitN"] <- cln

#Normalize AdditionalAssets column
adda1 <- x1[[4]]
maxadda <- max(adda1)
minadda <- min(adda1)
addan <- vector(mode="numeric", length=y)
i<-1
while(i<y+1) {
  addan[i] <- ((adda1[i] - minadda)/(maxadda-minadda))
  i <- i+1
}

#Check/verify max & min normalized AdditionalAssets column values (max=1,
min=0)
View(max(addan))
View(min(addan))

#Add normalized AdditionalAssets column to data frame x1
x1["AdditionalAssetsN"] <- addan
```

```
#Normalize LatePayments column
lp1 <- x1[[5]]
maxlp <- max(lp1)
minlp <- min(lp1)
lpn <- vector(mode="numeric", length=y)
i<-1
while(i<y+1) {
  lpn[i] <- ((lp1[i] - minlp)/(maxlp-minlp))
  i <- i+1
}

#Check/verify max & min normalized LatePayments column values (max=1,
min=0)
View(max(lpn))
View(min(lpn))

#Add normalized LatePayments column to data frame x1
x1["LatePaymentsN"] <- lpn
View(x1)
#-------------------------------------------------------------------------------------------

#Create data frame of normalized original variables for performing principal
components analysis
CustomerData1 <- data.frame(x1[ , c("AccountAgeN", "CreditLimitN",
"AdditionalAssetsN", "LatePaymentsN")])
View(CustomerData1)

#Perform principal component analysis and show associated information
trans <- prcomp(CustomerData1, center=TRUE, scale.=TRUE)
print(trans)
summary(trans)
plot(trans, typ="l", col="2", lwd="2")

PC = predict(trans, CustomerData1)
print(PC[1:10, ])
View(PC)
plot(PC, type="b", col="2")

#Add all principal components to x1, thus creating x1.pc
x1.pc <- cbind(x1, PC)
View(x1.pc)

#Normalize four principal components to be clustered
#Assumes dimensions to be normalized are in columns 10, 11, 12, and 13 of
data
```

```
#Normalize PC1 column
pc1 <- x1.pc[[10]]
maxpc1 <- max(pc1)
minpc1 <- min(pc1)
pc1n <- vector(mode="numeric", length=y)
i<-1
while(i<y+1) {
  pc1n[i] <- ((pc1[i] - minpc1)/(maxpc1-minpc1))
  i <- i+1
}

#Check/verify max & min normalized PC1 column values (max=1, min=0)
View(max(pc1n))
View(min(pc1n))

#Add normalized PC1 column to data frame x1.pc
x1.pc["PC1N"] <- pc1n

#Normalize PC2 column
pc2 <- x1.pc[[11]]
maxpc2 <- max(pc2)
minpc2 <- min(pc2)
pc2n <- vector(mode="numeric", length=y)
i<-1
while(i<y+1) {
  pc2n[i] <- ((pc2[i] - minpc2)/(maxpc2-minpc2))
  i <- i+1
}

#Check/verify max & min normalized PC2 column values (max=1, min=0)
View(max(pc2n))
View(min(pc2n))

#Add normalized PC2 column to data frame x1.pc
x1.pc["PC2N"] <- pc2n

#Normalize PC3 column
pc3 <- x1.pc[[12]]
maxpc3 <- max(pc3)
minpc3 <- min(pc3)
pc3n <- vector(mode="numeric", length=y)
i<-1
while(i<y+1) {
  pc3n[i] <- ((pc3[i] - minpc3)/(maxpc3-minpc3))
  i <- i+1
}
```

```
#Check/verify max & min normalized PC3 column values (max=1, min=0)
View(max(pc3n))
View(min(pc3n))

#Add normalized PC3 column to data frame x1.pc
x1.pc["PC3N"] <- pc3n

#Normalize PC4 column
pc4 <- x1.pc[[13]]
maxpc4 <- max(pc4)
minpc4 <- min(pc4)
pc4n <- vector(mode="numeric", length=y)
i<-1
while(i<y+1) {
  pc4n[i] <- ((pc4[i] - minpc4)/(maxpc4-minpc4))
  i <- i+1
}

#Check/verify max & min normalized PC4 column values (max=1, min=0)
View(max(pc4n))
View(min(pc4n))

#Add normalized PC4 column to data frame x1.pc
x1.pc["PC4N"] <- pc4n
View(x1.pc)

#Get data frame of x1.pc for clustering
CustomerDataPC <- data.frame(x1.pc [ , c("PC1N", "PC2N", "PC3N")])
View(CustomerDataPC)

#Export x1.pc and CustomerDataPC
write.csv(x1.pc, "K:/Itau/Credit
Card/Paul/Dissertation/ClusteringStageOne/x1.pc.csv", row.names=FALSE)
write.csv(CustomerDataPC, "K:/Itau/Credit
Card/Paul/Dissertation/ClusteringStageOne/CustomerDataPC.csv",
row.names=FALSE)


#Stage one - Part II
#Conduct simulation routines to locate best stage one model (silhouette
coefficient analysis)
#Import data for analysis (if not currently loaded)
x1.pc <- read.csv("K:/Itau/Credit
Card/Paul/Dissertation/ClusteringStageOne/x1.pc.csv")
View(x1.pc)
```

```
#Get a random sample of 40,000 records from population of x1.pc (set s =
sample size)
#Necessary because silhouette calculation will not allow use of long vectors (>
2^31)
#Also necessary because processing is extremely inefficient on PAM
#Investigating the use of "bigmemory.sri" to mitigte this problem (unresolved as
of 7/27/15)
s <- 40000
p <- s/nrow(x1.pc)
View(p)
x1.pcs <- x1.pc[sample(nrow(x1.pc),replace=F,size=p*nrow(x1.pc)),]
row.names(x1.pcs)<-NULL
View(x1.pcs)

#Create a data frame CustomerDataPCS of 3 top PCs from x1.pcs for computing
silhouette coeffients
c <- ncol(x1.pcs)
c1 <- c-3
c2 <- c-1
CustomerDataPCS <- data.frame(x1.pcs[ , c1:c2])
View(CustomerDataPCS)


#Execute Kmeans clustering model performance sequence
#Use loop structure for initial Kmeans model building process
nclust=12
x=matrix(nrow=nclust-1, ncol=2)
d <- dist(CustomerDataPCS, "euclidean")
y <- 1
c <- 2
for (c in c:nclust) {
   fit <- kmeans(CustomerDataPCS, c, iter.max=500, nstart=10,
algorithm="Lloyd")
   sil <- silhouette(fit$cluster, d)
   avgsil <- summary(sil, FUN=mean) $avg.width
   x[y, ] <- c(c, avgsil)
   y<-y+1
}
View(x)
max(x[ ,2])

#Create data frame z, Sort in descending order by Silhouette Coefficient
z <- data.frame(x)
names(z)[1] <- "Clusters"
names(z)[2] <- "Silhouette"
```

```
View(z)

s <- arrange(z, desc(Silhouette))
View(s)

#Export processed data
write.csv(s, "K:/Itau/Credit
Card/Paul/Dissertation/ClusteringStageOne/SilhouetteKmeans.csv")


#Execute PAM clustering model performance sequence
#Use loop structure for initial model building process
nclust= 12
x=matrix(nrow=nclust-1, ncol=2)
d <- dist(CustomerDataPCS, method = "euclidean")
y <- 1
c <- 2
for (c in c:nclust) {
  fit <- pam(CustomerDataPCS, c)
  sil <- silhouette(fit$clustering, d)
  avgsil <- summary(sil, FUN=mean) $avg.width
  x[y, ] <- c(c, avgsil)
  y<-y+1
}
View(x)
max(x[ ,2])

#Create data frame z, Sort in descending order by Silhouette Coefficient
z <- data.frame(x)
names(z)[1] <- "Clusters"
names(z)[2] <- "Silhouette"
View(z)

s <- arrange(z, desc(Silhouette))
View(s)

#Export processed data
write.csv(s, "K:/Itau/Credit
Card/Paul/Dissertation/ClusteringStageOne/SilhouettePAM.csv")


#Execute Complete-Link Hierarchical clustering model performance sequence
#Use loop structure for initial model building process
nclust= 12
x=matrix(nrow=nclust-1, ncol=2)
d <- dist(CustomerDataPCS, method = "euclidean")
```

```
y <- 1
c <- 2
for (c in c:nclust) {
  fit2 <- hclust(d, method = "complete")
  memb <- cutree(fit2, k=c)
  fit$clusters <- assignCluster(CustomerDataPCS, x1.pcs, memb)
  fit$cluster <- as.integer(fit$clusters)
  sil <- silhouette(fit$cluster, d)
  avgsil <- summary(sil, FUN=mean) $avg.width
  x[y, ] <- c(c, avgsil)
  y<-y+1
}
View(x)
max(x[ ,2])

#Create data frame z, Sort in descending order by Silhouette Coefficient
z <- data.frame(x)
names(z)[1] <- "Clusters"
names(z)[2] <- "Silhouette"
View(z)

s <- arrange(z, desc(Silhouette))
View(s)

#Export processed data
write.csv(s, "K:/Itau/Credit
Card/Paul/Dissertation/ClusteringStageOne/SilhouetteComplete.csv")


#Execute Ward's Method Hierarchical clustering model performance sequence
#Use loop structure for initial model building process
nclust= 12
x=matrix(nrow=nclust-1, ncol=2)
d <- dist(CustomerDataPCS, method = "euclidean")
y <- 1
c <- 2
for (c in c:nclust) {
  fit2 <- hclust(d, method = "ward.D2")
  memb <- cutree(fit2, k=c)
  fit$clusters <- assignCluster(CustomerDataPCS, x1.pcs, memb)
  fit$cluster <- as.integer(fit$clusters)
  sil <- silhouette(fit$cluster, d)
  avgsil <- summary(sil, FUN=mean) $avg.width
  x[y, ] <- c(c, avgsil)
  y<-y+1
}
```

```
View(x)
max(x[ ,2])

#Create data frame z, Sort in descending order by Silhouette Coefficient
z <- data.frame(x)
names(z)[1] <- "Clusters"
names(z)[2] <- "Silhouette"
View(z)

s <- arrange(z, desc(Silhouette))
View(s)

#Export processed data
write.csv(s, "K:/Itau/Credit
Card/Paul/Dissertation/ClusteringStageOne/SilhouetteWard.csv")


#Execute Expectation Maximization clustering model performance sequence
#Use loop structure for initial Kmeans model building process
nclust=12
x=matrix(nrow=nclust-1, ncol=2)
d <- dist(CustomerDataPCS, "euclidean")
y <- 1
c <- 2
for (c in c:nclust) {
  fit <- init.EM(CustomerDataPCS, nclass = c, EMC = .EMC.Rnd, stable.solution
= TRUE, min.n = 1, min.n.iter = 10, method = "Rnd.EM")
  sil <- silhouette(fit$class, d)
  avgsil <- summary(sil, FUN=mean) $avg.width
  x[y, ] <- c(c, avgsil)
  y<-y+1
}
View(x)
max(x[ ,2])

#Create data frame z, Sort in descending order by Silhouette Coefficient
z <- data.frame(x)
names(z)[1] <- "Clusters"
names(z)[2] <- "Silhouette"
View(z)

s <- arrange(z, desc(Silhouette))
View(s)

#Export processed data
```

```
write.csv(s, "K:/Itau/Credit
Card/Paul/Dissertation/ClusteringStageOne/SilhouetteEM.csv")


#Stage One - Part III
#Build final stage one model
#Read data sets for clustering in R (if necessary)
CustomerDataPC <- read.csv("K:/Itau/Credit
Card/Paul/Dissertation/ClusteringStageOne/CustomerDataPC.csv")
x1.pc <- read.csv("K:/Itau/Credit
Card/Paul/Dissertation/ClusteringStageOne/x1.pc.csv")
View(CustomerDataPC)
View(x1.pc)
r1 <- nrow(x1.pc)
c3 <- ncol(x1.pc)

#Get random sample of records for building training model
#Default sample size set at 50,000 (change if needed)
sample <- 50000
p1 <- sample/r1
x1.pcs <- x1.pc[sample(nrow(x1.pc),replace=F,size = p1*nrow(x1.pc)),]
row.names(x1.pcs) <- NULL
View(x1.pcs)

c4 <- c3-3
c5 <- c3-1
CustomerDataPCS <- data.frame(x1.pcs[ , c4:c5])
row.names(CustomerDataPCS)<-NULL
View(CustomerDataPCS)

#Build/evaluate stage one training model (Decision 07/27/2015: Build 3 cluster
Complete-Link model)
#Set c = number of clusters
c <- 3

#Compute distance matrix for training data
d1 <- dist(CustomerDataPCS, method="euclidean")

#Build training model
fit2 <- hclust(d1, method = "complete")
memb <- cutree(fit2, k=c)

#Append cluster assignment column
fit.clusters <- assignCluster(CustomerDataPCS, x1.pcs, memb)
fit.cluster <- as.integer(fit.clusters)
StageOneModel <- data.frame(x1.pcs, fit.cluster)
```

```
View(StageOneModel)

#3D Plots of top 3 principal components with cluster assignments depicted
p2 <- data.frame(CustomerDataPCS)
ddd <- data.frame(p2, StageOneModel[ , "fit.cluster"])
names(ddd)[1] <- "PC1"
names(ddd)[2] <- "PC2"
names(ddd)[3] <- "PC3"
names(ddd)[4] <- "Cluster"
scatterplot3d(ddd, col.axis = "green", col.grid = "green", main = "Principal
Components Scatterplot", angle=50, pch = 1)
scatterplot3d(ddd, col.axis = "green", col.grid = "green", main = "Principal
Components Scatterplot", angle=140, pch = 1)
scatterplot3d(ddd, col.axis = "green", col.grid = "green", main = "Principal
Components Scatterplot", angle=230, pch = 1)
scatterplot3d(ddd, col.axis = "green", col.grid = "green", main = "Principal
Components Scatterplot", angle=320, pch = 1)

#Generate data frames for subsequent analysis
x3 <- data.frame(CustomerDataPCS)
x4 <- data.frame(StageOneModel[ , 6:9])
x5 <- data.frame(StageOneModel[ , 2:5])

#Get cluster means for normalized pcs, normalized original variables, and
original variables
aggregate(x3,by=list(fit.cluster),FUN=mean)
aggregate(x4, by=list(fit.cluster), FUN=mean)
aggregate(x5, by=list(fit.cluster), FUN=mean)

#Get cluster medians for normalized pcs, normalized original variables, and
original variables
aggregate(x3,by=list(fit.cluster),FUN=median)
aggregate(x4, by=list(fit.cluster), FUN=median)
aggregate(x5, by=list(fit.cluster), FUN=median)


#Peform oUtlier detection via mahalanobis distance (training model)
#Set initial variables for subsequent usage
r2 <- nrow(StageOneModel)
c6 <- ncol(CustomerDataPCS)
b1 <- r2*.01
x6 <- max(StageOneModel[ ,"fit.cluster"])
p3 <- data.frame(CustomerDataPCS, fit.cluster)
p4 <- data.frame(arrange(p3, fit.cluster), row.names=NULL)
p5 <- data.frame(arrange(StageOneModel, fit.cluster), row.names=NULL)
```

```
#Separate data by cluster, compute medians, calculate mahalanobis distances,
and create outlier plots for each cluster
for (c in 1:x6) {
  nam1 <- paste("Cluster", c, sep = "")
  z <- assign(nam1, subset(p5, fit.cluster==c))
  z1 <- subset(p4, fit.cluster==c)
  med <- vector(mode="numeric", length=c6)
  for (cc in 1:c6) {
    med[cc] <- median(z1[ , cc])
  }
  z2 <- data.frame(z1[ ,1:c6])
  z3 <- cov(z2, use = "everything" )
  md <- mahalanobis(z2, center = med, cov = z3, inverted = TRUE)
  nam4 <- paste("Cluster", c, sep = "")
  z4 <- assign(nam4, data.frame(z, md, row.names=NULL))
  y2 <- nrow(z4)
  row.names(z4) <- 1:y2
  df.2 <- c(1:y2)
  if (y2 >= b1) {
    Out.999 = quantile(md, .999)
    plot(df.2, md, xlab="Object", ylab="Mahalanobis Distance",
main=c("Cluster",c,"Outlier Plot"))
    abline(h=Out.999, col="red", lwd=2)
    cat("The following rows in cluster",c,"data appear very
suspicious:",which(md>Out.999),"\n")
    qqplot(md, df.2, xlab="Mahalanobis Distance", ylab="Ranked Observation",
main=c("Cluster",c,"QQ Plot"))
    abline(v=Out.999, col="red", lwd=2)
  }
  if (y2 %in% c(3:b1)) {
    Out.999 = quantile(md, .999)
    plot(df.2, md, xlab="Object", ylab="Mahalanobis Distance",
main=c("Cluster",c,"Outlier Plot"))
    abline(h=Out.999, col="red", lwd=2)
    cat("The following rows in cluster",c,"data appear very
suspicious:",which(md>Out.999),"\n")
    qqplot(md, df.2, xlab="Mahalanobis Distance", ylab="Ranked Observation",
main=c("Cluster",c,"QQ Plot"))
    abline(v=Out.999, col="red", lwd=2)
    cat("Cluster",c,"has small relative membership and this warrants investigation
of all associated objects","\n")
  }
  if (y2 < 3) {
    cat("Cluster",c,"has an extremely small membership and this clearly warrants
investigation of all associated objects","\n")
  }
}

#Export processed information, segregated by cluster
write.csv(Cluster1, row.names=TRUE, "K:/Itau/Credit
Card/Paul/Dissertation/ClusteringStageOne/Cluster1TrainResults.csv")
write.csv(Cluster2, row.names=TRUE, "K:/Itau/Credit
Card/Paul/Dissertation/ClusteringStageOne/Cluster2TrainResults.csv")
write.csv(Cluster3, row.names=TRUE, "K:/Itau/Credit
Card/Paul/Dissertation/ClusteringStageOne/Cluster3TrainResults.csv")


#Assign records clusters based on Training Model
#Create new data frames and variables for clustering operations
TestData <- data.frame(x1.pc)
View(TestData)
ClusteringTestDataPC <- data.frame(CustomerDataPC)
View(ClusteringTestDataPC)

c10 <- ncol(StageOneModel)
FitModel <- data.frame(CustomerDataPCS, StageOneModel[ , c10])
c11 <- ncol(FitModel)
names(FitModel)[c11] <- "fit.cluster"
View(FitModel)

#Compute training model cluster reps and calculate distances from all reps to
test records
c3.1 <- ncol(ClusteringTestDataPC)
c12 <- max(StageOneModel[ , "fit.cluster"])
r4 <- nrow(ClusteringTestDataPC)
for (a in 1:c12) {
  g1 <- subset(FitModel, fit.cluster==a)
  v <- vector(mode="numeric", length=c3.1)
  for (b in 1:c3.1) {
    v[b] <- mean(g1[ , b])
  }
  v <- rbind(v)
  v1 <- vector(mode="numeric", length=r4)
  for (c1.1 in 1:r4) {
    v1[c1.1] <- rdist(ClusteringTestDataPC[c1.1, 1:c3.1], v)
  }
  head <- paste("EucDist", a, sep = "")
  ClusteringTestDataPC[head] <- v1
}
View(ClusteringTestDataPC)

#Assign test records to clusters
```

```
c13 <- ncol(ClusteringTestDataPC)
z3 <- c3.1 + 1
v2 <- vector(mode="numeric", length=r4)
v3 <- vector(mode="numeric", length=r4)
for (h in 1:r4) {
  t1 <- min(ClusteringTestDataPC[h, z3:c13])
  v2[h] <- t1
  t2 <- which.min(apply(ClusteringTestDataPC[h, z3:c13], MARGIN=2, min))
  v3[h] <- t2
}
ClusteringTestDataPC["EuclideanDistance"] <- v2
ClusteringTestDataPC["Test.Cluster"] <- v3
TestData["fit.cluster2"] <- v3
TestModel <- data.frame(TestData)
View(TestModel)

#3D Plots of top 3 principal components with cluster assignments depicted
c13 <- ncol(TestModel)
c14 <- c13-4
c15 <- c13-2
p6 <- data.frame(TestModel[ , c14:c15])
ddd <- data.frame(p6, TestModel[ , c13])
names(ddd)[1] <- "PC1"
names(ddd)[2] <- "PC2"
names(ddd)[3] <- "PC3"
names(ddd)[4] <- "Cluster"
scatterplot3d(ddd, col.axis = "green", col.grid = "green", main = "Principal
Components Scatterplot", angle=50, pch = 1)
scatterplot3d(ddd, col.axis = "green", col.grid = "green", main = "Principal
Components Scatterplot", angle=140, pch = 1)
scatterplot3d(ddd, col.axis = "green", col.grid = "green", main = "Principal
Components Scatterplot", angle=230, pch = 1)
scatterplot3d(ddd, col.axis = "green", col.grid = "green", main = "Principal
Components Scatterplot", angle=320, pch = 1)

#Generate data frames for subsequent analysis
x7 <- data.frame(ClusteringTestDataPC[ , 1:c3.1])
x8 <- data.frame(TestModel[ , 6:9])
x9 <- data.frame(TestModel[ , 2:5])

#Get cluster means for normalized pcs, normalized original variables, and
original variables
aggregate(x7,by=list(ClusteringTestDataPC[ , "Test.Cluster"]),FUN=mean)
aggregate(x8, by=list(TestModel[ , c13]), FUN=mean)
aggregate(x9, by=list(TestModel[ , c13]), FUN=mean)
```

```
#Get cluster medians for normalized pcs, normalized original variables, and
original variables
aggregate(x7,by=list(ClusteringTestDataPC[ , "Test.Cluster"]),FUN=median)
aggregate(x8, by=list(TestModel[ , c16]), FUN=median)
aggregate(x9, by=list(TestModel[ , c16]), FUN=median)

#Get cluster counts
count1 <- count(TestModel[ , c13]==1)
View(count1)
count2 <- count(TestModel[ , c13]==2)
View(count2)
count3 <- count(TestModel[ , c13]==3)
View(count3)

#Perform statistical testing on population
#generate data frame of normalized principal components for statistical testing
ss1 <- data.frame(TestModel[ ,c("PC1N","PC2N","PC3N","fit.cluster2")])
View(ss1)

#perform Kruskal-Wallis tests of normalized Principal components
kruskal.test(PC1N~fit.cluster2, data=ss1)
kruskal.test(PC2N~fit.cluster2, data=ss1)
kruskal.test(PC3N~fit.cluster2, data=ss1)
kruskal.test(PC1N+PC2N+PC3N~fit.cluster2, data=ss1)

#generate data frame of normalized original variables for statistical testing
ss2 <- data.frame(TestModel[
,c("AccountAgeN","CreditLimitN","AdditionalAssetsN","LatePaymentsN","fit.cluste
r2")])
View(ss2)

#perform Kruskal-Wallis tests of normalized original variables
kruskal.test(AccountAgeN~fit.cluster2, data=ss2)
kruskal.test(CreditLimitN~fit.cluster2, data=ss2)
kruskal.test(AdditionalAssetsN~fit.cluster2, data=ss2)
kruskal.test(LatePaymentsN~fit.cluster2, data=ss2)
kruskal.test(AccountAgeN+CreditLimitN+AdditionalAssetsN+LatePaymentsN~fit.
cluster2, data=ss2)

#generate data frame of original variables for statistical testing
ss3 <- data.frame(TestModel[
,c("AccountAge","CreditLimit","AdditionalAssets","LatePayments","fit.cluster2")])
View(ss3)

#perform Kruskal-Wallis tests of original variables
kruskal.test(AccountAge~fit.cluster2, data=ss3)
kruskal.test(CreditLimit~fit.cluster2, data=ss3)
kruskal.test(AdditionalAssets~fit.cluster2, data=ss3)
kruskal.test(LatePayments~fit.cluster2, data=ss3)
kruskal.test(AccountAge+CreditLimit+AdditionalAssets+LatePayments~fit.cluster
2, data=ss3)

#Get a representative sample of TestModel data for statistical analysis
s <- arrange(TestModel,fit.cluster2)
View(s)
s1 <- data.frame(s[ ,1:c13])
View(s1)

r <- nrow(s1)
View(r)
col <- ncol(s1)
View(col)
sc=30000
samplepct=round(sc/r,8)
View(samplepct)
step=round(r/sc,0)
View(step)
z=matrix(nrow=r*samplepct-21, ncol=col)
View(z)
y <- 1
i <- 1
while (i < r) {
  z[y, ] <- as.matrix(s1[i, 1:c13])
  y<-y+1
  i<-i+step
}
View(z)
s2 <- data.frame(z[ , 1:c13])
View(s2)

#Attach appropriate column names to data frame s2
names(s2)[1] <- "Record"
names(s2)[2] <- "AccountAge"
names(s2)[3] <- "CreditLimit"
names(s2)[4] <- "AdditionalAssets"
names(s2)[5] <- "LatePayments"
names(s2)[6] <- "AccountAgeN"
names(s2)[7] <- "CreditLimitN"
names(s2)[8] <- "AdditionalAssetsN"
names(s2)[9] <- "LatePaymentsN"
names(s2)[10] <- "PC1"
names(s2)[11] <- "PC2"
names(s2)[12] <- "PC3"
names(s2)[13] <- "PC4"
names(s2)[14] <- "PC1N"
names(s2)[15] <- "PC2N"
names(s2)[16] <- "PC3N"
names(s2)[17] <- "PC4N"
names(s2)[18] <- "fit.cluster2"
s3 <- arrange(s2,Record)
View(s3)

#Perform statistical testing on representative sample
#generate data frame of normalized principal components for statistical testing
ss1 <- data.frame(s3[ ,c("PC1N","PC2N","PC3N","fit.cluster2")])
View(ss1)

#perform Kruskal-Wallis tests of normalized Principal components
kruskal.test(PC1N~fit.cluster2, data=ss1)
kruskal.test(PC2N~fit.cluster2, data=ss1)
kruskal.test(PC3N~fit.cluster2, data=ss1)
kruskal.test(PC1N+PC2N+PC3N~fit.cluster2, data=ss1)

#generate data frame of normalized original variables for statistical testing
ss2 <- data.frame(s3[
,c("AccountAgeN","CreditLimitN","AdditionalAssetsN","LatePaymentsN","fit.cluste
r2")])
View(ss2)

#perform Kruskal-Wallis tests of normalized original variables
kruskal.test(AccountAgeN~fit.cluster2, data=ss2)
kruskal.test(CreditLimitN~fit.cluster2, data=ss2)
kruskal.test(AdditionalAssetsN~fit.cluster2, data=ss2)
kruskal.test(LatePaymentsN~fit.cluster2, data=ss2)
kruskal.test(AccountAgeN+CreditLimitN+AdditionalAssetsN+LatePaymentsN~fit.
cluster2, data=ss2)

#generate data frame of original variables for statistical testing
ss3 <- data.frame(s3[
,c("AccountAge","CreditLimit","AdditionalAssets","LatePayments","fit.cluster2")])
View(ss3)

#perform Kruskal-Wallis tests of original variables
kruskal.test(AccountAge~fit.cluster2, data=ss3)
kruskal.test(CreditLimit~fit.cluster2, data=ss3)
kruskal.test(AdditionalAssets~fit.cluster2, data=ss3)
kruskal.test(LatePayments~fit.cluster2, data=ss3)
kruskal.test(AccountAge+CreditLimit+AdditionalAssets+LatePayments~fit.cluster
2, data=ss3)


#Perform oUtlier detection via mahalanobis distance (test model -- all objects)
#Set initial variables for subsequent usage
r5 <- nrow(TestModel)
b2 <- r5*.01
x10 <- max(TestModel[ , c13])
p7 <- data.frame(ClusteringTestDataPC[ , 1:c3.1], TestModel[ , c13])
names(p7)[ncol(p7)] <- "fit.cluster2"
View(p7)
p8 <- data.frame(arrange(p7, fit.cluster2), row.names=NULL)
p9 <- data.frame(arrange(TestModel, fit.cluster2), row.names=NULL)

#Separate data by cluster, compute medians, calculate mahalanobis distances,
and create outlier plots for each cluster
for (c in 1:x10) {
  nam1 <- paste("Cluster", c, sep = "")
  z <- assign(nam1, subset(p9, fit.cluster2==c))
  z1 <- subset(p8, fit.cluster2==c)
  med <- vector(mode="numeric", length=c3.1)
  for (cc in 1:c3.1) {
    med[cc] <- median(z1[ , cc])
  }
  z2 <- data.frame(z1[ ,1:c3.1])
  z3 <- cov(z2, use = "everything" )
  md <- mahalanobis(z2, center = med, cov = z3, inverted = TRUE)
  nam4 <- paste("Cluster", c, sep = "")
  z4 <- assign(nam4, data.frame(z, md, row.names=NULL))
  y2 <- nrow(z4)
  row.names(z4) <- 1:y2
  df.2 <- c(1:y2)
  if (y2 >= b2) {
    Out.999 = quantile(md, .999)
    plot(df.2, md, xlab="Object", ylab="Mahalanobis Distance",
main=c("Cluster",c,"Outlier Plot"))
    abline(h=Out.999, col="red", lwd=2)
    cat("The following rows in cluster",c,"data appear very
suspicious:",which(md>Out.999),"\n")
    qqplot(md, df.2, xlab="Mahalanobis Distance", ylab="Ranked Observation",
main=c("Cluster",c,"QQ Plot"))
    abline(v=Out.999, col="red", lwd=2)
  }
  if (y2 %in% c(3:b2)) {
    Out.999 = quantile(md, .999)
    plot(df.2, md, xlab="Object", ylab="Mahalanobis Distance",
main=c("Cluster",c,"Outlier Plot"))
    abline(h=Out.999, col="red", lwd=2)
    cat("The following rows in cluster",c,"data appear very
suspicious:",which(md>Out.999),"\n")
    qqplot(md, df.2, xlab="Mahalanobis Distance", ylab="Ranked Observation",
main=c("Cluster",c,"QQ Plot"))
    abline(v=Out.999, col="red", lwd=2)
    cat("Cluster",c,"has small relative membership and this warrants investigation
of all associated objects","\n")
  }
  if (y2 < 3) {
    cat("Cluster",c,"has an extremely small membership and this clearly warrants
investigation of all associated objects","\n")
  }
}

#Export processed information, segregated by cluster
write.csv(Cluster1, row.names=TRUE, "K:/Itau/Credit
Card/Paul/Dissertation/ClusteringStageOne/Cluster1TestResults.csv")
write.csv(Cluster2, row.names=TRUE, "K:/Itau/Credit
Card/Paul/Dissertation/ClusteringStageOne/Cluster2TestResults.csv")
write.csv(Cluster3, row.names=TRUE, "K:/Itau/Credit
Card/Paul/Dissertation/ClusteringStageOne/Cluster3TestResults.csv")

#Export ClusteringTestDataPC and TestModel
write.csv(ClusteringTestDataPC, row.names=TRUE, "K:/Itau/Credit
Card/Paul/Dissertation/ClusteringStageOne/ClusteringTestDataPC.csv")
write.csv(TestModel, row.names=TRUE, "K:/Itau/Credit
Card/Paul/Dissertation/ClusteringStageOne/TestModel.csv")
```

APPENDIX B: RSCRIPT - CHAPTER 3 CLUSTERING

```
#Chapter Two - Stage Two Cluster Analysis and Testing
#Analyze the 3 Primary Clusters (nested cluster analysis)
library(cluster)
library(mclust)
library(stats)
library(plyr)
library(Rcmdr)
library(gplots)
library(scatterplot3d)
library(chemometrics)
library(fields)
library(fastcluster)
library(EMCluster)

#Increase memory limit
memory.size(max=TRUE)
memory.limit(size=999999)

#Read data set StageOneFinal in R
Stage2 <- read.csv("K:/Itau/Credit
Card/Paul/Dissertation/ClusteringStageOne/StageOneFinalPC.csv")
View(Stage2)
x <- data.frame(Stage2)
View(x)

#Create data frames and find n for each cluster
cluster1 <- subset(x, fit.cluster2==1)
View(cluster1)
n1 <- nrow(cluster1)
View(n1)
cluster2 <- subset(x, fit.cluster2==2)
View(cluster2)
n2 <- nrow(cluster2)
View(n2)
cluster3 <- subset(x, fit.cluster2==3)
View(cluster3)
n3 <- nrow(cluster3)
View(n3)

#------------------------------------------------------------------------------------------
#Obtain discount % descriptive statistics for each cluster (a digression)
#Entails calculations for each cluster and measure
```

```
s1 <- cluster1[[20]]
st1 <- data.frame(s1)
names(st1)[1] <- "Discount"
stat1 <- subset(st1, Discount>-0)
stats1 <- stat1[ ,1]
mean1 <- mean(stats1)
median1 <- median(stats1)
mode1 <- sort(table(stats1),decreasing=TRUE)[1:1]
View(mode1)
min1 <- min(stats1)
max1 <- max(stats1)
cluster1stats <- c(mean1, median1, min1, max1)
View(cluster1stats)

s2 <- cluster2[[20]]
st2 <- data.frame(s2)
names(st2)[1] <- "Discount"
stat2 <- subset(st2, Discount>-0)
stats2 <- stat2[ ,1]
mean2 <- mean(stats2)
median2 <- median(stats2)
mode2 <- sort(table(stats2),decreasing=TRUE)[1:1]
View(mode2)
min2 <- min(stats2)
max2 <-max(stats2)
cluster2stats <- c(mean2, median2, min2, max2)
View(cluster2stats)

s3 <- cluster3[[20]]
st3 <- data.frame(s3)
names(st3)[1] <- "Discount"
stat3 <- subset(st3, Discount>-0)
stats3 <- stat3[ ,1]
mean3 <- mean(stats3)
median3 <- median(stats3)
mode3 <- sort(table(stats3),decreasing=TRUE)[1:1]
View(mode3)
min3 <- min(stats3)
max3 <-max(stats3)
cluster3stats <- c(mean3, median3, min3, max3)
View(cluster3stats)
#-------------------------------------------------------------------------------------------

#CLUSTER 1 - BEGIN STAGE 2 PREPROCESSING AND ANALYSIS
#Create data frame of normalized original variables for performing principal
components analysis
```

```
x1 <- data.frame(cluster1[ , c("Record", "AccountAge", "CreditLimit",
"AdditionalAssets", "LatePayments")], row.names=NULL)
View(x1)

#Normalize four dimensions to be clustered
#Assumes dimensions to be normalized are in columns 2, 3, 4, and 5 of data
#Normalize AccountAge column
aa1 <- data.frame(x1[ , 2])
names(aa1)[1] <- "AccountAge"
View(aa1)
maxaa <- max(aa1)
minaa <- min(aa1)
aan <- vector(mode="numeric", length=n1)
i<-1
while(i<n1+1) {
  aan[i] <- ((aa1[i, 1] - minaa)/(maxaa-minaa))
  i <- i+1
}

#Check/verify max & min normalized AccountAge column values (max=1, min=0)
View(max(aan))
View(min(aan))

#Add normalized AccountAge column to data frame x1
x1["AccountAgeN"] <- aan
View(x1)

#Normalize CreditLimit column
cl1 <- data.frame(x1[ , 3])
names(cl1)[1] <- "CreditLimit"
View(cl1)
maxcl <- max(cl1)
View(maxcl)
mincl <- min(cl1)
View(mincl)
cln <- vector(mode="numeric", length=n1)
i<-1
while(i<n1+1) {
  cln[i] <- ((cl1[i, 1] - mincl)/(maxcl-mincl))
  i <- i+1
}

#Check/verify max & min normalized CreditLmit column values (max=1, min=0)
View(max(cln))
View(min(cln))
```

```
#Add normalized CreditLimit column to data frame x1
x1["CreditLimitN"] <- cln

#Normalize AdditionalAssets column
adda1 <- data.frame(x1[ , 4])
names(adda1)[1] <- "AdditionalAssets"
View(adda1)
maxadda <- max(adda1)
View(maxadda)
minadda <- min(adda1)
View(minadda)
addan <- vector(mode="numeric", length=n1)
i<-1
while(i<n1+1) {
  addan[i] <- ((adda1[i, 1] - minadda)/(maxadda-minadda))
  i <- i+1
}

#Check/verify max & min normalized AdditionalAssets column values (max=1,
min=0)
View(max(addan))
View(min(addan))

#Add normalized AdditionalAssets column to data frame x1
x1["AdditionalAssetsN"] <- addan
View(x1)

#Normalize LatePayments column
lp1 <- data.frame(x1[ , 5])
names(lp1)[1] <- "LatePayments"
View(lp1)
maxlp <- max(lp1)
View(maxlp)
minlp <- min(lp1)
View(minlp)
lpn <- vector(mode="numeric", length=n1)
i<-1
while(i<n1+1) {
  lpn[i] <- ((lp1[i, 1] - minlp)/(maxlp-minlp))
  i <- i+1
}

#Check/verify max & min normalized LatePayments column values (max=1,
min=0)
View(max(lpn))
View(min(lpn))
```

```
#Add normalized LatePayments column to data frame x1
x1["LatePaymentsN"] <- lpn
View(x1)

#Create data frame of normalized original variables for performing principal
components analysis
grp1 <- data.frame(x1[ , c("AccountAgeN", "CreditLimitN", "AdditionalAssetsN",
"LatePaymentsN")])
View(grp1)

#Perform principal component analysis and show associated information
trans <- prcomp(grp1, center=TRUE, scale.=TRUE)
print(trans)
summary(trans)
plot(trans, typ="l", col="2", lwd="2")

PC = predict(trans, grp1)
print(PC[1:10, ])
View(PC)
plot(PC, type="p", col="2")
plot(PC, type="l", col="2")
plot(PC, type="b", col="2")

#Create a data frame of x1 and principal components, thus creating x1.pc
x1.pc <- cbind(x1, PC, row.names=NULL)
View(x1.pc)

#Normalize all four principal components
#Assumes dimensions to be normalized are in columns 10, 11, 12, and 13 of
data
#Normalize PC1 column
pc1 <- x1.pc[[10]]
View(pc1)
maxpc1 <- max(pc1)
View(maxpc1)
minpc1 <- min(pc1)
View(minpc1)
y <- n1
View(y)
pc1n <- vector(mode="numeric", length=y)
View(pc1n)
i<-1
while(i<y+1) {
  pc1n[i] <- ((pc1[i] - minpc1)/(maxpc1-minpc1))
  i <- i+1
```

```
}
View(pc1n)

#Check/verify max & min normalized PC1 column values (max=1, min=0)
View(max(pc1n))
View(min(pc1n))

#Add normalized PC1 column to data frame x1.pc
x1.pc["PC1N"] <- pc1n
View(x1.pc)

#Normalize PC2 column
pc2 <- x1.pc[[11]]
View(pc2)
maxpc2 <- max(pc2)
View(maxpc2)
minpc2 <- min(pc2)
View(minpc2)
pc2n <- vector(mode="numeric", length=y)
i<-1
while(i<y+1) {
  pc2n[i] <- ((pc2[i] - minpc2)/(maxpc2-minpc2))
  i <- i+1
}
View(pc2n)

#Check/verify max & min normalized PC2 column values (max=1, min=0)
View(max(pc2n))
View(min(pc2n))

#Add normalized PC2 column to data frame x1.pc
x1.pc["PC2N"] <- pc2n
View(x1.pc)

#Normalize PC3 column
pc3 <- x1.pc[[12]]
View(pc3)
maxpc3 <- max(pc3)
View(maxpc3)
minpc3 <- min(pc3)
View(minpc3)
pc3n <- vector(mode="numeric", length=y)
i<-1
while(i<y+1) {
  pc3n[i] <- ((pc3[i] - minpc3)/(maxpc3-minpc3))
  i <- i+1
```

```
}
View(pc3n)

#Check/verify max & min normalized PC3 column values (max=1, min=0)
View(max(pc3n))
View(min(pc3n))

#Add normalized PC3 column to data frame x1.pc
x1.pc["PC3N"] <- pc3n
View(x1.pc)

#Normalize PC4 column
pc4 <- x1.pc[[13]]
View(pc4)
maxpc4 <- max(pc4)
View(maxpc4)
minpc4 <- min(pc4)
View(minpc4)
pc4n <- vector(mode="numeric", length=y)
i<-1
while(i<y+1) {
  pc4n[i] <- ((pc4[i] - minpc4)/(maxpc4-minpc4))
  i <- i+1
}
View(pc4n)

#Check/verify max & min normalized PC4 column values (max=1, min=0)
View(max(pc4n))
View(min(pc4n))

#Add normalized PC4 column to data frame x1.pc
x1.pc["PC4N"] <- pc4n
View(x1.pc)

#Export x1.pc for subsequent model building (pre-processed cluster 1 data)
write.csv(x1.pc, "K:/Itau/Credit
Card/Paul/Dissertation/ClusteringStageTwo/Cluster1Data.csv")
#Reload x1.pc (if not currently loaded)
x1.pc <- read.csv("K:/Itau/Credit
Card/Paul/Dissertation/ClusteringStageTwo/Cluster1Data.csv")
View(x1.pc)

#Perform model building simulation routines
#First get a random sample of 40,000 records from population of x1.pc (set s =
sample size)
```

```
#Necessary because silhouette calculation will not allow use of long vectors (>
2^31)
#Also necessary because processing is extremely inefficient on pam
s <- 40000
p <- s/nrow(x1.pc)
View(p)
x1.pcs <- x1.pc[sample(nrow(x1.pc),replace=F,size=p*nrow(x1.pc)),]
row.names(x1.pcs)<-NULL
View(x1.pcs)

#Create a data frame Cluster1PCS of 3 top PCs from x1.pcs for computing
silhouette coefficients
c <- ncol(x1.pcs)
c1 <- c-3
c2 <- c-1
Cluster1PCS <- data.frame(x1.pcs[ , c1:c2])
View(Cluster1PCS)

#Execute Kmeans clustering model performance sequence
#Use loop structure for initial Kmeans model building process
nclust=6
x=matrix(nrow=nclust-1, ncol=2)
d <- dist(Cluster1PCS, "euclidean")
y <- 1
c <- 2
for (c in c:nclust) {
  fit <- kmeans(Cluster1PCS, c, iter.max=500, nstart=10, algorithm="Lloyd")
  sil <- silhouette(fit$cluster, d)
  avgsil <- summary(sil, FUN=mean) $avg.width
  x[y, ] <- c(c, avgsil)
  y<-y+1
}
View(x)
max(x[ ,2])

#Create data frame z, Sort in descending order by Silhouette Coefficient
z <- data.frame(x)
names(z)[1] <- "Clusters"
names(z)[2] <- "Silhouette"
View(z)

s <- arrange(z, desc(Silhouette))
View(s)

#Export processed data
```

```
write.csv(s, "K:/Itau/Credit
Card/Paul/Dissertation/ClusteringStageTwo/Cluster1SCKmeans.csv")


#Execute PAM clustering model performance sequence
#Use loop structure for initial model building process
nclust= 6
x=matrix(nrow=nclust-1, ncol=2)
d <- dist(Cluster1PCS, method = "euclidean")
y <- 1
c <- 2
for (c in c:nclust) {
  fit <- pam(Cluster1PCS, c)
  sil <- silhouette(fit$clustering, d)
  avgsil <- summary(sil, FUN=mean) $avg.width
  x[y, ] <- c(c, avgsil)
  y<-y+1
}
View(x)
max(x[ ,2])

#Create data frame z, Sort in descending order by Silhouette Coefficient
z <- data.frame(x)
names(z)[1] <- "Clusters"
names(z)[2] <- "Silhouette"
View(z)

s <- arrange(z, desc(Silhouette))
View(s)

#Export processed data
write.csv(s, "K:/Itau/Credit
Card/Paul/Dissertation/ClusteringStageTwo/Cluster1SCPAM.csv")


#Execute Complete-Link Hierarchical clustering model performance sequence
#Use loop structure for initial model building process
nclust= 6
x=matrix(nrow=nclust-1, ncol=2)
d <- dist(Cluster1PCS, method = "euclidean")
y <- 1
c <- 2
for (c in c:nclust) {
  fit2 <- hclust(d, method = "complete")
  memb <- cutree(fit2, k=c)
  fit$clusters <- assignCluster(Cluster1PCS, x1.pcs, memb)
  fit$cluster <- as.integer(fit$clusters)
  sil <- silhouette(fit$cluster, d)
  avgsil <- summary(sil, FUN=mean) $avg.width
  x[y, ] <- c(c, avgsil)
  y<-y+1
}
View(x)
max(x[ ,2])

#Create data frame z, Sort in descending order by Silhouette Coefficient
z <- data.frame(x)
names(z)[1] <- "Clusters"
names(z)[2] <- "Silhouette"
View(z)

s <- arrange(z, desc(Silhouette))
View(s)

#Export processed data
write.csv(s, "K:/Itau/Credit
Card/Paul/Dissertation/ClusteringStageTwo/Cluster1SCComplete.csv")


#Execute Ward's Method Hierarchical clustering model performance sequence
#Use loop structure for initial model building process
nclust= 6
x=matrix(nrow=nclust-1, ncol=2)
d <- dist(Cluster1PCS, method = "euclidean")
y <- 1
c <- 2
for (c in c:nclust) {
  fit2 <- hclust(d, method = "ward.D2")
  memb <- cutree(fit2, k=c)
  fit$clusters <- assignCluster(Cluster1PCS, x1.pcs, memb)
  fit$cluster <- as.integer(fit$clusters)
  sil <- silhouette(fit$cluster, d)
  avgsil <- summary(sil, FUN=mean) $avg.width
  x[y, ] <- c(c, avgsil)
  y<-y+1
}
View(x)
max(x[ ,2])

#Create data frame z, Sort in descending order by Silhouette Coefficient
z <- data.frame(x)
names(z)[1] <- "Clusters"
names(z)[2] <- "Silhouette"
View(z)

s <- arrange(z, desc(Silhouette))
View(s)

#Export processed data
write.csv(s, "K:/Itau/Credit
Card/Paul/Dissertation/ClusteringStageTwo/Cluster1SCWard.csv")


#Execute Expectation Maximization clustering model performance sequence
#Use loop structure for initial EM model building process
nclust=6
x=matrix(nrow=nclust-1, ncol=2)
d <- dist(Cluster1PCS, "euclidean")
y <- 1
c <- 2
for (c in c:nclust) {
  fit <- init.EM(Cluster1PCS, nclass = c, EMC = .EMC.Rnd, stable.solution =
TRUE, min.n = 1, min.n.iter = 10, method = "Rnd.EM")
  sil <- silhouette(fit$class, d)
  avgsil <- summary(sil, FUN=mean) $avg.width
  x[y, ] <- c(c, avgsil)
  y<-y+1
}
View(x)
max(x[ ,2])

#Create data frame z, Sort in descending order by Silhouette Coefficient
z <- data.frame(x)
names(z)[1] <- "Clusters"
names(z)[2] <- "Silhouette"
View(z)

s <- arrange(z, desc(Silhouette))
View(s)

#Export processed data
write.csv(s, "K:/Itau/Credit
Card/Paul/Dissertation/ClusteringStageTwo/Cluster1SCEM.csv")


#Build final cluster 1 model - based upon silhouette results
#Decision 7/29/15:  create K-means 2-cluster model (set c=2)
c <- 2
```

```
#Import cluster 1 data for clustering (if necessary)
ClusterOne <- read.csv("K:/Itau/Credit
Card/Paul/Dissertation/ClusteringStageTwo/Cluster1Data.csv")
x1.pc <- data.frame(ClusterOne)
r1 <- nrow(x1.pc)
c3 <- ncol(x1.pc)
View(x1.pc)

#Get random sample of records for building training model
#Default sample size set at 50,000 (change if needed)
sample <- 50000
p1 <- sample/r1
x1.pcs <- x1.pc[sample(nrow(x1.pc),replace=F,size = p1*nrow(x1.pc)),]
row.names(x1.pcs) <- NULL
View(x1.pcs)

#Set variables and data frame of principal components for clustering
c4 <- c3-3
c5 <- c3-1
Cluster1.pcs <- data.frame(x1.pcs[ , c4:c5])
View(Cluster1.pcs)

#Create sub-cluster training model relative to cluster 1
fit <- kmeans(Cluster1.pcs, c, iter.max=500, nstart=10, algorithm="Lloyd")
ClusterOneModel <- data.frame(Cluster1.pcs, fit$cluster)
View(ClusterOneModel)

#3D Plots of top 3 principal components with cluster assignments depicted
p2 <- data.frame(Cluster1.pcs)
ddd <- data.frame(p2, ClusterOneModel[ , "fit.cluster"])
names(ddd)[1] <- "PC1"
names(ddd)[2] <- "PC2"
names(ddd)[3] <- "PC3"
names(ddd)[4] <- "Cluster"
scatterplot3d(ddd, col.axis = "green", col.grid = "green", main = "Principal
Components Scatterplot", angle=50, pch = 1)
scatterplot3d(ddd, col.axis = "green", col.grid = "green", main = "Principal
Components Scatterplot", angle=140, pch = 1)
scatterplot3d(ddd, col.axis = "green", col.grid = "green", main = "Principal
Components Scatterplot", angle=255, pch = 1)
scatterplot3d(ddd, col.axis = "green", col.grid = "green", main = "Principal
Components Scatterplot", angle=285, pch = 1)

#Generate data frames for subsequent analysis of training data
x3 <- data.frame(Cluster1.pcs)
```

```
x4 <- data.frame(x1.pcs[ , 6:9])
x5 <- data.frame(x1.pcs[ , 2:5])

#Get cluster means for normalized pcs, normalized original variables, and
original variables
aggregate(x3,by=list(fit$cluster),FUN=mean)
aggregate(x4, by=list(fit$cluster), FUN=mean)
aggregate(x5, by=list(fit$cluster), FUN=mean)

#Get cluster medians for normalized pcs, normalized original variables, and
original variables
aggregate(x3,by=list(fit$cluster),FUN=median)
aggregate(x4, by=list(fit$cluster), FUN=median)
aggregate(x5, by=list(fit$cluster), FUN=median)

#Create final stage two model data set
StageTwoModel <- data.frame(x1.pcs, ClusterOneModel[ , "fit.cluster"])
names(StageTwoModel)[ncol(StageTwoModel)] <- "fit.cluster"
View(StageTwoModel)

#Cluster 1 - Sub-cluster OUtlier Detection Via Mahalanobis Distance (training
model)
#Set initial variables for subsequent usage
r2 <- nrow(StageTwoModel)
c6 <- ncol(Cluster1.pcs)
b1 <- r2*.01
x6 <- max(StageTwoModel[ ,"fit.cluster"])
p3 <- data.frame(ClusterOneModel)
p4 <- data.frame(arrange(p3, fit.cluster), row.names=NULL)
p5 <- data.frame(arrange(StageTwoModel, fit.cluster), row.names=NULL)

#Separate data by cluster, compute medians, calculate mahalanobis distances,
and create outlier plots for each cluster
for (c in 1:x6) {
  nam1 <- paste("Cluster", c, sep = "")
  z <- assign(nam1, subset(p5, fit.cluster==c))
  z1 <- subset(p4, fit.cluster==c)
  med <- vector(mode="numeric", length=c6)
  for (cc in 1:c6) {
    med[cc] <- median(z1[ , cc])
  }
  z2 <- data.frame(z1[ ,1:c6])
  z3 <- cov(z2, use = "everything" )
  md <- mahalanobis(z2, center = med, cov = z3, inverted = TRUE)
  nam4 <- paste("Cluster", c, sep = "")
  z4 <- assign(nam4, data.frame(z, md, row.names=NULL))
  y2 <- nrow(z4)
  row.names(z4) <- 1:y2
  df.2 <- c(1:y2)
  if (y2 >= b1) {
    Out.999 = quantile(md, .999)
    plot(df.2, md, xlab="Object", ylab="Mahalanobis Distance", main=c("Cluster 1,
Subcluster",c,"Outlier Plot"))
    abline(h=Out.999, col="red", lwd=2)
    cat("The following rows in cluster 1, sub-cluster",c,"data appear very
suspicious:",which(md>Out.999),"\n")
    qqplot(md, df.2, xlab="Mahalanobis Distance", ylab="Ranked Observation",
main=c("Cluster 1, Subcluster",c,"QQ Plot"))
    abline(v=Out.999, col="red", lwd=2)
  }
  if (y2 %in% c(3:b1)) {
    Out.999 = quantile(md, .999)
    plot(df.2, md, xlab="Object", ylab="Mahalanobis Distance", main=c("Cluster 1,
Subcluster",c,"Outlier Plot"))
    abline(h=Out.999, col="red", lwd=2)
    cat("The following rows in cluster 1, sub-cluster",c,"data appear very
suspicious:",which(md>Out.999),"\n")
    qqplot(md, df.2, xlab="Mahalanobis Distance", ylab="Ranked Observation",
main=c("Cluster1, Subcluster",c,"QQ Plot"))
    abline(v=Out.999, col="red", lwd=2)
    cat("Cluster 1, sub-cluster",c,"has small relative membership and this warrants
investigation of all associated objects","\n")
  }
  if (y2 < 3) {
    cat("Cluster 1, sub-Cluster",c,"has an extremely small membership and this
clearly warrants investigation of all associated objects","\n")
  }
}

#Export processed training model information, segregated by sub-cluster
write.csv(Cluster1, row.names=TRUE, "K:/Itau/Credit
Card/Paul/Dissertation/ClusteringStageTwo/Cluster1Subcluster1TrainResults.csv
")
write.csv(Cluster2, row.names=TRUE, "K:/Itau/Credit
Card/Paul/Dissertation/ClusteringStageTwo/Cluster1Subcluster2TrainResults.csv
")


#Cluster 1 - Final assignment of all records to sub-clusters based upon training
model results
#Create new data frames and variables for clustering operations
TestData <- data.frame(x1.pc)
```

```
View(TestData)
c3 <- ncol(x1.pc)
c4 <- c3-3
c5 <- c3-1
Cluster1TestDataPC <- data.frame(x1.pc[ , c4:c5])
View(Cluster1TestDataPC)

FitModel <- data.frame(ClusterOneModel)
View(FitModel)

#Compute training model cluster reps and calculate distances from all reps to
test records
c3.1 <- ncol(Cluster1TestDataPC)
c12 <- max(ClusterOneModel[ , "fit.cluster"])
r4 <- nrow(Cluster1TestDataPC)
for (a in 1:c12) {
  g1 <- subset(FitModel, fit.cluster==a)
  v <- vector(mode="numeric", length=c3.1)
  for (b in 1:c3.1) {
    v[b] <- mean(g1[ , b])
  }
  v <- rbind(v)
  v1 <- vector(mode="numeric", length=r4)
  for (c1.1 in 1:r4) {
    v1[c1.1] <- rdist(Cluster1TestDataPC[c1.1, 1:c3.1], v)
  }
  head <- paste("EucDist", a, sep = "")
  Cluster1TestDataPC[head] <- v1
}
View(Cluster1TestDataPC)

#Assign test records to clusters
c13 <- ncol(Cluster1TestDataPC)
z3 <- c3.1 + 1
v2 <- vector(mode="numeric", length=r4)
v3 <- vector(mode="numeric", length=r4)
for (h in 1:r4) {
  t1 <- min(Cluster1TestDataPC[h, z3:c13])
  v2[h] <- t1
  t2 <- which.min(apply(Cluster1TestDataPC[h, z3:c13], MARGIN=2, min))
  v3[h] <- t2
}
Cluster1TestDataPC["EuclideanDistance"] <- v2
Cluster1TestDataPC["Test.Cluster"] <- v3
TestData["fit.cluster2"] <- v3
TestModel <- data.frame(TestData)
```

```
View(TestModel)

#3D Plots of top 3 principal components with cluster assignments depicted
c13 <- ncol(TestModel)
c14 <- c13-4
c15 <- c13-2
p6 <- data.frame(TestModel[ , c14:c15])
ddd <- data.frame(p6, TestModel[ , c13])
names(ddd)[1] <- "PC1"
names(ddd)[2] <- "PC2"
names(ddd)[3] <- "PC3"
names(ddd)[4] <- "Cluster"
scatterplot3d(ddd, col.axis = "green", col.grid = "green", main = "Principal
Components Scatterplot", angle=50, pch = 1)
scatterplot3d(ddd, col.axis = "green", col.grid = "green", main = "Principal
Components Scatterplot", angle=140, pch = 1)
scatterplot3d(ddd, col.axis = "green", col.grid = "green", main = "Principal
Components Scatterplot", angle=230, pch = 1)
scatterplot3d(ddd, col.axis = "green", col.grid = "green", main = "Principal
Components Scatterplot", angle=320, pch = 1)

#Generate data frames for subsequent analysis
x7 <- data.frame(Cluster1TestDataPC[ , 1:c3.1])
x8 <- data.frame(TestModel[ , 6:9])
x9 <- data.frame(TestModel[ , 2:5])

#Get cluster means for normalized pcs, normalized original variables, and
original variables
aggregate(x7,by=list(Cluster1TestDataPC[ , "Test.Cluster"]),FUN=mean)
aggregate(x8, by=list(TestModel[ , c13]), FUN=mean)
aggregate(x9, by=list(TestModel[ , c13]), FUN=mean)

#Get cluster medians for normalized pcs, normalized original variables, and
original variables
aggregate(x7,by=list(Cluster1TestDataPC[ , "Test.Cluster"]),FUN=median)
aggregate(x8, by=list(TestModel[ , c13]), FUN=median)
aggregate(x9, by=list(TestModel[ , c13]), FUN=median)

#Get sub-cluster counts
count1 <- count(TestModel[ , c13]==1)
View(count1)
count2 <- count(TestModel[ , c13]==2)
View(count2)

#Perform statistical testing on population
#generate data frame of normalized principal components for statistical testing
```

```
ss1 <- data.frame(TestModel[ ,c("PC1N","PC2N","PC3N","fit.cluster2")])
View(ss1)

#perform Kruskal-Wallis tests of normalized Principal components
kruskal.test(PC1N~fit.cluster2, data=ss1)
kruskal.test(PC2N~fit.cluster2, data=ss1)
kruskal.test(PC3N~fit.cluster2, data=ss1)
kruskal.test(PC1N+PC2N+PC3N~fit.cluster2, data=ss1)

#generate data frame of normalized original variables for statistical testing
ss2 <- data.frame(TestModel[
,c("AccountAgeN","CreditLimitN","AdditionalAssetsN","LatePaymentsN","fit.cluste
r2")])
View(ss2)

#perform Kruskal-Wallis tests of normalized original variables
kruskal.test(AccountAgeN~fit.cluster2, data=ss2)
kruskal.test(CreditLimitN~fit.cluster2, data=ss2)
kruskal.test(AdditionalAssetsN~fit.cluster2, data=ss2)
kruskal.test(LatePaymentsN~fit.cluster2, data=ss2)
kruskal.test(AccountAgeN+CreditLimitN+AdditionalAssetsN+LatePaymentsN~fit.
cluster2, data=ss2)

#generate data frame of original variables for statistical testing
ss3 <- data.frame(TestModel[
,c("AccountAge","CreditLimit","AdditionalAssets","LatePayments","fit.cluster2")])
View(ss3)

#perform Kruskal-Wallis tests of original variables
kruskal.test(AccountAge~fit.cluster2, data=ss3)
kruskal.test(CreditLimit~fit.cluster2, data=ss3)
kruskal.test(AdditionalAssets~fit.cluster2, data=ss3)
kruskal.test(LatePayments~fit.cluster2, data=ss3)
kruskal.test(AccountAge+CreditLimit+AdditionalAssets+LatePayments~fit.cluster
2, data=ss3)

#Get a representative sample of TestModel data for statistical analysis
s <- arrange(TestModel,fit.cluster2)
View(s)
s1 <- data.frame(s[ ,1:c13])
View(s1)

r <- nrow(s1)
col <- ncol(s1)
sc=369
samplepct=round(sc/r,8)
```

```
step=round(r/sc,0)
z=matrix(nrow=r*samplepct+2, ncol=col)
y <- 1
i <- 1
while (i < r) {
  z[y, ] <- as.matrix(s1[i, 1:c13])
  y<-y+1
  i<-i+step
}
View(z)
s2 <- data.frame(z[ , 1:c13])
View(s2)

#Attach appropriate column names to data frame s2
names(s2)[1] <- "Record"
names(s2)[2] <- "AccountAge"
names(s2)[3] <- "CreditLimit"
names(s2)[4] <- "AdditionalAssets"
names(s2)[5] <- "LatePayments"
names(s2)[6] <- "AccountAgeN"
names(s2)[7] <- "CreditLimitN"
names(s2)[8] <- "AdditionalAssetsN"
names(s2)[9] <- "LatePaymentsN"
names(s2)[10] <- "PC1"
names(s2)[11] <- "PC2"
names(s2)[12] <- "PC3"
names(s2)[13] <- "PC4"
names(s2)[14] <- "PC1N"
names(s2)[15] <- "PC2N"
names(s2)[16] <- "PC3N"
names(s2)[17] <- "PC4N"
names(s2)[18] <- "fit.cluster2"
s3 <- arrange(s2,Record)
View(s3)

#Perform statistical testing on representative sample
#generate data frame of normalized principal components for statistical testing
ss1 <- data.frame(s3[ ,c("PC1N","PC2N","PC3N","fit.cluster2")])
View(ss1)

#perform Kruskal-Wallis tests of normalized Principal components
kruskal.test(PC1N~fit.cluster2, data=ss1)
kruskal.test(PC2N~fit.cluster2, data=ss1)
kruskal.test(PC3N~fit.cluster2, data=ss1)
kruskal.test(PC1N+PC2N+PC3N~fit.cluster2, data=ss1)
```

```
#generate data frame of normalized original variables for statistical testing
ss2 <- data.frame(s3[
,c("AccountAgeN","CreditLimitN","AdditionalAssetsN","LatePaymentsN","fit.cluste
r2")])
View(ss2)

#perform Kruskal-Wallis tests of normalized original variables
kruskal.test(AccountAgeN~fit.cluster2, data=ss2)
kruskal.test(CreditLimitN~fit.cluster2, data=ss2)
kruskal.test(AdditionalAssetsN~fit.cluster2, data=ss2)
kruskal.test(LatePaymentsN~fit.cluster2, data=ss2)
kruskal.test(AccountAgeN+CreditLimitN+AdditionalAssetsN+LatePaymentsN~fit.
cluster2, data=ss2)

#generate data frame of original variables for statistical testing
ss3 <- data.frame(s3[
,c("AccountAge","CreditLimit","AdditionalAssets","LatePayments","fit.cluster2")])
View(ss3)

#perform Kruskal-Wallis tests of original variables
kruskal.test(AccountAge~fit.cluster2, data=ss3)
kruskal.test(CreditLimit~fit.cluster2, data=ss3)
kruskal.test(AdditionalAssets~fit.cluster2, data=ss3)
kruskal.test(LatePayments~fit.cluster2, data=ss3)
kruskal.test(AccountAge+CreditLimit+AdditionalAssets+LatePayments~fit.cluster
2, data=ss3)


#Cluster 1 - Sub-Cluster OUtlier Detection Via Mahalanobis Distance (test model
-- all objects)
#Set initial variables for subsequent usage
r5 <- nrow(TestModel)
b2 <- r5*.01
x10 <- max(TestModel[ , c13])
p7 <- data.frame(Cluster1TestDataPC[ , 1:c3.1], TestModel[ , c13])
names(p7)[ncol(p7)] <- "fit.cluster2"
View(p7)
p8 <- data.frame(arrange(p7, fit.cluster2), row.names=NULL)
p9 <- data.frame(arrange(TestModel, fit.cluster2), row.names=NULL)

#Separate data by cluster, compute medians, calculate mahalanobis distances,
and create outlier plots for each cluster
for (c in 1:x10) {
  nam1 <- paste("Cluster", c, sep = "")
  z <- assign(nam1, subset(p9, fit.cluster2==c))
  z1 <- subset(p8, fit.cluster2==c)
  med <- vector(mode="numeric", length=c3.1)
  for (cc in 1:c3.1) {
    med[cc] <- median(z1[ , cc])
  }
  z2 <- data.frame(z1[ , 1:c3.1])
  z3 <- cov(z2, use = "everything" )
  md <- mahalanobis(z2, center = med, cov = z3, inverted = TRUE)
  nam4 <- paste("Cluster", c, sep = "")
  z4 <- assign(nam4, data.frame(z, md, row.names=NULL))
  y2 <- nrow(z4)
  row.names(z4) <- 1:y2
  df.2 <- c(1:y2)
  if (y2 >= b2) {
    Out.999 = quantile(md, .999)
    plot(df.2, md, xlab="Object", ylab="Mahalanobis Distance", main=c("Cluster 1,
Subcluster",c,"Outlier Plot"))
    abline(h=Out.999, col="red", lwd=2)
    cat("The following rows in cluster 1, sub-cluster",c,"data appear very
suspicious:",which(md>Out.999),"\n")
    qqplot(md, df.2, xlab="Mahalanobis Distance", ylab="Ranked Observation",
main=c("Cluster 1, Subcluster",c,"QQ Plot"))
    abline(v=Out.999, col="red", lwd=2)
  }
  if (y2 %in% c(3:b2)) {
    Out.999 = quantile(md, .999)
    plot(df.2, md, xlab="Object", ylab="Mahalanobis Distance", main=c("Cluster 1,
Subcluster",c,"Outlier Plot"))
    abline(h=Out.999, col="red", lwd=2)
    cat("The following rows in cluster 1, sub-cluster",c,"data appear very
suspicious:",which(md>Out.999),"\n")
    qqplot(md, df.2, xlab="Mahalanobis Distance", ylab="Ranked Observation",
main=c("Cluster 1, Subcluster",c,"QQ Plot"))
    abline(v=Out.999, col="red", lwd=2)
    cat("Cluster 1, sub-cluster",c,"has small relative membership and this warrants
investigation of all associated objects","\n")
  }
  if (y2 < 3) {
    cat("Cluster 1, sub-cluster",c,"has an extremely small membership and this
clearly warrants investigation of all associated objects","\n")
  }
}

#Export processed cluster 1 information, segregated by sub-clusters
write.csv(Cluster1, row.names=TRUE, "K:/Itau/Credit
Card/Paul/Dissertation/ClusteringStageTwo/Cluster1.1TestResults.csv")
```

```
write.csv(Cluster2, row.names=TRUE, "K:/Itau/Credit
Card/Paul/Dissertation/ClusteringStageTwo/Cluster1.2TestResults.csv")

#Export Cluster1DataPC and TestModel
write.csv(Cluster1TestDataPC, row.names=TRUE, "K:/Itau/Credit
Card/Paul/Dissertation/ClusteringStageTwo/Cluster1TestDataPC.csv")
write.csv(TestModel, row.names=TRUE, "K:/Itau/Credit
Card/Paul/Dissertation/ClusteringStageTwo/Cluster1SubclusterTestModel.csv")


#CLUSTER 2 - BEGIN STAGE 2 PREPROCESSING AND ANALYSIS
#Create data frame of normalized original variables for performing principal
components analysis
x2 <- data.frame(cluster2[ , c("Record", "AccountAge", "CreditLimit",
"AdditionalAssets", "LatePayments")], row.names=NULL)
View(x2)

#Normalize four dimensions to be clustered
#Assumes dimensions to be normalized are in columns 2, 3, 4, and 5 of data
#Normalize AccountAge column
aa2 <- data.frame(x2[ , 2])
names(aa2)[1] <- "AccountAge"
View(aa2)
maxaa <- max(aa2)
minaa <- min(aa2)
aan <- vector(mode="numeric", length=n2)
i<-1
while(i<n2+1) {
  aan[i] <- ((aa2[i, 1] - minaa)/(maxaa-minaa))
  i <- i+1
}

#Check/verify max & min normalized AccountAge column values (max=1, min=0)
View(max(aan))
View(min(aan))

#Add normalized AccountAge column to data frame x2
x2["AccountAgeN"] <- aan
View(x2)

#Normalize CreditLimit column
cl2 <- data.frame(x2[ , 3])
names(cl2)[1] <- "CreditLimit"
View(cl2)
maxcl <- max(cl2)
View(maxcl)
```

```
mincl <- min(cl2)
View(mincl)
cln <- vector(mode="numeric", length=n2)
i<-1
while(i<n2+1) {
  cln[i] <- ((cl2[i, 1] - mincl)/(maxcl-mincl))
  i <- i+1
}

#Check/verify max & min normalized CreditLmit column values (max=1, min=0)
View(max(cln))
View(min(cln))

#Add normalized CreditLimit column to data frame x2
x2["CreditLimitN"] <- cln
View(x2)

#Normalize AdditionalAssets column
adda2 <- data.frame(x2[ , 4])
names(adda2)[1] <- "AdditionalAssets"
View(adda2)
maxadda <- max(adda2)
View(maxadda)
minadda <- min(adda2)
View(minadda)
addan <- vector(mode="numeric", length=n2)
i<-1
while(i<n2+1) {
  addan[i] <- ((adda2[i, 1] - minadda)/(maxadda-minadda))
  i <- i+1
}

#Check/verify max & min normalized AdditionalAssets column values (max=1,
min=0)
View(max(addan))
View(min(addan))

#Add normalized AdditionalAssets column to data frame x2
x2["AdditionalAssetsN"] <- addan
View(x2)

#Normalize LatePayments column
lp2 <- data.frame(x2[ , 5])
names(lp2)[1] <- "LatePayments"
View(lp2)
maxlp <- max(lp2)
```

```
View(maxlp)
minlp <- min(lp2)
View(minlp)
lpn <- vector(mode="numeric", length=n2)
i<-1
while(i<n2+1) {
  lpn[i] <- ((lp2[i, 1] - minlp)/(maxlp-minlp))
  i <- i+1
}

#Check/verify max & min normalized LatePayments column values (max=1,
min=0)
View(max(lpn))
View(min(lpn))

#Add normalized LatePayments column to data frame x2
x2["LatePaymentsN"] <- lpn
View(x2)

#Create data frame of normalized original variables for performing principal
components analysis
grp2 <- data.frame(x2[ , c("AccountAgeN", "CreditLimitN", "AdditionalAssetsN",
"LatePaymentsN")])
View(grp2)

#Perform principal component analysis and show associated information
trans <- prcomp(grp2, center=TRUE, scale.=TRUE)
print(trans)
summary(trans)
plot(trans, typ="l", col="2", lwd="2")

PC = predict(trans, grp2)
print(PC[1:10, ])
View(PC)
plot(PC, type="p", col="2")
plot(PC, type="l", col="2")
plot(PC, type="b", col="2")

#Create a data frame of x2 and principal components, thus creating x2.pc
x2.pc <- cbind(x2, PC, row.names=NULL)
View(x2.pc)

#Normalize all four principal components
#Assumes dimensions to be normalized are in columns 10, 11, 12, and 13 of
data
#Normalize PC1 column
```

```
pc1 <- x2.pc[[10]]
View(pc1)
maxpc1 <- max(pc1)
View(maxpc1)
minpc1 <- min(pc1)
View(minpc1)
y <- n2
View(y)
pc1n <- vector(mode="numeric", length=y)
View(pc1n)
i<-1
while(i<y+1) {
  pc1n[i] <- ((pc1[i] - minpc1)/(maxpc1-minpc1))
  i <- i+1
}
View(pc1n)

#Check/verify max & min normalized PC1 column values (max=1, min=0)
View(max(pc1n))
View(min(pc1n))

#Add normalized PC1 column to data frame x2.pc
x2.pc["PC1N"] <- pc1n
View(x2.pc)

#Normalize PC2 column
pc2 <- x2.pc[[11]]
View(pc2)
maxpc2 <- max(pc2)
View(maxpc2)
minpc2 <- min(pc2)
View(minpc2)
pc2n <- vector(mode="numeric", length=y)
i<-1
while(i<y+1) {
  pc2n[i] <- ((pc2[i] - minpc2)/(maxpc2-minpc2))
  i <- i+1
}
View(pc2n)

#Check/verify max & min normalized PC2 column values (max=1, min=0)
View(max(pc2n))
View(min(pc2n))

#Add normalized PC2 column to data frame x2.pc
x2.pc["PC2N"] <- pc2n
```

```
View(x2.pc)

#Normalize PC3 column
pc3 <- x2.pc[[12]]
View(pc3)
maxpc3 <- max(pc3)
View(maxpc3)
minpc3 <- min(pc3)
View(minpc3)
pc3n <- vector(mode="numeric", length=y)
i<-1
while(i<y+1) {
  pc3n[i] <- ((pc3[i] - minpc3)/(maxpc3-minpc3))
  i <- i+1
}
View(pc3n)

#Check/verify max & min normalized PC3 column values (max=1, min=0)
View(max(pc3n))
View(min(pc3n))

#Add normalized PC3 column to data frame x2.pc
x2.pc["PC3N"] <- pc3n
View(x2.pc)

#Normalize PC4 column
pc4 <- x2.pc[[13]]
View(pc4)
maxpc4 <- max(pc4)
View(maxpc4)
minpc4 <- min(pc4)
View(minpc4)
pc4n <- vector(mode="numeric", length=y)
i<-1
while(i<y+1) {
  pc4n[i] <- ((pc4[i] - minpc4)/(maxpc4-minpc4))
  i <- i+1
}
View(pc4n)

#Check/verify max & min normalized PC4 column values (max=1, min=0)
View(max(pc4n))
View(min(pc4n))

#Add normalized PC4 column to data frame x2.pc
x2.pc["PC4N"] <- pc4n
```

```
View(x2.pc)

#Export x2.pc for subsequent model building (pre-processed cluster 2 data)
write.csv(x2.pc, "K:/Itau/Credit
Card/Paul/Dissertation/ClusteringStageTwo/Cluster2Data.csv")

#Read x2.pc data (if not currently loaded)
x2.pc <- read.csv("K:/Itau/Credit
Card/Paul/Dissertation/ClusteringStageTwo/Cluster2Data.csv")
View(x2.pc)

#Begin silhouette analysis for Cluster 2
#Cluster 2 contains less than 40,000 objects (no sample needed)
#Create a data frame Cluster2PC of 3 top PCs from x2.pc for computing
silhouette coeffients
c <- ncol(x2.pc)
c1 <- c-3
c2 <- c-1
Cluster2PC <- data.frame(x2.pc[ , c1:c2])
View(Cluster2PC)

#Execute Kmeans clustering model performance sequence
#Use loop structure for initial Kmeans model building process
nclust=6
x=matrix(nrow=nclust-1, ncol=2)
d <- dist(Cluster2PC, "euclidean")
y <- 1
c <- 2
for (c in c:nclust) {
  fit <- kmeans(Cluster2PC, c, iter.max=500, nstart=10, algorithm="Lloyd")
  sil <- silhouette(fit$cluster, d)
  avgsil <- summary(sil, FUN=mean) $avg.width
  x[y, ] <- c(c, avgsil)
  y<-y+1
}
View(x)
max(x[ ,2])

#Create data frame z, Sort in descending order by Silhouette Coefficient
z <- data.frame(x)
names(z)[1] <- "Clusters"
names(z)[2] <- "Silhouette"
View(z)

s <- arrange(z, desc(Silhouette))
View(s)
```

```
#Export processed data
write.csv(s, "K:/Itau/Credit
Card/Paul/Dissertation/ClusteringStageTwo/Cluster2SCKmeans.csv")


#Execute PAM clustering model performance sequence
#Use loop structure for initial model building process
nclust= 6
x=matrix(nrow=nclust-1, ncol=2)
d <- dist(Cluster2PC, method = "euclidean")
y <- 1
c <- 2
for (c in c:nclust) {
  fit <- pam(Cluster2PC, c)
  sil <- silhouette(fit$clustering, d)
  avgsil <- summary(sil, FUN=mean) $avg.width
  x[y, ] <- c(c, avgsil)
  y<-y+1
}
View(x)
max(x[ ,2])

#Create data frame z, Sort in descending order by Silhouette Coefficient
z <- data.frame(x)
names(z)[1] <- "Clusters"
names(z)[2] <- "Silhouette"
View(z)

s <- arrange(z, desc(Silhouette))
View(s)

#Export processed data
write.csv(s, "K:/Itau/Credit
Card/Paul/Dissertation/ClusteringStageTwo/Cluster2SCPAM.csv")


#Execute Complete-Link Hierarchical clustering model performance sequence
#Use loop structure for initial model building process
nclust= 6
x=matrix(nrow=nclust-1, ncol=2)
d <- dist(Cluster2PC, method = "euclidean")
y <- 1
c <- 2
for (c in c:nclust) {
  fit2 <- hclust(d, method = "complete")
  memb <- cutree(fit2, k=c)
  fit$clusters <- assignCluster(Cluster2PC, x2.pc, memb)
  fit$cluster <- as.integer(fit$clusters)
  sil <- silhouette(fit$cluster, d)
  avgsil <- summary(sil, FUN=mean) $avg.width
  x[y, ] <- c(c, avgsil)
  y<-y+1
}
View(x)
max(x[ ,2])

#Create data frame z, Sort in descending order by Silhouette Coefficient
z <- data.frame(x)
names(z)[1] <- "Clusters"
names(z)[2] <- "Silhouette"
View(z)

s <- arrange(z, desc(Silhouette))
View(s)

#Export processed data
write.csv(s, "K:/Itau/Credit
Card/Paul/Dissertation/ClusteringStageTwo/Cluster2SCComplete.csv")


#Execute Ward's Method Hierarchical clustering model performance sequence
#Use loop structure for initial model building process
nclust= 6
x=matrix(nrow=nclust-1, ncol=2)
d <- dist(Cluster2PC, method = "euclidean")
y <- 1
c <- 2
for (c in c:nclust) {
  fit2 <- hclust(d, method = "ward.D2")
  memb <- cutree(fit2, k=c)
  fit$clusters <- assignCluster(Cluster2PC, x2.pc, memb)
  fit$cluster <- as.integer(fit$clusters)
  sil <- silhouette(fit$cluster, d)
  avgsil <- summary(sil, FUN=mean) $avg.width
  x[y, ] <- c(c, avgsil)
  y<-y+1
}
View(x)
max(x[ ,2])

#Create data frame z, Sort in descending order by Silhouette Coefficient
```

```
z <- data.frame(x)
names(z)[1] <- "Clusters"
names(z)[2] <- "Silhouette"
View(z)

s <- arrange(z, desc(Silhouette))
View(s)

#Export processed data
write.csv(s, "K:/Itau/Credit
Card/Paul/Dissertation/ClusteringStageTwo/Cluster2SCWard.csv")


#Execute Expectation Maximization clustering model performance sequence
#Use loop structure for initial EM model building process
nclust=8
x=matrix(nrow=nclust-1, ncol=2)
d <- dist(Cluster2PC, "euclidean")
y <- 1
c <- 2
for (c in c:nclust) {
  fit <- init.EM(Cluster2PC, nclass = c, EMC = .EMC.Rnd, stable.solution = TRUE,
min.n = 1, min.n.iter = 10, method = "Rnd.EM")
  sil <- silhouette(fit$class, d)
  avgsil <- summary(sil, FUN=mean) $avg.width
  x[y, ] <- c(c, avgsil)
  y<-y+1
}
View(x)
max(x[ ,2])

#Create data frame z, Sort in descending order by Silhouette Coefficient
z <- data.frame(x)
names(z)[1] <- "Clusters"
names(z)[2] <- "Silhouette"
View(z)

s <- arrange(z, desc(Silhouette))
View(s)

#Export processed data
write.csv(s, "K:/Itau/Credit
Card/Paul/Dissertation/ClusteringStageTwo/Cluster2SCEM.csv")


#Build final cluster 2 model - based upon silhouette results
```

```
#Decision 7/29/15:  Create Complete-Link 2-cluster model (set c=2)
c <- 2

#Read x2.pc data (if not already loaded)
x2.pc <- read.csv("K:/Itau/Credit
Card/Paul/Dissertation/ClusteringStageTwo/Cluster2Data.csv")
View(x2.pc)

#Create data frame for clustering
c3 <- ncol(x2.pc)
c4 <- c3-3
c5 <- c3-1
Cluster2PC <- data.frame(x2.pc[ , c4:c5])
View(Cluster2PC)

#Generate cluster 2 sub-cluster model
d <- dist(Cluster2PC, method = "euclidean")
fit2 <- hclust(d, method = "complete")
memb <- cutree(fit2, k=c)

#Append cluster assignment column
fit$clusters <- assignCluster(Cluster2PC, x2.pc, memb)
fit$cluster <- as.integer(fit$clusters)
Cluster2Model <- data.frame(x2.pc, fit$cluster)
View(Cluster2Model)

#3D Plots of top 3 principal components with cluster assignments depicted
p2 <- data.frame(Cluster2PC)
ddd <- data.frame(p2, Cluster2Model[ , "fit.cluster"])
names(ddd)[1] <- "PC1"
names(ddd)[2] <- "PC2"
names(ddd)[3] <- "PC3"
names(ddd)[4] <- "Cluster"
scatterplot3d(ddd, col.axis = "green", col.grid = "green", main = "Principal
Components Scatterplot", angle=50, pch = 1)
scatterplot3d(ddd, col.axis = "green", col.grid = "green", main = "Principal
Components Scatterplot", angle=140, pch = 1)
scatterplot3d(ddd, col.axis = "green", col.grid = "green", main = "Principal
Components Scatterplot", angle=230, pch = 1)
scatterplot3d(ddd, col.axis = "green", col.grid = "green", main = "Principal
Components Scatterplot", angle=340, pch = 1)

#Generate data frames for subsequent analysis
x3 <- data.frame(Cluster2PC)
x4 <- data.frame(Cluster2Model[ , 6:9])
x5 <- data.frame(Cluster2Model[ , 2:5])
```

```
#Get cluster means for normalized pcs, normalized original variables, and
original variables
aggregate(x3,by=list(fit$cluster),FUN=mean)
aggregate(x4, by=list(fit$cluster), FUN=mean)
aggregate(x5, by=list(fit$cluster), FUN=mean)

#Get cluster medians for normalized pcs, normalized original variables, and
original variables
aggregate(x3,by=list(fit$cluster),FUN=median)
aggregate(x4, by=list(fit$cluster), FUN=median)
aggregate(x5, by=list(fit$cluster), FUN=median)

#Get sub-cluster counts
c6 <- ncol(Cluster2Model)
count1 <- count(Cluster2Model[ , c6]==1)
View(count1)
count2 <- count(Cluster2Model[ , c6]==2)
View(count2)

#Perform statistical testing on population
#generate data frame of normalized principal components for statistical testing
names(Cluster2Model)[ncol(Cluster2Model)] <- "fit.cluster"
ss1 <- data.frame(Cluster2Model[ ,c("PC1N","PC2N","PC3N","fit.cluster")])
View(ss1)

#perform Kruskal-Wallis tests of normalized Principal components
kruskal.test(PC1N~fit.cluster, data=ss1)
kruskal.test(PC2N~fit.cluster, data=ss1)
kruskal.test(PC3N~fit.cluster, data=ss1)
kruskal.test(PC1N+PC2N+PC3N~fit.cluster, data=ss1)

#generate data frame of normalized original variables for statistical testing
ss2 <- data.frame(Cluster2Model[
,c("AccountAgeN","CreditLimitN","AdditionalAssetsN","LatePaymentsN","fit.cluste
r")])
View(ss2)

#perform Kruskal-Wallis tests of normalized original variables
kruskal.test(AccountAgeN~fit.cluster, data=ss2)
kruskal.test(CreditLimitN~fit.cluster, data=ss2)
kruskal.test(AdditionalAssetsN~fit.cluster, data=ss2)
kruskal.test(LatePaymentsN~fit.cluster, data=ss2)
kruskal.test(AccountAgeN+CreditLimitN+AdditionalAssetsN+LatePaymentsN~fit.
cluster, data=ss2)
```

```
#generate data frame of original variables for statistical testing
ss3 <- data.frame(Cluster2Model[
,c("AccountAge","CreditLimit","AdditionalAssets","LatePayments","fit.cluster")])
View(ss3)

#perform Kruskal-Wallis tests of original variables
kruskal.test(AccountAge~fit.cluster, data=ss3)
kruskal.test(CreditLimit~fit.cluster, data=ss3)
kruskal.test(AdditionalAssets~fit.cluster, data=ss3)
kruskal.test(LatePayments~fit.cluster, data=ss3)
kruskal.test(AccountAge+CreditLimit+AdditionalAssets+LatePayments~fit.cluster,
data=ss3)

#Get a representative sample of Cluster2Model data for statistical analysis
s <- arrange(Cluster2Model,fit.cluster)
View(s)
c13 <- ncol(Cluster2Model)
s1 <- data.frame(s[ , 1:c13])
View(s1)

r <- nrow(s1)
col <- ncol(s1)
sc=3944
samplepct=round(sc/r, 2)
step=round(r/sc,0)
z=matrix(nrow=r*samplepct, ncol=col)
y <- 1
i <- 1
while (i < r) {
  z[y, ] <- as.matrix(s1[i, 1:c13])
  y<-y+1
  i<-i+step
}
View(z)
s2 <- data.frame(z[ , 1:c13])
View(s2)

#Attach appropriate column names to data frame s2
names(s2)[1] <- "Record"
names(s2)[2] <- "AccountAge"
names(s2)[3] <- "CreditLimit"
names(s2)[4] <- "AdditionalAssets"
names(s2)[5] <- "LatePayments"
names(s2)[6] <- "AccountAgeN"
names(s2)[7] <- "CreditLimitN"
names(s2)[8] <- "AdditionalAssetsN"
names(s2)[9] <- "LatePaymentsN"
names(s2)[10] <- "PC1"
names(s2)[11] <- "PC2"
names(s2)[12] <- "PC3"
names(s2)[13] <- "PC4"
names(s2)[14] <- "PC1N"
names(s2)[15] <- "PC2N"
names(s2)[16] <- "PC3N"
names(s2)[17] <- "PC4N"
names(s2)[18] <- "fit.cluster"
s3 <- arrange(s2,Record)
View(s3)

#Perform statistical testing on representative sample
#generate data frame of normalized principal components for statistical testing
ss1 <- data.frame(s3[ ,c("PC1N","PC2N","PC3N","fit.cluster")])
View(ss1)

#perform Kruskal-Wallis tests of normalized Principal components
kruskal.test(PC1N~fit.cluster, data=ss1)
kruskal.test(PC2N~fit.cluster, data=ss1)
kruskal.test(PC3N~fit.cluster, data=ss1)
kruskal.test(PC1N+PC2N+PC3N~fit.cluster, data=ss1)

#generate data frame of normalized original variables for statistical testing
ss2 <- data.frame(s3[
,c("AccountAgeN","CreditLimitN","AdditionalAssetsN","LatePaymentsN","fit.cluste
r")])
View(ss2)

#perform Kruskal-Wallis tests of normalized original variables
kruskal.test(AccountAgeN~fit.cluster, data=ss2)
kruskal.test(CreditLimitN~fit.cluster, data=ss2)
kruskal.test(AdditionalAssetsN~fit.cluster, data=ss2)
kruskal.test(LatePaymentsN~fit.cluster, data=ss2)
kruskal.test(AccountAgeN+CreditLimitN+AdditionalAssetsN+LatePaymentsN~fit.
cluster, data=ss2)

#generate data frame of original variables for statistical testing
ss3 <- data.frame(s3[
,c("AccountAge","CreditLimit","AdditionalAssets","LatePayments","fit.cluster")])
View(ss3)

#perform Kruskal-Wallis tests of original variables
kruskal.test(AccountAge~fit.cluster, data=ss3)
kruskal.test(CreditLimit~fit.cluster, data=ss3)
kruskal.test(AdditionalAssets~fit.cluster, data=ss3)
kruskal.test(LatePayments~fit.cluster, data=ss3)
kruskal.test(AccountAge+CreditLimit+AdditionalAssets+LatePayments~fit.cluster,
data=ss3)


#Cluster 2 - Sub-Cluster OUtlier Detection Via Mahalanobis Distance
#Set initial variables for subsequent usage
r2 <- nrow(Cluster2Model)
c6 <- ncol(Cluster2PC)
c7 <- ncol(Cluster2Model)
names(Cluster2Model)[c7] <- "fit.cluster"
b1 <- r2*.01
x6 <- max(Cluster2Model[ , "fit.cluster"])
p3 <- data.frame(Cluster2PC, Cluster2Model[ , c7])
names(p3)[c6+1] <- "fit.cluster2"
p4 <- data.frame(arrange(p3, fit.cluster2), row.names=NULL)
p5 <- data.frame(arrange(Cluster2Model, fit.cluster), row.names=NULL)

#Separate data by cluster, compute medians, calculate mahalanobis distances,
and create outlier plots for each cluster
for (c in 1:x6) {
  nam1 <- paste("Cluster", c, sep = "")
  z <- assign(nam1, subset(p5, fit.cluster==c))
  z1 <- subset(p4, fit.cluster2==c)
  med <- vector(mode="numeric", length=c6)
  for (cc in 1:c6) {
    med[cc] <- median(z1[ , cc])
  }
  z2 <- data.frame(z1[ , 1:c6])
  z3 <- cov(z2, use = "everything" )
  md <- mahalanobis(z2, center = med, cov = z3, inverted = TRUE)
  nam4 <- paste("Cluster", c, sep = "")
  z4 <- assign(nam4, data.frame(z, md, row.names=NULL))
  y2 <- nrow(z4)
  row.names(z4) <- 1:y2
  df.2 <- c(1:y2)
  if (y2 >= b1) {
    Out.999 = quantile(md, .999)
    plot(df.2, md, xlab="Object", ylab="Mahalanobis Distance", main=c("Cluster 2,
Subcluster",c,"Outlier Plot"))
    abline(h=Out.999, col="red", lwd=2)
    cat("The following rows in cluster 2, sub-cluster",c,"data appear very
suspicious:",which(md>Out.999),"\n")
    qqplot(md, df.2, xlab="Mahalanobis Distance", ylab="Ranked Observation",
main=c("Cluster 2, Subcluster",c,"QQ Plot"))
```

```
    abline(v=Out.999, col="red", lwd=2)
  }
  if (y2 %in% c(3:b1)) {
    Out.999 = quantile(md, .999)
    plot(df.2, md, xlab="Object", ylab="Mahalanobis Distance", main=c("Cluster 2,
Subcluster",c,"Outlier Plot"))
    abline(h=Out.999, col="red", lwd=2)
    cat("The following rows in cluster 2, sub-cluster",c,"data appear very
suspicious:",which(md>Out.999),"\n")
    qqplot(md, df.2, xlab="Mahalanobis Distance", ylab="Ranked Observation",
main=c("Cluster 2, Subcluster",c,"QQ Plot"))
    abline(v=Out.999, col="red", lwd=2)
    cat("Cluster 2, sub-cluster",c,"has small relative membership and this warrants
investigation of all associated objects","\n")
  }
  if (y2 < 3) {
    cat("Cluster 2, sub-cluster",c,"has an extremely small membership and this
clearly warrants investigation of all associated objects","\n")
  }
}

#Export processed information, segregated by cluster
write.csv(Cluster1, row.names=TRUE, "K:/Itau/Credit
Card/Paul/Dissertation/ClusteringStageTwo/Cluster2.1Results.csv")
write.csv(Cluster2, row.names=TRUE, "K:/Itau/Credit
Card/Paul/Dissertation/ClusteringStageTwo/Cluster2.2Results.csv")

#Export Cluster2PC and Cluster2Model
write.csv(Cluster2PC, row.names=TRUE, "K:/Itau/Credit
Card/Paul/Dissertation/ClusteringStageTwo/Cluster2PC.csv")
write.csv(Cluster2Model, row.names=TRUE, "K:/Itau/Credit
Card/Paul/Dissertation/ClusteringStageTwo/Cluster2Model.csv")
```

APPENDIX C: RSCRIPT - CHAPTER 4 OUTLIER DETECTION

```
#Chapter 3 - Outlier Detection
#Calculates distances/dissmilarities and outlier scores and locates anomalies
#Code supports sequential analysis of four clusters (1, 2.1, 2.2, and 3 in this
case)

#install necessary packages
library(knitr)
library(stats)
library(chemometrics)
library(fields)

#Read Cluster1.2Final data, rename, and read number of rows
Cluster1 <- read.csv("K:/Itau/Credit
Card/Paul/Dissertation/ClusteringStageOne/Cluster1.csv")
View(Cluster1)
x1 <- Cluster1
y <- nrow(x1)

#Create new data frames for outlier detection purposes
df.1 <- data.frame(x1[ , c("PC1N","PC2N","PC3N")], row.names=NULL)
View(df.1)
df.2 <- c(1:y)
View(df.2)

#Compute medians of 3 relevant normalized principal components
#Entails calculation of median for the column vector corresponding to each
dimension
d1 <- df.1[[1]]
m1 <- median(d1)
View(m1)

d2 <- df.1[[2]]
m2 <- median(d2)
View(m2)

d3 <- df.1[[3]]
m3 <- median(d3)
View(m3)

#Compute Mahalanobis Distances
z <- cov(df.1, use = "everything")
md <- mahalanobis(df.1, center = c(m1, m2, m3), cov = z, inverted=TRUE)
```

```
#Add md column to data set x1
x1["MahalanobisDistance"] <- md

#Normalize md column
mah1 <- x1[[18]]
max1 <- max(mah1)
i <- 1
mdn <- vector(mode="numeric", length=y)
while (i<y+1) {
  mdn[i] <- mah1[i]/max1
  i<-i+1
}

#Check/verify max & min normalized values (max=1, min>=0)
maxmdn <- max(mdn)
View(maxmdn)
minmdn <- min(mdn)
View(minmdn)

#Add normalized md column to data set x1
x1["MahalanobisNorm"] <- mdn

#Cluster 1 Data - Mahalanobis Plots/Visuals
C1.995 = quantile(mdn, .995)
plot(df.2, mdn, xlab="Object", ylab="Mahalanobis Distance", main="Cluster 1
Outlier Plot")
abline(h=C1.995, col="red", lwd=2)
which(mdn>C1.995)

qqplot(mdn, df.2, xlab="Mahalanobis Distance", main="Cluster 1 - QQ Plot")
abline(v=C1.995, col="red", lwd=2)

#Compute Euclidean distances
med <- rbind(c(m1,m2,m3))
ed <- rdist(df.1, med)

#Add ed column to data set x1
x1["EuclideanDistance"] <- ed

#Normalize ed column
euc1 <- x1[[20]]
max2 <- max(euc1)
i <- 1
edn <- vector(mode="numeric", length=y)
while (i<y+1) {
  edn[i] <- euc1[i]/max2
  i<-i+1
}

#Check/verify max & min normalized values (max=1, min>=0)
maxedn <- max(edn)
View(maxedn)
minedn <- min(edn)
View(minedn)

#Add normalized ed column to data set x1
x1["EuclideanNorm"] <- edn

#Cluster 1 Data - Euclidean Plots/Visuals
C1.995 = quantile(edn, .995)
plot(df.2, edn, xlab="Object", ylab="Euclidean Distance", main="Cluster 1 Outlier
Plot")
abline(h=C1.995, col="red", lwd=2)
which(edn>C1.995)

qqplot(edn, df.2, xlab="Euclidean Distance", main="Cluster 1 - QQ Plot")
abline(v=C1.995, col="red", lwd=2)

#Compute Cosine Dissimilarities (1 - cosine similarity)
cd=vector(mode="numeric", length=y)
i<-1
while(i<y+1) {
  cd[i] <- 1-
(((df.1[i,1]*m1)+(df.1[i,2]*m2)+(df.1[i,3]*m3))/((sqrt(df.1[i,1]^2+df.1[i,2]^2+df.1[i,3]^
2))*(sqrt(m1^2+m2^2+m3^2))))
  i<-i+1
}

#Add cd column to data set x1
x1["CosineDissimilarity"] <- cd

#Normalize cd column
cos1 <- x1[[22]]
max3 <- max(cos1)
i <- 1
cdn <- vector(mode="numeric", length=y)
while (i<y+1) {
  cdn[i] <- cos1[i]/max3
  i<-i+1
}
```

```
#Check/verify max & min normalized values (max=1, min>=0)
maxcdn <- max(cdn)
View(maxcdn)
mincdn <- min(cdn)
View(mincdn)

#Add normalized cd column to data set x1
x1["CosineDissimilarityNorm"] <- cdn

#Cluster 1 Data - Cosine Plots/Visuals
C1.995 = quantile(cdn, .995)
plot(df.2, cdn, xlab="Object", ylab="Cosine Dissimilarity", main="Cluster 1 Outlier
Plot")
abline(h=C1.995, col="red", lwd=2)
which(cdn>C1.995)

qqplot(cdn, df.2, xlab="Cosine Dissimilarity", main="Cluster 1 - QQ Plot")
abline(v=C1.995, col="red", lwd=2)

#Compute Tanimoto Dissimilarities (1 - Tanimoto Coefficient)
td <- vector(mode="numeric", length=y)
i<-1
while(i<y+1){
  td[i] <- 1-
((((df.1[i,1]*m1)+(df.1[i,2]*m2)+(df.1[i,3]*m3))/(((sqrt(df.1[i,1]^2+df.1[i,2]^2+df.1[i,3
]^2))^2)+((sqrt(m1^2+m2^2+m3^2))^2)-
((df.1[i,1]*m1)+(df.1[i,2]*m2)+(df.1[i,3]*m3)))))
  i<-i+1
}

#Add td column to data set x1
x1["TanimotoDissimilarity"] <- td

#Normalize td column
tan1 <- x1[[24]]
max4 <- max(tan1)
i <- 1
tdn <- vector(mode="numeric", length=y)
while (i<y+1) {
  tdn[i] <- tan1[i]/max4
  i<-i+1
}

#Check/verify max & min normalized values (max=1, min>=0)
maxtdn <- max(tdn)
View(maxtdn)
```

```
mintdn <- min(tdn)
View(mintdn)

#Add normalized td column to data set x1
x1["TanimotoDissimilarityNorm"] <- tdn

#Cluster 1 Data - Tanimoto Plots/Visuals
C1.995 = quantile(tdn, .995)
plot(df.2, tdn, xlab="Object", ylab="Tanimoto Dissimilarity", main="Cluster 1
Outlier Plot")
abline(h=C1.995, col="red", lwd=2)
which(tdn>C1.995)

qqplot(tdn, df.2, xlab="Tanimoto Dissimilarity", main="Cluster 1 - QQ Plot")
abline(v=C1.995, col="red", lwd=2)

#Calculate Outlier Scores from all 4 normalized values
#Includes weight parameters w1 to w4 for each metric (default value is 1 -- sum
of weights = 4)
w1 <- 1
w2 <- 1
w3 <- 1
w4 <- 1
df.3 <- x1[18:25]
View(df.3)
os <- vector(mode="numeric", length=y)
i<-1
while(i<y+1){
  os[i] <- ((df.3[i,2]*w1)+(df.3[i,4]*w2)+(df.3[i,6]*w3)+(df.3[i,8]*w4))
  i<-i+1
}

#Add outlier score column to data set x1
x1["OutlierScore"] <- os
View(x1)

#View max and min outlier scores (max<=4, min>0)
maxos <- max(os)
View(maxos)
minos <- min(os)
View(minos)

#Cluster 1 Data - Outlier Score Plots/Visuals
C1.995 = quantile(os, .995)
plot(df.2, os, xlab="Object", ylab="Outlier Score", main="Cluster 1 Outlier Plot")
abline(h=C1.995, col="red", lwd=2)
```

```
which(os>C1.995)

qqplot(os, df.2, xlab="Outlier Score", main="Cluster 1 - QQ Plot")
abline(v=C1.995, col="red", lwd=2)

#Export processed info as csv file on hard drive
write.csv(x1, "K:/Itau/Credit
Card/Paul/Dissertation/OutlierDetectionStageThree/Cluster1Outliers.csv")



#Read Cluster2.1 data, rename, and read number of rows
Cluster2.1 <- read.csv("K:/Itau/Credit
Card/Paul/Dissertation/ClusteringStageTwo/Cluster2.1.csv")
View(Cluster2.1)
x1 <- Cluster2.1
y <- nrow(x1)

#Create new data frames for outlier detection purposes
df.1 <- data.frame(x1[ , c("PC1N","PC2N","PC3N")], row.names=NULL)
View(df.1)
df.2 <- c(1:y)
View(df.2)

#Compute medians of 3 relevant normalized principal components
#Entails calculation of median for the column vector corresponding to each
dimension
d1 <- df.1[[1]]
m1 <- median(d1)
View(m1)

d2 <- df.1[[2]]
m2 <- median(d2)
View(m2)

d3 <- df.1[[3]]
m3 <- median(d3)
View(m3)

#Compute Mahalanobis Distances
z <- cov(df.1, use = "everything")
md <- mahalanobis(df.1, center = c(m1, m2, m3), cov = z, inverted=TRUE)

#Add md column to data set x1
x1["MahalanobisDistance"] <- md
```

```
#Normalize md column
mah1 <- x1[[18]]
max1 <- max(mah1)
i <- 1
mdn <- vector(mode="numeric", length=y)
while (i<y+1) {
  mdn[i] <- mah1[i]/max1
  i<-i+1
}

#Check/verify max & min normalized values (max=1, min>=0)
maxmdn <- max(mdn)
View(maxmdn)
minmdn <- min(mdn)
View(minmdn)

#Add normalized md column to data set x1
x1["MahalanobisNorm"] <- mdn

#Cluster 2.1 Data - Mahalanobis Plots/Visuals
C1.995 = quantile(mdn, .995)
plot(df.2, mdn, xlab="Object", ylab="Mahalanobis Distance", main="Cluster 2.1
Outlier Plot")
abline(h=C1.995, col="red", lwd=2)
which(mdn>C1.995)

qqplot(mdn, df.2, xlab="Mahalanobis Distance", main="Cluster 2.1 - QQ Plot")
abline(v=C1.995, col="red", lwd=2)

#Compute Euclidean distances
med <- rbind(c(m1,m2,m3))
ed <- rdist(df.1, med)

#Add ed column to data set x1
x1["EuclideanDistance"] <- ed

#Normalize ed column
euc1 <- x1[[20]]
max2 <- max(euc1)
i <- 1
edn <- vector(mode="numeric", length=y)
while (i<y+1) {
  edn[i] <- euc1[i]/max2
  i<-i+1
}
```

```
#Check/verify max & min normalized values (max=1, min>=0)
maxedn <- max(edn)
View(maxedn)
minedn <- min(edn)
View(minedn)

#Add normalized ed column to data set x1
x1["EuclideanNorm"] <- edn

#Cluster 2.1 Data - Euclidean Plots/Visuals
C1.995 = quantile(edn, .995)
plot(df.2, edn, xlab="Object", ylab="Euclidean Distance", main="Cluster 2.1
Outlier Plot")
abline(h=C1.995, col="red", lwd=2)
which(edn>C1.995)

qqplot(edn, df.2, xlab="Euclidean Distance", main="Cluster 2.1 - QQ Plot")
abline(v=C1.995, col="red", lwd=2)

#Compute Cosine Dissimilarities (1 - cosine similarity)
cd=vector(mode="numeric", length=y)
i<-1
while(i<y+1) {
  cd[i] <- 1-
(((df.1[i,1]*m1)+(df.1[i,2]*m2)+(df.1[i,3]*m3))/((sqrt(df.1[i,1]^2+df.1[i,2]^2+df.1[i,3]^
2))*(sqrt(m1^2+m2^2+m3^2))))
  i<-i+1
}

#Add cd column to data set x1
x1["CosineDissimilarity"] <- cd

#Normalize cd column
cos1 <- x1[[22]]
max3 <- max(cos1)
i <- 1
cdn <- vector(mode="numeric", length=y)
while (i<y+1) {
  cdn[i] <- cos1[i]/max3
  i<-i+1
}

#Check/verify max & min normalized values (max=1, min>=0)
maxcdn <- max(cdn)
View(maxcdn)
mincdn <- min(cdn)
```

```
View(mincdn)

#Add normalized cd column to data set x1
x1["CosineDissimilarityNorm"] <- cdn

#Cluster 2.1 Data - Cosine Plots/Visuals
C1.995 = quantile(cdn, .995)
plot(df.2, cdn, xlab="Object", ylab="Cosine Dissimilarity", main="Cluster 2.1
Outlier Plot")
abline(h=C1.995, col="red", lwd=2)
which(cdn>C1.995)

qqplot(cdn, df.2, xlab="Cosine Dissimilarity", main="Cluster 2.1 - QQ Plot")
abline(v=C1.995, col="red", lwd=2)

#Compute Tanimoto Dissimilarities (1 - Tanimoto Coefficient)
td <- vector(mode="numeric", length=y)
i<-1
while(i<y+1){
  td[i] <- 1-
((((df.1[i,1]*m1)+(df.1[i,2]*m2)+(df.1[i,3]*m3))/(((sqrt(df.1[i,1]^2+df.1[i,2]^2+df.1[i,3
]^2))^2)+((sqrt(m1^2+m2^2+m3^2))^2)-
((df.1[i,1]*m1)+(df.1[i,2]*m2)+(df.1[i,3]*m3)))))
  i<-i+1
}

#Add td column to data set x1
x1["TanimotoDissimilarity"] <- td

#Normalize td column
tan1 <- x1[[24]]
max4 <- max(tan1)
i <- 1
tdn <- vector(mode="numeric", length=y)
while (i<y+1) {
  tdn[i] <- tan1[i]/max4
  i<-i+1
}

#Check/verify max & min normalized values (max=1, min>=0)
maxtdn <- max(tdn)
View(maxtdn)
mintdn <- min(tdn)
View(mintdn)

#Add normalized td column to data set x1
```

```
x1["TanimotoDissimilarityNorm"] <- tdn

#Cluster 2.1 Data - Tanimoto Plots/Visuals
C1.995 = quantile(tdn, .995)
plot(df.2, tdn, xlab="Object", ylab="Tanimoto Dissimilarity", main="Cluster 2.1
Outlier Plot")
abline(h=C1.995, col="red", lwd=2)
which(tdn>C1.995)

qqplot(tdn, df.2, xlab="Tanimoto Dissimilarity", main="Cluster 2.1 - QQ Plot")
abline(v=C1.995, col="red", lwd=2)

#Calculate Outlier Scores from all 4 normalized values
#Includes weight parameters w1 to w4 for each metric (default value is 1 -- sum
of weights = 4)
w1 <- 1
w2 <- 1
w3 <- 1
w4 <- 1
df.3 <- x1[18:25]
View(df.3)
os <- vector(mode="numeric", length=y)
i<-1
while(i<y+1){
  os[i] <- ((df.3[i,2]*w1)+(df.3[i,4]*w2)+(df.3[i,6]*w3)+(df.3[i,8]*w4))
  i<-i+1
}
View(os)

#Add outlier score column to data set x1
x1["OutlierScore"] <- os
View(x1)

#View max and min outlier scores (max<=4, min>0)
maxos <- max(os)
View(maxos)
minos <- min(os)
View(minos)

#Cluster 2.1 Data - Outlier Score Plots/Visuals
C1.995 = quantile(os, .995)
plot(df.2, os, xlab="Object", ylab="Outlier Score", main="Cluster 2.1 Outlier Plot")
abline(h=C1.995, col="red", lwd=2)
which(os>C1.995)

qqplot(os, df.2, xlab="Outlier Score", main="Cluster 2.1 - QQ Plot")
```

```
abline(v=C1.995, col="red", lwd=2)

#Export processed info as csv file on hard drive
write.csv(x1, "K:/Itau/Credit
Card/Paul/Dissertation/OutlierDetectionStageThree/Cluster2.1Outliers.csv")



#Read Cluster2.2 data, rename, and read number of rows
Cluster2.2 <- read.csv("K:/Itau/Credit
Card/Paul/Dissertation/ClusteringStageTwo/Cluster2.2.csv")
View(Cluster2.2)
x2 <- Cluster2.2
y <- nrow(x2)

#Create new data frames for outlier detection
df.1 <- data.frame(x2[ , c("PC1N","PC2N","PC3N")], row.names=NULL)
View(df.1)
df.2 <- c(1:y)

#Compute medians of 3 PCs
#Entails calculation of median for the column vector corresponding to each
dimension
d1 <- df.1[[1]]
m1 <- median(d1)
View(m1)

d2 <- df.1[[2]]
m2 <- median(d2)
View(m2)

d3 <- df.1[[3]]
m3 <- median(d3)
View(m3)

#Compute Mahalanobis Distances
z <- cov(df.1, use = "everything")
md <- mahalanobis(df.1, center = c(m1, m2, m3), cov = z, inverted=TRUE)

#Add md column to data set x2
x2["MahalanobisDistance"] <- md

#Normalize md column
mah1 <- x2[[18]]
max1 <- max(mah1)
i <- 1
```

```
mdn <- vector(mode="numeric", length=y)
while (i<y+1) {
  mdn[i] <- mah1[i]/max1
  i<-i+1
}

#Check/verify max & min normalized values (max=1, min>=0)
maxmdn <- max(mdn)
View(maxmdn)
minmdn <- min(mdn)
View(minmdn)

#Add normalized md column to data set x2
x2["MahalanobisNorm"] <- mdn

#Cluster 2.2 Data - Mahalanobis Plots/Visuals
C2.995 = quantile(mdn, .995)
plot(df.2, mdn, xlab="Object", ylab="Mahalanobis Distance", main="Cluster 2.2
Outlier Plot")
abline(h=C2.995, col="red", lwd=2)
which(mdn>C2.995)

qqplot(mdn, df.2, xlab="Mahalanobis Distance", main="Cluster 2.2 - QQ Plot")
abline(v=C2.995, col="red", lwd=2)

#Compute Euclidean distances
med <- rbind(c(m1,m2,m3))
ed <- rdist(df.1, med)

#Add ed column to data set x2
x2["EuclideanDistance"] <- ed

#Normalize ed column
euc1 <- x2[[20]]
max2 <- max(euc1)
i <- 1
edn <- vector(mode="numeric", length=y)
while (i<y+1) {
  edn[i] <- euc1[i]/max2
  i<-i+1
}

#Check/verify max & min normalized values (max=1, min>=0)
maxedn <- max(edn)
View(maxedn)
minedn <- min(edn)
```

```
View(minedn)

#Add normalized ed column to data set x2
x2["EuclideanNorm"] <- edn

#Cluster 2.2 Data - Euclidean Plots/Visuals
C2.995 = quantile(edn, .995)
plot(df.2, edn, xlab="Object", ylab="Euclidean Distance", main="Cluster 2.2
Outlier Plot")
abline(h=C2.995, col="red", lwd=2)
which(edn>C2.995)

qqplot(edn, df.2, xlab="Euclidean Distance", main="Cluster 2.2 - QQ Plot")
abline(v=C2.995, col="red", lwd=2)

#Compute Cosine Dissimilarities (1 - cosine similarity)
cd=vector(mode="numeric", length=y)
i<-1
while(i<y+1) {
  cd[i] <- 1-
(((df.1[i,1]*m1)+(df.1[i,2]*m2)+(df.1[i,3]*m3))/((sqrt(df.1[i,1]^2+df.1[i,2]^2+df.1[i,3]^
2))*(sqrt(m1^2+m2^2+m3^2))))
  i<-i+1
}

#Add cd column to data set x2
x2["CosineDissimilarity"] <- cd

#Normalize cd column
cos1 <- x2[[22]]
max3 <- max(cos1)
i <- 1
cdn <- vector(mode="numeric", length=y)
while (i<y+1) {
  cdn[i] <- cos1[i]/max3
  i<-i+1
}

#Check/verify max & min normalized values (max=1, min>=0)
maxcdn <- max(cdn)
View(maxcdn)
mincdn <- min(cdn)
View(mincdn)

#Add normalized cd column to data set x2
x2["CosineDissimilarityNorm"] <- cdn
```

```
#Cluster 2.2 Data - Cosine Plots/Visuals
C2.995 = quantile(cdn, .995)
plot(df.2, cdn, xlab="Object", ylab="Cosine Dissimilarity", main="Cluster 2.2
Outlier Plot")
abline(h=C2.995, col="red", lwd=2)
which(cdn>C2.995)

qqplot(cdn, df.2, xlab="Cosine Dissimilarity", main="Cluster 2.2 - QQ Plot")
abline(v=C2.995, col="red", lwd=2)

#Compute Tanimoto Dissimilarities (1 - Tanimoto Coefficient)
td <- vector(mode="numeric", length=y)
i<-1
while(i<y+1){
  td[i] <- 1-
((((df.1[i,1]*m1)+(df.1[i,2]*m2)+(df.1[i,3]*m3))/(((sqrt(df.1[i,1]^2+df.1[i,2]^2+df.1[i,3
]^2))^2)+((sqrt(m1^2+m2^2+m3^2))^2)-
((df.1[i,1]*m1)+(df.1[i,2]*m2)+(df.1[i,3]*m3)))))
  i<-i+1
}

#Add td column to data set x2
x2["TanimotoDissimilarity"] <- td

#Normalize td column
tan1 <- x2[[24]]
max4 <- max(tan1)
i <- 1
tdn <- vector(mode="numeric", length=y)
while (i<y+1) {
  tdn[i] <- tan1[i]/max4
  i<-i+1
}

#Check/verify max & min normalized values (max=1, min>=0)
maxtdn <- max(tdn)
View(maxtdn)
mintdn <- min(tdn)
View(mintdn)

#Add normalized td column to data set x2
x2["TanimotoDissimilarityNorm"] <- tdn

#Cluster 2.2 Data - Tanimoto Plots/Visuals
C2.995 = quantile(tdn, .995)
```

```
plot(df.2, tdn, xlab="Object", ylab="Tanimoto Dissimilarity", main="Cluster 2.2
Outlier Plot")
abline(h=C2.995, col="red", lwd=2)
which(tdn>C2.995)

qqplot(tdn, df.2, xlab="Tanimoto Dissimilarity", main="Cluster 2.2 - QQ Plot")
abline(v=C2.995, col="red", lwd=2)

#Calculate Outlier Scores from all 4 normalized values
#Includes weight parameters w1 to w4 for each metric (default value is 1 -- sum
of weights = 4)
w1 <- 1
w2 <- 1
w3 <- 1
w4 <- 1
df.3 <- x2[18:25]
View(df.3)
os <- vector(mode="numeric", length=y)
i<-1
while(i<y+1){
  os[i] <- ((df.3[i,2]*w1)+(df.3[i,4]*w2)+(df.3[i,6]*w3)+(df.3[i,8]*w4))
  i<-i+1
}

#Add outlier score column to data set x2
x2["OutlierScore"] <- os
View(x2)

#View max and min outlier scores (max<=4, min>0)
maxos <- max(os)
View(maxos)
minos <- min(os)
View(minos)

#Cluster 2.2 Data - Outlier Score Plots/Visuals
C2.995 = quantile(os, .995)
plot(df.2, os, xlab="Object", ylab="Outlier Score", main="Cluster 2.2 Outlier Plot")
abline(h=C2.995, col="red", lwd=2)
which(os>C2.995)

qqplot(os, df.2, xlab="Outlier Score", main="Cluster 2.2 - QQ Plot")
abline(v=C2.995, col="red", lwd=2)

#Export processed info as csv file on hard drive
write.csv(x2, "K:/Itau/Credit
Card/Paul/Dissertation/OutlierDetectionStageThree/Cluster2.2Outliers.csv")
```

```
#Read Cluster3 data, rename, and read number of rows
Cluster3 <- read.csv("K:/Itau/Credit
Card/Paul/Dissertation/ClusteringStageOne/Cluster3.csv")
View(Cluster3)
x1 <- Cluster3
y <- nrow(x1)

#Create new data frames for outlier detection purposes
df.1 <- data.frame(x1[ , c("PC1N","PC2N","PC3N")], row.names=NULL)
View(df.1)
df.2 <- c(1:y)
View(df.2)

#Compute medians of 3 relevant normalized principal components
#Entails calculation of median for the column vector corresponding to each
dimension
d1 <- df.1[[1]]
m1 <- median(d1)
View(m1)

d2 <- df.1[[2]]
m2 <- median(d2)
View(m2)

d3 <- df.1[[3]]
m3 <- median(d3)
View(m3)

#Compute Mahalanobis Distances
z <- cov(df.1, use = "everything")
md <- mahalanobis(df.1, center = c(m1, m2, m3), cov = z, inverted=TRUE)

#Add md column to data set x1
x1["MahalanobisDistance"] <- md

#Normalize md column
mah1 <- x1[[18]]
max1 <- max(mah1)
i <- 1
mdn <- vector(mode="numeric", length=y)
while (i<y+1) {
  mdn[i] <- mah1[i]/max1
  i<-i+1
}
```

```
#Check/verify max & min normalized values (max=1, min>=0)
maxmdn <- max(mdn)
View(maxmdn)
minmdn <- min(mdn)
View(minmdn)

#Add normalized md column to data set x1
x1["MahalanobisNorm"] <- mdn

#Cluster 3 Data - Mahalanobis Plots/Visuals
C1.995 = quantile(mdn, .995)
plot(df.2, mdn, xlab="Object", ylab="Mahalanobis Distance", main="Cluster 3
Outlier Plot")
abline(h=C1.995, col="red", lwd=2)
which(mdn>C1.995)

qqplot(mdn, df.2, xlab="Mahalanobis Distance", main="Cluster 3 - QQ Plot")
abline(v=C1.995, col="red", lwd=2)

#Compute Euclidean distances
med <- rbind(c(m1,m2,m3))
ed <- rdist(df.1, med)

#Add ed column to data set x1
x1["EuclideanDistance"] <- ed

#Normalize ed column
euc1 <- x1[[20]]
max2 <- max(euc1)
i <- 1
edn <- vector(mode="numeric", length=y)
while (i<y+1) {
  edn[i] <- euc1[i]/max2
  i<-i+1
}

#Check/verify max & min normalized values (max=1, min>=0)
maxedn <- max(edn)
View(maxedn)
minedn <- min(edn)
View(minedn)

#Add normalized ed column to data set x1
x1["EuclideanNorm"] <- edn
```

```
#Cluster 3 Data - Euclidean Plots/Visuals
C1.995 = quantile(edn, .995)
plot(df.2, edn, xlab="Object", ylab="Euclidean Distance", main="Cluster 3 Outlier
Plot")
abline(h=C1.995, col="red", lwd=2)
which(edn>C1.995)

qqplot(edn, df.2, xlab="Euclidean Distance", main="Cluster 3 - QQ Plot")
abline(v=C1.995, col="red", lwd=2)

#Compute Cosine Dissimilarities (1 - cosine similarity)
cd=vector(mode="numeric", length=y)
i<-1
while(i<y+1) {
  cd[i] <- 1-
(((df.1[i,1]*m1)+(df.1[i,2]*m2)+(df.1[i,3]*m3))/((sqrt(df.1[i,1]^2+df.1[i,2]^2+df.1[i,3]^
2))*(sqrt(m1^2+m2^2+m3^2))))
  i<-i+1
}

#Add cd column to data set x1
x1["CosineDissimilarity"] <- cd

#Normalize cd column
cos1 <- x1[[22]]
max3 <- max(cos1)
i <- 1
cdn <- vector(mode="numeric", length=y)
while (i<y+1) {
  cdn[i] <- cos1[i]/max3
  i<-i+1
}

#Check/verify max & min normalized values (max=1, min>=0)
maxcdn <- max(cdn)
View(maxcdn)
mincdn <- min(cdn)
View(mincdn)

#Add normalized cd column to data set x1
x1["CosineDissimilarityNorm"] <- cdn

#Cluster 3 Data - Cosine Plots/Visuals
C1.995 = quantile(cdn, .995)
plot(df.2, cdn, xlab="Object", ylab="Cosine Dissimilarity", main="Cluster 3 Outlier
Plot")
```

```
abline(h=C1.995, col="red", lwd=2)
which(cdn>C1.995)

qqplot(cdn, df.2, xlab="Cosine Dissimilarity", main="Cluster 3 - QQ Plot")
abline(v=C1.995, col="red", lwd=2)

#Compute Tanimoto Dissimilarities (1 - Tanimoto Coefficient)
td <- vector(mode="numeric", length=y)
i<-1
while(i<y+1){
  td[i] <- 1-
((((df.1[i,1]*m1)+(df.1[i,2]*m2)+(df.1[i,3]*m3))/(((sqrt(df.1[i,1]^2+df.1[i,2]^2+df.1[i,3
]^2))^2)+((sqrt(m1^2+m2^2+m3^2))^2)-
((df.1[i,1]*m1)+(df.1[i,2]*m2)+(df.1[i,3]*m3)))))
  i<-i+1
}

#Add td column to data set x1
x1["TanimotoDissimilarity"] <- td

#Normalize td column
tan1 <- x1[[24]]
max4 <- max(tan1)
i <- 1
tdn <- vector(mode="numeric", length=y)
while (i<y+1) {
  tdn[i] <- tan1[i]/max4
  i<-i+1
}

#Check/verify max & min normalized values (max=1, min>=0)
maxtdn <- max(tdn)
View(maxtdn)
mintdn <- min(tdn)
View(mintdn)

#Add normalized td column to data set x1
x1["TanimotoDissimilarityNorm"] <- tdn

#Cluster 3 Data - Tanimoto Plots/Visuals
C1.995 = quantile(tdn, .995)
plot(df.2, tdn, xlab="Object", ylab="Tanimoto Dissimilarity", main="Cluster 3
Outlier Plot")
abline(h=C1.995, col="red", lwd=2)
which(tdn>C1.995)
```

```
qqplot(tdn, df.2, xlab="Tanimoto Dissimilarity", main="Cluster 3 - QQ Plot")
abline(v=C1.995, col="red", lwd=2)

#Calculate Outlier Scores from all 4 normalized values
#Includes weight parameters w1 to w4 for each metric (default value is 1 -- sum
of weights = 4)
w1 <- 1
w2 <- 1
w3 <- 1
w4 <- 1
df.3 <- x1[18:25]
View(df.3)
os <- vector(mode="numeric", length=y)
i<-1
while(i<y+1){
  os[i] <- ((df.3[i,2]*w1)+(df.3[i,4]*w2)+(df.3[i,6]*w3)+(df.3[i,8]*w4))
  i<-i+1
}
View(os)

#Add outlier score column to data set x1
x1["OutlierScore"] <- os
View(x1)

#View max and min outlier scores (max<=4, min>0)
maxos <- max(os)
View(maxos)
minos <- min(os)
View(minos)

#Cluster 3 Data - Outlier Score Plots/Visuals
C1.995 = quantile(os, .995)
plot(df.2, os, xlab="Object", ylab="Outlier Score", main="Cluster 3 Outlier Plot")
abline(h=C1.995, col="red", lwd=2)
which(os>C1.995)

qqplot(os, df.2, xlab="Outlier Score", main="Cluster 3 - QQ Plot")
abline(v=C1.995, col="red", lwd=2)

#Export processed info as csv file on hard drive
write.csv(x1, "K:/Itau/Credit
Card/Paul/Dissertation/OutlierDetectionStageThree/Cluster3Outliers.csv")
```

APPENDIX D: CLUSTERING AND OUTLIER DETECTION APPLICATION VERSION1.0

```
#Automated Clustering and Outlier Detection Application V1
#Stage One - Clustering and Evaluation
#Load necessary packages
library(knitr)
library(rmarkdown)
library(cluster)
library(clusterSim)
library(NbClust)
library(plyr)
library(Rcmdr)
library(gplots)
library(scatterplot3d)
library(stats)
library(chemometrics)
library(fields)
library(fastcluster)

#Import target data set into R and read number of rows
MainData <- read.csv("C:/Users/Eric/Documents/Rutgers - Dissertation
Items/AutomatedClusteringProject/ClusteringExperiment1.csv")
r1 <- nrow(MainData)
c1 <- ncol(MainData)
zz <- c1-1

#Create data frames of MainData
x1 <- MainData[ , 1:c1]
View(x1)
x2 <- MainData[ , 2:c1]

#Normalize all dimensions to be clustered
m <- matrix(nrow=r1, ncol=zz)
for (c in 1:zz) {
  max <- max(x2[ , c])
  min <- min(x2[ , c])
    for (r in 1:r1) {
      m[r,c] <- ((x2[r, c] - min)/(max - min))
    }
}

#Add normalized results to data frame x1, thus creating x1.m
x1.m <- cbind(x1, m)
```

```
#Create data frame of matrix m for principal components analysis
mm <- data.frame(m)

#Perform principal components analysis and generate initial plots of two primary
PC's
trans <- prcomp(mm, center=TRUE, scale.=TRUE)
print(trans)
summary(trans)
plot(trans, typ="l", main="Principal Components - Explained Variance", col="2",
lwd="4")
PC = predict(trans, mm)
plot(PC, type="p", col="2")
plot(PC, type="b", col="2")

#Add principal components to x1.m, thus creating x1.p
x1.p <- cbind(x1.m, PC)

#Set variable and create data frame of PC columns
c2 <- ncol(PC)
x3 <- data.frame(PC)

#Normalize all principal components
p <- matrix(nrow=r1, ncol=c2)
for (c in 1:c2) {
  max <- max(x3[ , c])
  min <- min(x3[ , c])
    for (r in 1:r1) {
      p[r,c] <- ((x3[r, c] - min)/(max - min))
    }
}
View(p)

#Add normalized pc results data frame x1.p, thus creating x1.pc
x1.pc <- cbind(x1.p, p)
View(x1.pc)

#Automatically select principal components needed for clustering
ll <- as.data.frame(summary(trans)[6])
l <- as.vector(ll[3, ], mode="numeric")
s <- 0
PCC <- list()
for (u in 1:c2) {
  if (s < .8) {
    PCC[[u]] <- p[ , u]
  }
  s <- l[u]
}
PCCN <- as.data.frame(PCC)
c3 <- ncol(PCCN)
ClusteringDataPC <- data.frame(p[ , 1:c3])
View(ClusteringDataPC)

#Algorithm Selection Processes (using clusterSim and NbClust packages)
#Following two variables (minK & maxK) correspond to exploration range for
number of clusters
#These can be specified by the user (default range is from 3 to 15)
minK <- 3
maxK <- 15

#Subpart 1 - Explore Hierarchical (Ward and complete link) and PAM
(partitioning around medoids)
fit <- cluster.Sim(ClusteringDataPC, p=6, minClusterNo=minK,
maxClusterNo=maxK, icq="S", outputCsv="AlgorithmResults", distances=c("d2"),
methods=c("m2","m5","m6"))

#Create a full result set for the best model from cluster.Sim process (for choosing
algorithm)
m1 <- matrix(c(fit$method, fit$classes, fit$result), nrow=1, ncol=3)
best.Sim <- data.frame(m1)
names(best.Sim)[1] <- "BestMethod"
names(best.Sim)[2] <- "NumberOfClusters"
names(best.Sim)[3] <- "SilhouetteValue"

#Create a partial result set for the best model from cluster.Sim process (for
choosing k)
v1 <- as.numeric(fit$classes)
v2 <- as.numeric(fit$result)
m1 <- matrix(c(v1, v2), nrow=1, ncol=2)
best.SimClus <- data.frame(m1)
names(best.SimClus)[1] <- "NumberOfClusters"
names(best.SimClus)[2] <- "SilhouetteValue"

#Subpart 2 - Explore K-means
fit2 <- NbClust(ClusteringDataPC, min.nc=minK, max.nc=maxK,
method="kmeans", index="silhouette")

#Create a full result set for the best model from NbClust process (for choosing
algorithm)
m2 <- matrix(c("Kmeans", cbind(fit2$Best.nc)), nrow=1, ncol=3)
best.Nb <- data.frame(m2)
names(best.Nb)[1] <- "BestMethod"
names(best.Nb)[2] <- "NumberOfClusters"
names(best.Nb)[3] <- "SilhouetteValue"

#Create a partial result set for the best model from NbClust process (for choosing
k)
m1 <- matrix(fit2$Best.nc, nrow=1, ncol=2)
best.NbClus <- data.frame(m1)
names(best.NbClus)[1] <- "NumberOfClusters"
names(best.NbClus)[2] <- "SilhouetteValue"

#Combine the two full result sets into a single view, and find best model
information
combined.full <- as.matrix(rbind(best.Nb, best.Sim[ ,1:3]))
comp <- as.vector(combined.full[ , 3])
as.numeric(comp)
if (comp[1] > comp[2]) {
  best.full <- combined.full[1, 1]
  show(combined.full[1, ])
}
if (comp[2] > comp[1]) {
  best.full <- combined.full[2, 1]
  show(combined.full[2, ])
}

#Combine the two partial result sets into a single view, and sort for use in
subsequent processing
combined.partial <- rbind(best.NbClus, best.SimClus[ ,1:2])
sorted.partial <- data.frame(arrange(combined.partial, desc(combined.partial[
,2])))
best.partial <- data.frame(sorted.partial[1, ])

#Set number of clusters to be used in final model generation
k <- best.partial[1,1]


#Use 4 if statements to create the preferred model based upon above algorithm
selection routines

if (best.full == "Kmeans") {
  #Create final Kmeans model and show stats
  model <- kmeans(ClusteringDataPC, centers=k, iter.max=500, nstart=10,
algorithm="Lloyd")
  sil <- silhouette(model$cluster, dist(ClusteringDataPC,"euclidean"))
  avgsil <- summary(sil, FUN=mean) $avg.width
  show(avgsil)
  show(model$centers)
  show(model$size)
  show(model$totss)
  show(model$tot.withinss)
  show(model$betweenss)

  #Create cluster plot of final Kmeans model
  clusplot(ClusteringDataPC, model$cluster, main="Kmeans - 2D Plot of Final
Model", xlab="Principal Component 1", ylab="Principal Component 2",
col.clus=2)

  #Append cluster assignment (adds cluster assignment column to data)
  FinalModel <- data.frame(x1.pc, model$cluster)
  View(FinalModel)
}

if (best.full == "pam") {
  #Create final pam model and show stats
  model <- pam(ClusteringDataPC, k)
  show(model$medoids)
  show(model$id.med)
  show(model$objective)
  show(model$isolation)
  show(model$clusinfo)

  #Create cluster plot of final pam model
  clusplot(model, main="PAM - 2D Plot of Final Model", xlab="Principal
Component 1", ylab="Principal Component 2", shade=TRUE, col.clus=2)

  #Append cluster assignment (adds cluster assignment column to data)
  FinalModel <- data.frame(x1.pc, model$clustering)
  cc <- ncol(FinalModel)
  names(FinalModel)[cc] <- "model.cluster"
  View(FinalModel)
}

if (best.full == "ward") {
  #Build model and generate preliminary visuals
  model <- hclust(dist(ClusteringDataPC, method = "euclidean"), method =
"ward.D2")
  plot(model, main="Ward's Method Hierarchical Clustering - Initial Dendrogram",
hang=-1, col=2)

  #Set color scheme to traffic light style with green (red) for high (low) values
  color <- colorRampPalette(c("red", "orange", "green"))
  #Create horizontal clusters for rows and columns and convert these into
dendograms.
  RowDend <- as.dendrogram(hclust(dist(ClusteringDataPC,
method="euclidean"), method="ward"))
  ColDend <- as.dendrogram(hclust(dist(t(ClusteringDataPC),
method="euclidean"), method="ward"))

  #Draw combined heatmap and dendogram structure
  heatmap.2(as.matrix(ClusteringDataPC), Rowv=RowDend, Colv=ColDend,
main=NULL, xlab="Ward's Method - Row and Column Dendogram", col=color,
margins=c(7,5))

  #Generate final Ward's method model and append cluster assignment to data
  memb <- cutree(model, k=k)
  model$clusters <- assignCluster(ClusteringDataPC, x1.pc, memb)
  model$cluster <- as.integer(model$clusters)
  FinalModel <- data.frame(x1.pc, model$cluster)
  View(FinalModel)
}

if (best.full == "complete") {
  #Build model and generate preliminary visuals
  model <- hclust(dist(ClusteringDataPC, method = "euclidean"), method =
"complete")
  plot(model, main="Complete-Link Hierarchical Clustering - Initial Dendrogram",
hang=-1, col=2)

  #Set color scheme to traffic light style with green (red) for high (low) values
  color <- colorRampPalette(c("red", "orange", "green"))

  #Create horizontal clusters for rows and columns and convert these into
dendograms.
  RowDend <- as.dendrogram(hclust(dist(ClusteringDataPC,
method="euclidean"), method="complete"))
  ColDend <- as.dendrogram(hclust(dist(t(ClusteringDataPC),
method="euclidean"), method="complete"))

  #Draw combined heatmap and dendogram structure
  heatmap.2(as.matrix(ClusteringDataPC), Rowv=RowDend, Colv=ColDend,
main=NULL, xlab="Complete Link - Row and Column Dendogram", col=color,
margins=c(7,5))

  #Generate final Complete-Link model and append cluster assignment to data
  memb <- cutree(model, k=k)
  model$clusters <- assignCluster(ClusteringDataPC, x1.pc, memb)
  model$cluster <- as.integer(model$clusters)
  FinalModel <- data.frame(x1.pc, model$cluster)
  View(FinalModel)
}

#Generate 3D plots of final clustering model
t <- ncol(FinalModel)
assign <- FinalModel[ , t]

if (c3 == 3) {
  #3D Plots of principal components with cluster assignments (entailing four
orientations)
  ddd <- data.frame(p[ , 1:3], assign)
  names(ddd)[1] <- "PC1"
  names(ddd)[2] <- "PC2"
  names(ddd)[3] <- "PC3"
  names(ddd)[4] <- "Cluster"
  scatterplot3d(ddd, col.axis = "green", col.grid = "green", main = "Principal
Components Scatterplot", angle=50, pch = 1)
  scatterplot3d(ddd, col.axis = "green", col.grid = "green", main = "Principal
Components Scatterplot", angle=140, pch = 1)
  scatterplot3d(ddd, col.axis = "green", col.grid = "green", main = "Principal
Components Scatterplot", angle=230, pch = 1)
  scatterplot3d(ddd, col.axis = "green", col.grid = "green", main = "Principal
Components Scatterplot", angle=320, pch = 1)
}

if (c3 > 3) {
  #3D Plots of PCs 1, 2, and 3 with cluster assignments (entailing four
orientations)
  ddd1 <- data.frame(p[ , 1:3], assign)
  names(ddd1)[1] <- "PC1"
  names(ddd1)[2] <- "PC2"
  names(ddd1)[3] <- "PC3"
  names(ddd1)[4] <- "Cluster"
  scatterplot3d(ddd1, col.axis = "green", col.grid = "green", main = "Principal
Components Scatterplot", angle=50, pch = 1)
  scatterplot3d(ddd1, col.axis = "green", col.grid = "green", main = "Principal
Components Scatterplot", angle=140, pch = 1)
  scatterplot3d(ddd1, col.axis = "green", col.grid = "green", main = "Principal
Components Scatterplot", angle=230, pch = 1)
  scatterplot3d(ddd1, col.axis = "green", col.grid = "green", main = "Principal
Components Scatterplot", angle=320, pch = 1)

  #3D Plots of PCs 1, 2, and 4 with cluster assignments (entailing four
orientations)
  ddd2 <- data.frame(cbind(p[ , 1:2], p[ , 4]), assign)
  names(ddd2)[1] <- "PC1"
  names(ddd2)[2] <- "PC2"
  names(ddd2)[3] <- "PC4"
  names(ddd2)[4] <- "Cluster"
  scatterplot3d(ddd2, col.axis = "green", col.grid = "green", main = "Principal
Components Scatterplot", angle=50, pch = 1)
  scatterplot3d(ddd2, col.axis = "green", col.grid = "green", main = "Principal
Components Scatterplot", angle=140, pch = 1)
  scatterplot3d(ddd2, col.axis = "green", col.grid = "green", main = "Principal
Components Scatterplot", angle=230, pch = 1)
  scatterplot3d(ddd2, col.axis = "green", col.grid = "green", main = "Principal
Components Scatterplot", angle=320, pch = 1)

  #3D Plots of PCs 2, 3, and 4 with cluster assignments (entailing four
orientations)
  ddd3 <- data.frame(p[ , 2:4], assign)
  names(ddd3)[1] <- "PC2"
  names(ddd3)[2] <- "PC3"
  names(ddd3)[3] <- "PC4"
  names(ddd3)[4] <- "Cluster"
  scatterplot3d(ddd3, col.axis = "green", col.grid = "green", main = "Principal
Components Scatterplot", angle=50, pch = 1)
  scatterplot3d(ddd3, col.axis = "green", col.grid = "green", main = "Principal
Components Scatterplot", angle=140, pch = 1)
  scatterplot3d(ddd3, col.axis = "green", col.grid = "green", main = "Principal
Components Scatterplot", angle=230, pch = 1)
  scatterplot3d(ddd3, col.axis = "green", col.grid = "green", main = "Principal
Components Scatterplot", angle=320, pch = 1)
}

#Get cluster counts
xx <- max(FinalModel[ ,"model.cluster"])
for (j in 1:xx) {
  cnt1.1 <- count(FinalModel[ , t]==j)
  cnt1.2 <- cnt1.1[2,2]
  cat("Cluster",j,"contains",cnt1.2,"records.\n")
}

#Get cluster means for original variables
aggregate(x2, by=list(model$cluster), FUN=mean)

#Get cluster means for normalized original variables
aggregate(m, by=list(model$cluster), FUN=mean)

#Get cluster means for normalized principal components
aggregate(ClusteringDataPC, by=list(model$cluster), FUN=mean)
```

```
#Get cluster medians for original variables
aggregate(x2, by=list(model$cluster), FUN=median)

#Get cluster medians for normalized original variables
aggregate(m, by=list(model$cluster), FUN=median)

#Get cluster medians for normalized principal components
aggregate(ClusteringDataPC, by=list(model$cluster), FUN=median)

#Perform Kruskal-Wallis tests of significance
#Generate variables and perform statistical testing
col <- ncol(FinalModel)
dd1 <- (col/2)+1
ss1.2 <- data.frame (FinalModel[ ,dd1:col])
dd2 <- ncol(ss1.2)
dd3 <- as.integer(dd2/2)+1

if (c3 == 3) {
  dd4 <- dd3+2
  ss1.1 <- data.frame(cbind(ss1.2[ , dd3:dd4], ss1.2[ ,dd2]))
  names(ss1.1)[1] <- "PC1N"
  names(ss1.1)[2] <- "PC2N"
  names(ss1.1)[3] <- "PC3N"
  names(ss1.1)[4] <- "model.cluster"
  ss1 <- as.matrix(ss1.1)

  t1 <- kruskal.test(PC1N~model.cluster, data=ss1)
  show(t1)
  t2 <- kruskal.test(PC2N~model.cluster, data=ss1)
  show(t2)
  t3 <- kruskal.test(PC3N~model.cluster, data=ss1)
  show(t3)
  t4 <- kruskal.test(PC1N+PC2N+PC3N~model.cluster, data=ss1)
  show(t4)
  print("If p-value less than .05, results are significant and clusters are statistically
different")
}

if (c3 == 4) {
  dd4 <- dd3+3
  ss1.1 <- data.frame(cbind(ss1.2[ , dd3:dd4], ss1.2[ ,dd2]))
  View(ss1.1)
  names(ss1.1)[1] <- "PC1N"
  names(ss1.1)[2] <- "PC2N"
  names(ss1.1)[3] <- "PC3N"
  names(ss1.1)[4] <- "PC4N"
  names(ss1.1)[5] <- "model.cluster"
  ss1 <- as.matrix(ss1.1)

  t1 <- kruskal.test(PC1N~model.cluster, data=ss1)
  show(t1)
  t2 <- kruskal.test(PC2N~model.cluster, data=ss1)
  show(t2)
  t3 <- kruskal.test(PC3N~model.cluster, data=ss1)
  show(t3)
  t4 <- kruskal.test(PC4N~model.cluster, data=ss1)
  show(t4)
  t5 <- kruskal.test(PC1N+PC2N+PC3N+PC4N~model.cluster, data=ss1)
  show(t5)
  print("If p-value less than .05, results are significant and clusters are statistically
different")
}

if (c3 > 4) {
  dd4 <- dd3+4
  ss1.1 <- data.frame(cbind(ss1.2[ , dd3:dd4], ss1.2[ ,dd2]))
  names(ss1.1)[1] <- "PC1N"
  names(ss1.1)[2] <- "PC2N"
  names(ss1.1)[3] <- "PC3N"
  names(ss1.1)[4] <- "PC4N"
  names(ss1.1)[5] <- "PC5N"
  names(ss1.1)[6] <- "model.cluster"
  ss1 <- as.matrix(ss1.1)

  t1 <- kruskal.test(PC1N~model.cluster, data=ss1)
  show(t1)
  t2 <- kruskal.test(PC2N~model.cluster, data=ss1)
  show(t2)
  t3 <- kruskal.test(PC3N~model.cluster, data=ss1)
  show(t3)
  t4 <- kruskal.test(PC4N~model.cluster, data=ss1)
  show(t4)
  t5 <- kruskal.test(PC5N~model.cluster, data=ss1)
  show(t5)
  t6 <- kruskal.test(PC1N+PC2N+PC3N+PC4N+PC5N~model.cluster, data=ss1)
  show(t6)
  print("If p-value less than .05, results are significant and clusters are statistically
different")
}

#Export FinalModel final results (for additional analyses/visualizations)
```

```
write.csv(FinalModel, row.names=FALSE, "C:/Users/Eric/Documents/Rutgers -
Dissertation Items/AutomatedClusteringProject/FinalClusteringResults.csv")


#Stage Two - Outlier Detection via Mahalanobis Distance
#Set initial variables for subsequent usage
r2 <- nrow(FinalModel)
c4 <- ncol(PCCN)
b <- r2*.01
x <- max(FinalModel[ ,"model.cluster"])
p2 <- data.frame(ClusteringDataPC, model$cluster)
p3 <- data.frame(arrange(p2, model$cluster), row.names=NULL)
p4 <- data.frame(arrange(FinalModel, model$cluster), row.names=NULL)

#Separate data by cluster, compute medians, calculate mahalanobis distances,
and create outlier plots for each cluster
for (c in 1:x) {
  nam1 <- paste("Cluster", c, sep = "")
  z <- assign(nam1, subset(p4, model.cluster==c))
  z1 <- subset(p3, model.cluster==c)
  med <- vector(mode="numeric", length=c4)
  for (c5 in 1:c4) {
    med[c5] <- median(z1[ , c5])
  }
  z2 <- data.frame(z1[ ,1:c4])
  z3 <- cov(z2, use = "everything" )
  md <- mahalanobis(z2, center = med, cov = z3, inverted = TRUE)
  nam4 <- paste("Cluster", c, sep = "")
  z4 <- assign(nam4, data.frame(z, md, row.names=NULL))
  y2 <- nrow(z4)
  row.names(z4) <- 1:y2
  df.2 <- c(1:y2)
  if (y2 >= b) {
    Out.999 = quantile(md, .999)
    plot(df.2, md, xlab="Object", ylab="Mahalanobis Distance",
main=c("Cluster",c,"Outlier Plot"))
    abline(h=Out.999, col="red", lwd=2)
    cat("The following rows in cluster",c,"data appear very
suspicious:",which(md>Out.999),"\n")
    qqplot(md, df.2, xlab="Mahalanobis Distance", ylab="Ranked Observation",
main=c("Cluster",c,"QQ Plot"))
    abline(v=Out.999, col="red", lwd=2)
  }
  if (y2 %in% c(3:b)) {
    Out.999 = quantile(md, .999)
    plot(df.2, md, xlab="Object", ylab="Mahalanobis Distance",
main=c("Cluster",c,"Outlier Plot"))
    abline(h=Out.999, col="red", lwd=2)
    cat("The following rows in cluster",c,"data appear very
suspicious:",which(md>Out.999),"\n")
    qqplot(md, df.2, xlab="Mahalanobis Distance", ylab="Ranked Observation",
main=c("Cluster",c,"QQ Plot"))
    abline(v=Out.999, col="red", lwd=2)
    cat("Cluster",c,"has small relative membership and this warrants investigation
of all associated objects","\n")
  }
  if (y2 < 3) {
    cat("Cluster",c,"has an extremely small membership and this clearly warrants
investigation of all associated objects","\n")
  }
}

#Export final clustering results, segregated by cluster
if (c == 3) {
  write.csv(Cluster1, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers -
Dissertation Items/AutomatedClusteringProject/Cluster1Results.csv")
  write.csv(Cluster2, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers -
Dissertation Items/AutomatedClusteringProject/Cluster2Results.csv")
  write.csv(Cluster3, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers -
Dissertation Items/AutomatedClusteringProject/Cluster3Results.csv")
}

if (c == 4) {
  write.csv(Cluster1, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers -
Dissertation Items/AutomatedClusteringProject/Cluster1Results.csv")
  write.csv(Cluster2, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers -
Dissertation Items/AutomatedClusteringProject/Cluster2Results.csv")
  write.csv(Cluster3, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers -
Dissertation Items/AutomatedClusteringProject/Cluster3Results.csv")
  write.csv(Cluster4, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers -
Dissertation Items/AutomatedClusteringProject/Cluster4Results.csv")
}

if (c == 5) {
  write.csv(Cluster1, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers -
Dissertation Items/AutomatedClusteringProject/Cluster1Results.csv")
  write.csv(Cluster2, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers -
Dissertation Items/AutomatedClusteringProject/Cluster2Results.csv")
  write.csv(Cluster3, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers -
Dissertation Items/AutomatedClusteringProject/Cluster3Results.csv")
```

```
  write.csv(Cluster4, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers -
Dissertation Items/AutomatedClusteringProject/Cluster4Results.csv")
  write.csv(Cluster5, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers -
Dissertation Items/AutomatedClusteringProject/Cluster5Results.csv")
}

if (c == 6) {
  write.csv(Cluster1, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers -
Dissertation Items/AutomatedClusteringProject/Cluster1Results.csv")
  write.csv(Cluster2, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers -
Dissertation Items/AutomatedClusteringProject/Cluster2Results.csv")
  write.csv(Cluster3, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers -
Dissertation Items/AutomatedClusteringProject/Cluster3Results.csv")
  write.csv(Cluster4, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers -
Dissertation Items/AutomatedClusteringProject/Cluster4Results.csv")
  write.csv(Cluster5, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers -
Dissertation Items/AutomatedClusteringProject/Cluster5Results.csv")
  write.csv(Cluster6, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers -
Dissertation Items/AutomatedClusteringProject/Cluster6Results.csv")
}

if (c == 7) {
  write.csv(Cluster1, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers -
Dissertation Items/AutomatedClusteringProject/Cluster1Results.csv")
  write.csv(Cluster2, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers -
Dissertation Items/AutomatedClusteringProject/Cluster2Results.csv")
  write.csv(Cluster3, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers -
Dissertation Items/AutomatedClusteringProject/Cluster3Results.csv")
  write.csv(Cluster4, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers -
Dissertation Items/AutomatedClusteringProject/Cluster4Results.csv")
  write.csv(Cluster5, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers -
Dissertation Items/AutomatedClusteringProject/Cluster5Results.csv")
  write.csv(Cluster6, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers -
Dissertation Items/AutomatedClusteringProject/Cluster6Results.csv")
  write.csv(Cluster7, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers -
Dissertation Items/AutomatedClusteringProject/Cluster7Results.csv")
}

if (c == 8) {
  write.csv(Cluster1, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers -
Dissertation Items/AutomatedClusteringProject/Cluster1Results.csv")
  write.csv(Cluster2, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers -
Dissertation Items/AutomatedClusteringProject/Cluster2Results.csv")
  write.csv(Cluster3, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers -
Dissertation Items/AutomatedClusteringProject/Cluster3Results.csv")
```

```
  write.csv(Cluster4, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers -
Dissertation Items/AutomatedClusteringProject/Cluster4Results.csv")
  write.csv(Cluster5, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers -
Dissertation Items/AutomatedClusteringProject/Cluster5Results.csv")
  write.csv(Cluster6, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers -
Dissertation Items/AutomatedClusteringProject/Cluster6Results.csv")
  write.csv(Cluster7, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers -
Dissertation Items/AutomatedClusteringProject/Cluster7Results.csv")
  write.csv(Cluster8, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers -
Dissertation Items/AutomatedClusteringProject/Cluster8Results.csv")
}

if (c == 9) {
  write.csv(Cluster1, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers -
Dissertation Items/AutomatedClusteringProject/Cluster1Results.csv")
  write.csv(Cluster2, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers -
Dissertation Items/AutomatedClusteringProject/Cluster2Results.csv")
  write.csv(Cluster3, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers -
Dissertation Items/AutomatedClusteringProject/Cluster3Results.csv")
  write.csv(Cluster4, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers -
Dissertation Items/AutomatedClusteringProject/Cluster4Results.csv")
  write.csv(Cluster5, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers -
Dissertation Items/AutomatedClusteringProject/Cluster5Results.csv")
  write.csv(Cluster6, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers -
Dissertation Items/AutomatedClusteringProject/Cluster6Results.csv")
  write.csv(Cluster7, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers -
Dissertation Items/AutomatedClusteringProject/Cluster7Results.csv")
  write.csv(Cluster8, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers -
Dissertation Items/AutomatedClusteringProject/Cluster8Results.csv")
  write.csv(Cluster9, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers -
Dissertation Items/AutomatedClusteringProject/Cluster9Results.csv")
}

if (c == 10) {
  write.csv(Cluster1, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers -
Dissertation Items/AutomatedClusteringProject/Cluster1Results.csv")
  write.csv(Cluster2, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers -
Dissertation Items/AutomatedClusteringProject/Cluster2Results.csv")
  write.csv(Cluster3, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers -
Dissertation Items/AutomatedClusteringProject/Cluster3Results.csv")
  write.csv(Cluster4, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers -
Dissertation Items/AutomatedClusteringProject/Cluster4Results.csv")
  write.csv(Cluster5, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers -
Dissertation Items/AutomatedClusteringProject/Cluster5Results.csv")
  write.csv(Cluster6, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers -
Dissertation Items/AutomatedClusteringProject/Cluster6Results.csv")
```

```
  write.csv(Cluster7, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers - 
Dissertation Items/AutomatedClusteringProject/Cluster7Results.csv")
  write.csv(Cluster8, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers - 
Dissertation Items/AutomatedClusteringProject/Cluster8Results.csv")
  write.csv(Cluster9, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers - 
Dissertation Items/AutomatedClusteringProject/Cluster9Results.csv")
  write.csv(Cluster10, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers - 
Dissertation Items/AutomatedClusteringProject/Cluster10Results.csv")
}

if (c == 11) {
  write.csv(Cluster1, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers - 
Dissertation Items/AutomatedClusteringProject/Cluster1Results.csv")
  write.csv(Cluster2, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers - 
Dissertation Items/AutomatedClusteringProject/Cluster2Results.csv")
  write.csv(Cluster3, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers - 
Dissertation Items/AutomatedClusteringProject/Cluster3Results.csv")
  write.csv(Cluster4, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers - 
Dissertation Items/AutomatedClusteringProject/Cluster4Results.csv")
  write.csv(Cluster5, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers - 
Dissertation Items/AutomatedClusteringProject/Cluster5Results.csv")
  write.csv(Cluster6, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers - 
Dissertation Items/AutomatedClusteringProject/Cluster6Results.csv")
  write.csv(Cluster7, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers - 
Dissertation Items/AutomatedClusteringProject/Cluster7Results.csv")
  write.csv(Cluster8, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers - 
Dissertation Items/AutomatedClusteringProject/Cluster8Results.csv")
  write.csv(Cluster9, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers - 
Dissertation Items/AutomatedClusteringProject/Cluster9Results.csv")
  write.csv(Cluster10, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers - 
Dissertation Items/AutomatedClusteringProject/Cluster10Results.csv")
  write.csv(Cluster11, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers - 
Dissertation Items/AutomatedClusteringProject/Cluster11Results.csv")
}

if (c == 12) {
  write.csv(Cluster1, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers - 
Dissertation Items/AutomatedClusteringProject/Cluster1Results.csv")
  write.csv(Cluster2, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers - 
Dissertation Items/AutomatedClusteringProject/Cluster2Results.csv")
  write.csv(Cluster3, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers - 
Dissertation Items/AutomatedClusteringProject/Cluster3Results.csv")
  write.csv(Cluster4, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers - 
Dissertation Items/AutomatedClusteringProject/Cluster4Results.csv")
  write.csv(Cluster5, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers - 
Dissertation Items/AutomatedClusteringProject/Cluster5Results.csv")
```

```
  write.csv(Cluster6, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers -
Dissertation Items/AutomatedClusteringProject/Cluster6Results.csv")
  write.csv(Cluster7, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers -
Dissertation Items/AutomatedClusteringProject/Cluster7Results.csv")
  write.csv(Cluster8, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers -
Dissertation Items/AutomatedClusteringProject/Cluster8Results.csv")
  write.csv(Cluster9, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers -
Dissertation Items/AutomatedClusteringProject/Cluster9Results.csv")
  write.csv(Cluster10, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers -
Dissertation Items/AutomatedClusteringProject/Cluster10Results.csv")
  write.csv(Cluster11, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers -
Dissertation Items/AutomatedClusteringProject/Cluster11Results.csv")
  write.csv(Cluster12, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers -
Dissertation Items/AutomatedClusteringProject/Cluster12Results.csv")
}

if (c == 13) {
  write.csv(Cluster1, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers -
Dissertation Items/AutomatedClusteringProject/Cluster1Results.csv")
  write.csv(Cluster2, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers -
Dissertation Items/AutomatedClusteringProject/Cluster2Results.csv")
  write.csv(Cluster3, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers -
Dissertation Items/AutomatedClusteringProject/Cluster3Results.csv")
  write.csv(Cluster4, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers -
Dissertation Items/AutomatedClusteringProject/Cluster4Results.csv")
  write.csv(Cluster5, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers -
Dissertation Items/AutomatedClusteringProject/Cluster5Results.csv")
  write.csv(Cluster6, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers -
Dissertation Items/AutomatedClusteringProject/Cluster6Results.csv")
  write.csv(Cluster7, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers -
Dissertation Items/AutomatedClusteringProject/Cluster7Results.csv")
  write.csv(Cluster8, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers -
Dissertation Items/AutomatedClusteringProject/Cluster8Results.csv")
  write.csv(Cluster9, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers -
Dissertation Items/AutomatedClusteringProject/Cluster9Results.csv")
  write.csv(Cluster10, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers -
Dissertation Items/AutomatedClusteringProject/Cluster10Results.csv")
  write.csv(Cluster11, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers -
Dissertation Items/AutomatedClusteringProject/Cluster11Results.csv")
  write.csv(Cluster12, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers -
Dissertation Items/AutomatedClusteringProject/Cluster12Results.csv")
  write.csv(Cluster13, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers -
Dissertation Items/AutomatedClusteringProject/Cluster13Results.csv")
}

if (c == 14) {
  write.csv(Cluster1, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers -
Dissertation Items/AutomatedClusteringProject/Cluster1Results.csv")
  write.csv(Cluster2, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers -
Dissertation Items/AutomatedClusteringProject/Cluster2Results.csv")
  write.csv(Cluster3, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers -
Dissertation Items/AutomatedClusteringProject/Cluster3Results.csv")
  write.csv(Cluster4, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers -
Dissertation Items/AutomatedClusteringProject/Cluster4Results.csv")
  write.csv(Cluster5, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers -
Dissertation Items/AutomatedClusteringProject/Cluster5Results.csv")
  write.csv(Cluster6, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers -
Dissertation Items/AutomatedClusteringProject/Cluster6Results.csv")
  write.csv(Cluster7, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers -
Dissertation Items/AutomatedClusteringProject/Cluster7Results.csv")
  write.csv(Cluster8, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers -
Dissertation Items/AutomatedClusteringProject/Cluster8Results.csv")
  write.csv(Cluster9, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers -
Dissertation Items/AutomatedClusteringProject/Cluster9Results.csv")
  write.csv(Cluster10, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers -
Dissertation Items/AutomatedClusteringProject/Cluster10Results.csv")
  write.csv(Cluster11, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers -
Dissertation Items/AutomatedClusteringProject/Cluster11Results.csv")
  write.csv(Cluster12, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers -
Dissertation Items/AutomatedClusteringProject/Cluster12Results.csv")
  write.csv(Cluster13, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers -
Dissertation Items/AutomatedClusteringProject/Cluster13Results.csv")
  write.csv(Cluster14, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers -
Dissertation Items/AutomatedClusteringProject/Cluster14Results.csv")
}

if (c == 15) {
  write.csv(Cluster1, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers -
Dissertation Items/AutomatedClusteringProject/Cluster1Results.csv")
  write.csv(Cluster2, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers -
Dissertation Items/AutomatedClusteringProject/Cluster2Results.csv")
  write.csv(Cluster3, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers -
Dissertation Items/AutomatedClusteringProject/Cluster3Results.csv")
  write.csv(Cluster4, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers -
Dissertation Items/AutomatedClusteringProject/Cluster4Results.csv")
  write.csv(Cluster5, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers -
Dissertation Items/AutomatedClusteringProject/Cluster5Results.csv")
  write.csv(Cluster6, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers -
Dissertation Items/AutomatedClusteringProject/Cluster6Results.csv")
  write.csv(Cluster7, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers -
Dissertation Items/AutomatedClusteringProject/Cluster7Results.csv")
```

```
  write.csv(Cluster8, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers -
Dissertation Items/AutomatedClusteringProject/Cluster8Results.csv")
  write.csv(Cluster9, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers -
Dissertation Items/AutomatedClusteringProject/Cluster9Results.csv")
  write.csv(Cluster10, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers -
Dissertation Items/AutomatedClusteringProject/Cluster10Results.csv")
  write.csv(Cluster11, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers -
Dissertation Items/AutomatedClusteringProject/Cluster11Results.csv")
  write.csv(Cluster12, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers -
Dissertation Items/AutomatedClusteringProject/Cluster12Results.csv")
  write.csv(Cluster13, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers -
Dissertation Items/AutomatedClusteringProject/Cluster13Results.csv")
  write.csv(Cluster14, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers -
Dissertation Items/AutomatedClusteringProject/Cluster14Results.csv")
  write.csv(Cluster15, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers -
Dissertation Items/AutomatedClusteringProject/Cluster15Results.csv")
}
```

APPENDIX E: CLUSTERING AND OUTLIER DETECTION APPLICATION VERSION2.O

```
#Automated Clustering and Outlier Detection Program V2
#For use on both small and larger data sets
#Load necessary packages
library(knitr)
library(rmarkdown)
library(cluster)
library(plyr)
library(Rcmdr)
library(gplots)
library(scatterplot3d)
library(stats)
library(chemometrics)
library(fields)
library(fastcluster)
library(mclust)
library(EMCluster)

#Increase memory limit
memory.size(max=TRUE)
memory.limit(size=999999)

#Import target data set into R and read number of rows
#User specified parameter 1:  Set path name for data set to be imported,
processed, & clustered
MainData <- read.csv("C:/Users/Eric/Documents/Rutgers - Dissertation
Items/AutomatedClusteringProject/ClusteringExperiment1.csv")
r1 <- nrow(MainData)
c1 <- ncol(MainData)
zz <- c1-1
View(MainData)

#Create data frames of MainData
x1 <- MainData[ , 1:c1]
x2 <- MainData[ , 2:c1]


#Stage I  - Data Preprocessing
#Normalize all dimensions to be clustered
m <- matrix(nrow=r1, ncol=zz)
for (c in 1:zz) {
  max <- max(x2[ , c])
  min <- min(x2[ , c])
    for (r in 1:r1) {
      m[r,c] <- ((x2[r, c] - min)/(max - min))
    }
}

#Add normalized results to data frame x1, thus creating x1.m
x1.m <- cbind(x1, m)

#Create data frame of matrix m for principal components analysis
mm <- data.frame(m)

#Perform principal components analysis and generate initial plots of two primary
PC's
trans <- prcomp(mm, center=TRUE, scale.=TRUE)
print(trans)
summary(trans)
plot(trans, typ="l", main="Principal Components - Explained Variance", col="2",
lwd="4")
PC = predict(trans, mm)
plot(PC, type="p", col="2")
plot(PC, type="b", col="2")

#Add principal components to x1.m, thus creating x1.p
x1.p <- cbind(x1.m, PC)
c1.2 <- ncol(x1.p)

#Create variable and create data frame of PC columns
c2 <- ncol(PC)
x3 <- data.frame(PC)

#Normalize all principal components
p <- matrix(nrow=r1, ncol=c2)
for (c in 1:c2) {
  max <- max(x3[ , c])
  min <- min(x3[ , c])
    for (r in 1:r1) {
      p[r,c] <- ((x3[r, c] - min)/(max - min))
    }
}

#Add normalized pc results data frame x1.p, thus creating x1.pc
x1.pc <- cbind(x1.p, p)
View(x1.pc)

#Automatically select principal components needed for clustering
```

```
ll <- as.data.frame(summary(trans)[6])
l <- as.vector(ll[3, ], mode="numeric")
s <- 0
PCC <- list()
for (u in 1:c2) {
  if (s < .8) {
    PCC[[u]] <- p[ , u]
  }
  s <- l[u]
}
PCCN <- as.data.frame(PCC)
c3 <- ncol(PCCN)
ClusteringDataPC <- as.data.frame(PCCN)

#Stage II - Model Simulation Routines (to find model with highest silhouette
coefficient)
#Get a sample of records for simulations (and building the model in the next
stage)
#User specified parameter 2: set sample size (i.e., "sample")
#Note: Sample only used if data set to be processed has more than 40,000
records
#Default sample size set at 40,000 (for efficiency reasons, do not set higher than
this)
if (r1 > 40000) {
  sample <- 40000
  p1 <- sample/r1
  x1.pcs <- x1.pc[sample(nrow(x1.pc),replace=F,size = p1*nrow(x1.pc)),]
  row.names(x1.pcs) <- NULL
}

if (r1 <= 40000) {
  x1.pcs <- data.frame(x1.pc)
}

c4 <- c1.2 + 1
c5 <- c1.2 + c3
ClusteringDataPCS <- data.frame(x1.pcs[ , c4:c5])
row.names(ClusteringDataPCS)<-NULL
View(ClusteringDataPCS)

#Compute distance matrix for subsequent simulations and analyses
d1 <- dist(ClusteringDataPCS, method="euclidean")

#Execute similations using Kmeans, PAM, Complete-Link Hierarchical, and
Ward's Method
```

```
#User specified parameters 3 & 4: Set range (i.e., minclust & maxclust) for # of
clusters to be considered
#Default values set at minclust=3 and maxclust=15
minclust=3
maxclust=15
rowtot <- (maxclust-minclust+1)*4
x=matrix(nrow=rowtot, ncol=3)
y <- 1
for (c in minclust:maxclust) {
  fit <- kmeans(ClusteringDataPCS, c, iter.max=500, nstart=10,
algorithm="Lloyd")
  sil <- silhouette(fit$cluster, d1)
  avgsil <- summary(sil, FUN=mean) $avg.width
  x[y, ] <- c("kmeans", c, avgsil)
  y<-y+1
  fit <- pam(d1, c, diss=TRUE)
  sil <- silhouette(fit$clustering, d1)
  avgsil <- summary(sil, FUN=mean) $avg.width
  x[y, ] <- c("pam", c, avgsil)
  y<-y+1
  fit2 <- hclust(d1, method = "complete")
  memb <- cutree(fit2, k=c)
  fit.clusters <- assignCluster(ClusteringDataPCS, x1.pcs, memb)
  fit.cluster <- as.integer(fit.clusters)
  sil <- silhouette(fit.cluster, d1)
  avgsil <- summary(sil, FUN=mean) $avg.width
  x[y, ] <- c("complete", c, avgsil)
  y<-y+1
  fit2 <- hclust(d1, method = "ward.D2")
  memb <- cutree(fit2, k=c)
  fit.clusters <- assignCluster(ClusteringDataPCS, x1.pcs, memb)
  fit.cluster <- as.integer(fit.clusters)
  sil <- silhouette(fit.cluster, d1)
  avgsil <- summary(sil, FUN=mean) $avg.width
  x[y, ] <- c("ward", c, avgsil)
  y<-y+1
  fit <- init.EM(ClusteringDataPCS, nclass = c, EMC = .EMC.Rnd, stable.solution
= TRUE, min.n = 1, min.n.iter = 10, method = "Rnd.EM")
  sil <- silhouette(fit$class, d1)
  avgsil <- summary(sil, FUN=mean) $avg.width
  x[y, ] <- c("em", c, avgsil)
  y<-y+1
}
View(x)

#Create data frame z, Sort in descending order by Silhouette Coefficient
```

```
z <- data.frame(x)
names(z)[1] <- "Algorithm"
names(z)[2] <- "Clusters"
names(z)[3] <- "Silhouette"
View(z)

s <- arrange(z, desc(Silhouette))
View(s)

#Export processed simulation data
#User specified parameter 5:  Set path name corresponding to desired output
destination
write.csv(s, row.names=FALSE, "C:/Users/Eric/Documents/Rutgers -
Dissertation Items/AutomatedClusteringProject/Silhouettes.csv")


#Stage III - Model Construction
#Automatically select model based upon simulation results
method <- s[1, 1]
View(method)
ss <- as.matrix(s[ , 2:3])
sss <- as.vector(ss[ , 1], mode="numeric")
k <- sss[1]
View(k)

View(ClusteringDataPCS)
View(x1.pcs)

#Build model and create visualizations
if (method == "complete") {
  fit2 <- hclust(d1, method = "complete")
  memb <- cutree(fit2, k=k)

  #Append cluster assignment column
  fit.clusters <- assignCluster(ClusteringDataPCS, x1.pcs, memb)
  fit.cluster <- as.integer(fit.clusters)
  StageOneModel <- data.frame(x1.pcs, fit.cluster)
  View(StageOneModel)
}

if (method == "ward") {
  fit2 <- hclust(d1, method = "ward.D2")
  memb <- cutree(fit2, k=k)

  #Append cluster assignment column
  fit.clusters <- assignCluster(ClusteringDataPCS, x1.pcs, memb)
  fit.cluster <- as.integer(fit.clusters)
  StageOneModel <- data.frame(x1.pcs, fit.cluster)
  View(StageOneModel)
}

if (method == "kmeans") {
  fit <- kmeans(ClusteringDataPCS, centers=k, iter.max=500, nstart=10,
algorithm="Lloyd")

  #Append cluster assignment column
  StageOneModel <- data.frame(x1.pcs, fit$cluster)
  cc <- ncol(StageOneModel)
  names(StageOneModel)[cc] <- "fit.cluster"
  View(StageOneModel)
}

if (method == "pam") {
  fit <- pam(d1, k, diss=TRUE)

  #Append cluster assignment column
  StageOneModel <- data.frame(x1.pcs, fit$clustering)
  cc <- ncol(StageOneModel)
  names(StageOneModel)[cc] <- "fit.cluster"
  View(StageOneModel)
}

if (method == "em") {
  fit <- init.EM(ClusteringDataPCS, nclass = k, EMC = .EMC.Rnd, stable.solution
= TRUE, min.n = 1, min.n.iter = 10, method = "Rnd.EM")

  #Append cluster assignment column
  StageOneModel <- data.frame(x1.pcs, fit$class)
  cc <- ncol(StageOneModel)
  names(StageOneModel)[cc] <- "fit.cluster"
  View(StageOneModel)
}

#Create plots of principal components
if (c3 == 3) {
  p2 <- data.frame(ClusteringDataPCS)
  ddd <- data.frame(p2, StageOneModel[ , "fit.cluster"])
  names(ddd)[1] <- "PC1"
  names(ddd)[2] <- "PC2"
  names(ddd)[3] <- "PC3"
  names(ddd)[4] <- "Cluster"
  scatterplot3d(ddd, col.axis = "green", col.grid = "green", main = "Principal
Components Scatterplot", angle=30, pch = 1)
  scatterplot3d(ddd, col.axis = "green", col.grid = "green", main = "Principal
Components Scatterplot", angle=60, pch = 1)
  scatterplot3d(ddd, col.axis = "green", col.grid = "green", main = "Principal
Components Scatterplot", angle=120, pch = 1)
  scatterplot3d(ddd, col.axis = "green", col.grid = "green", main = "Principal
Components Scatterplot", angle=150, pch = 1)
  scatterplot3d(ddd, col.axis = "green", col.grid = "green", main = "Principal
Components Scatterplot", angle=210, pch = 1)
  scatterplot3d(ddd, col.axis = "green", col.grid = "green", main = "Principal
Components Scatterplot", angle=240, pch = 1)
  scatterplot3d(ddd, col.axis = "green", col.grid = "green", main = "Principal
Components Scatterplot", angle=300, pch = 1)
  scatterplot3d(ddd, col.axis = "green", col.grid = "green", main = "Principal
Components Scatterplot", angle=330, pch = 1)
  }

if (c3 > 3) {
  #3D Plots of PCs 1, 2, and 3 with cluster assignments
  p2.1 <- data.frame(ClusteringDataPCS[ , 1:3])
  ddd1 <- data.frame(p2.1, StageOneModel[ , "fit.cluster"])
  names(ddd1)[1] <- "PC1"
  names(ddd1)[2] <- "PC2"
  names(ddd1)[3] <- "PC3"
  names(ddd1)[4] <- "Cluster"
  scatterplot3d(ddd1, col.axis = "green", col.grid = "green", main = "Principal
Components Scatterplot", angle=50, pch = 1)
  scatterplot3d(ddd1, col.axis = "green", col.grid = "green", main = "Principal
Components Scatterplot", angle=140, pch = 1)
  scatterplot3d(ddd1, col.axis = "green", col.grid = "green", main = "Principal
Components Scatterplot", angle=230, pch = 1)
  scatterplot3d(ddd1, col.axis = "green", col.grid = "green", main = "Principal
Components Scatterplot", angle=320, pch = 1)

  #3D Plots of PCs 1, 2, and 4 with cluster assignments
  p2.2 <- data.frame(ClusteringDataPCS[ , 1:2], ClusteringDataPCS[ , 4])
  ddd2 <- data.frame(p2.2, StageOneModel[ , "fit.cluster"])
  names(ddd2)[1] <- "PC1"
  names(ddd2)[2] <- "PC2"
  names(ddd2)[3] <- "PC4"
  names(ddd2)[4] <- "Cluster"
  scatterplot3d(ddd2, col.axis = "green", col.grid = "green", main = "Principal
Components Scatterplot", angle=50, pch = 1)
  scatterplot3d(ddd2, col.axis = "green", col.grid = "green", main = "Principal
Components Scatterplot", angle=140, pch = 1)
```

```
  scatterplot3d(ddd2, col.axis = "green", col.grid = "green", main = "Principal
Components Scatterplot", angle=230, pch = 1)
  scatterplot3d(ddd2, col.axis = "green", col.grid = "green", main = "Principal
Components Scatterplot", angle=320, pch = 1)

  #3D Plots of PCs 2, 3, and 4 with cluster assignments
  p2.3 <- data.frame(ClusteringDataPCS[ , 2:4])
  ddd3 <- data.frame(p2.3, StageOneModel[ , "fit.cluster"])
  names(ddd3)[1] <- "PC2"
  names(ddd3)[2] <- "PC3"
  names(ddd3)[3] <- "PC4"
  names(ddd3)[4] <- "Cluster"
  scatterplot3d(ddd3, col.axis = "green", col.grid = "green", main = "Principal
Components Scatterplot", angle=50, pch = 1)
  scatterplot3d(ddd3, col.axis = "green", col.grid = "green", main = "Principal
Components Scatterplot", angle=140, pch = 1)
  scatterplot3d(ddd3, col.axis = "green", col.grid = "green", main = "Principal
Components Scatterplot", angle=230, pch = 1)
  scatterplot3d(ddd3, col.axis = "green", col.grid = "green", main = "Principal
Components Scatterplot", angle=320, pch = 1)
}

#Generate data frames for subsequent analysis
c7 <- c1+1
c8 <- (c1*2)-1
x3 <- data.frame(ClusteringDataPCS)
x4 <- data.frame(StageOneModel[ , c7:c8])
x5 <- data.frame(StageOneModel[ , 2:c1])

#Get cluster means for normalized pcs, normalized original variables, and
original variables
aggregate(x3,by=list(fit.cluster),FUN=mean)
aggregate(x4, by=list(fit.cluster), FUN=mean)
aggregate(x5, by=list(fit.cluster), FUN=mean)

#Get cluster medians for normalized pcs, normalized original variables, and
original variables
aggregate(x3,by=list(fit.cluster),FUN=median)
aggregate(x4, by=list(fit.cluster), FUN=median)
aggregate(x5, by=list(fit.cluster), FUN=median)


#Stage IV - OUtlier Detection Via Mahalanobis Distance (initial model)
#Set initial variables for subsequent usage
r2 <- nrow(StageOneModel)
c9 <- ncol(PCCN)
```

```
b1 <- r2*.01
x6 <- max(StageOneModel[ ,"fit.cluster"])
p3 <- data.frame(ClusteringDataPCS, fit.cluster)
p4 <- data.frame(arrange(p3, fit.cluster), row.names=NULL)
p5 <- data.frame(arrange(StageOneModel, fit.cluster), row.names=NULL)

#Separate data by cluster, compute medians, calculate mahalanobis distances,
and create outlier plots for each cluster
for (c in 1:x6) {
  nam1 <- paste("Cluster", c, sep = "")
  z <- assign(nam1, subset(p5, fit.cluster==c))
  z1 <- subset(p4, fit.cluster==c)
  med <- vector(mode="numeric", length=c9)
  for (cc in 1:c9) {
    med[cc] <- median(z1[ , cc])
  }
  z2 <- data.frame(z1[ ,1:c9])
  z3 <- cov(z2, use = "everything" )
  md <- mahalanobis(z2, center = med, cov = z3, inverted = TRUE)
  nam4 <- paste("Cluster", c, sep = "")
  z4 <- assign(nam4, data.frame(z, md, row.names=NULL))
  y2 <- nrow(z4)
  row.names(z4) <- 1:y2
  df.2 <- c(1:y2)
  if (y2 >= b1) {
    Out.999 = quantile(md, .999)
    plot(df.2, md, xlab="Object", ylab="Mahalanobis Distance",
main=c("Cluster",c,"Outlier Plot"))
    abline(h=Out.999, col="red", lwd=2)
    cat("The following rows in cluster",c,"data appear very
suspicious:",which(md>Out.999),"\n")
    qqplot(md, df.2, xlab="Mahalanobis Distance", ylab="Ranked Observation",
main=c("Cluster",c,"QQ Plot"))
    abline(v=Out.999, col="red", lwd=2)
  }
  if (y2 %in% c(3:b1)) {
    Out.999 = quantile(md, .999)
    plot(df.2, md, xlab="Object", ylab="Mahalanobis Distance",
main=c("Cluster",c,"Outlier Plot"))
    abline(h=Out.999, col="red", lwd=2)
    cat("The following rows in cluster",c,"data appear very
suspicious:",which(md>Out.999),"\n")
    qqplot(md, df.2, xlab="Mahalanobis Distance", ylab="Ranked Observation",
main=c("Cluster",c,"QQ Plot"))
    abline(v=Out.999, col="red", lwd=2)
    cat("Cluster",c,"has small relative membership and this warrants investigation
of all associated objects","\n")
  }
  if (y2 < 3) {
    cat("Cluster",c,"has an extremely small membership and this clearly warrants
investigation of all associated objects","\n")
  }
}

#Export processed information, segregated by cluster
#User specified parameter 6: Set path name corresponding to desired output
destination
#Note:  Set path name for each of the following "write.csv" line items in this code
block
if (c == 3) {
  write.csv(Cluster1, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers -
Dissertation Items/AutomatedClusteringProject/Cluster1Results.csv")
  write.csv(Cluster2, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers -
Dissertation Items/AutomatedClusteringProject/Cluster2Results.csv")
  write.csv(Cluster3, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers -
Dissertation Items/AutomatedClusteringProject/Cluster3Results.csv")
}

if (c == 4) {
  write.csv(Cluster1, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers -
Dissertation Items/AutomatedClusteringProject/Cluster1Results.csv")
  write.csv(Cluster2, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers -
Dissertation Items/AutomatedClusteringProject/Cluster2Results.csv")
  write.csv(Cluster3, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers -
Dissertation Items/AutomatedClusteringProject/Cluster3Results.csv")
  write.csv(Cluster4, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers -
Dissertation Items/AutomatedClusteringProject/Cluster4Results.csv")
}

if (c == 5) {
  write.csv(Cluster1, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers -
Dissertation Items/AutomatedClusteringProject/Cluster1Results.csv")
  write.csv(Cluster2, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers -
Dissertation Items/AutomatedClusteringProject/Cluster2Results.csv")
  write.csv(Cluster3, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers -
Dissertation Items/AutomatedClusteringProject/Cluster3Results.csv")
  write.csv(Cluster4, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers -
Dissertation Items/AutomatedClusteringProject/Cluster4Results.csv")
  write.csv(Cluster5, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers -
Dissertation Items/AutomatedClusteringProject/Cluster5Results.csv")
}
```

```
if (c == 6) {
  write.csv(Cluster1, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers - Dissertation Items/AutomatedClusteringProject/Cluster1Results.csv")
  write.csv(Cluster2, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers - Dissertation Items/AutomatedClusteringProject/Cluster2Results.csv")
  write.csv(Cluster3, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers - Dissertation Items/AutomatedClusteringProject/Cluster3Results.csv")
  write.csv(Cluster4, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers - Dissertation Items/AutomatedClusteringProject/Cluster4Results.csv")
  write.csv(Cluster5, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers - Dissertation Items/AutomatedClusteringProject/Cluster5Results.csv")
  write.csv(Cluster6, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers - Dissertation Items/AutomatedClusteringProject/Cluster6Results.csv")
}

if (c == 7) {
  write.csv(Cluster1, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers - Dissertation Items/AutomatedClusteringProject/Cluster1Results.csv")
  write.csv(Cluster2, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers - Dissertation Items/AutomatedClusteringProject/Cluster2Results.csv")
  write.csv(Cluster3, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers - Dissertation Items/AutomatedClusteringProject/Cluster3Results.csv")
  write.csv(Cluster4, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers - Dissertation Items/AutomatedClusteringProject/Cluster4Results.csv")
  write.csv(Cluster5, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers - Dissertation Items/AutomatedClusteringProject/Cluster5Results.csv")
  write.csv(Cluster6, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers - Dissertation Items/AutomatedClusteringProject/Cluster6Results.csv")
  write.csv(Cluster7, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers - Dissertation Items/AutomatedClusteringProject/Cluster7Results.csv")
}

if (c == 8) {
  write.csv(Cluster1, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers - Dissertation Items/AutomatedClusteringProject/Cluster1Results.csv")
  write.csv(Cluster2, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers - Dissertation Items/AutomatedClusteringProject/Cluster2Results.csv")
  write.csv(Cluster3, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers - Dissertation Items/AutomatedClusteringProject/Cluster3Results.csv")
  write.csv(Cluster4, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers - Dissertation Items/AutomatedClusteringProject/Cluster4Results.csv")
  write.csv(Cluster5, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers - Dissertation Items/AutomatedClusteringProject/Cluster5Results.csv")
  write.csv(Cluster6, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers - Dissertation Items/AutomatedClusteringProject/Cluster6Results.csv")
  write.csv(Cluster7, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers -
Dissertation Items/AutomatedClusteringProject/Cluster7Results.csv")
  write.csv(Cluster8, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers -
Dissertation Items/AutomatedClusteringProject/Cluster8Results.csv")
}

if (c == 9) {
  write.csv(Cluster1, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers -
Dissertation Items/AutomatedClusteringProject/Cluster1Results.csv")
  write.csv(Cluster2, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers -
Dissertation Items/AutomatedClusteringProject/Cluster2Results.csv")
  write.csv(Cluster3, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers -
Dissertation Items/AutomatedClusteringProject/Cluster3Results.csv")
  write.csv(Cluster4, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers -
Dissertation Items/AutomatedClusteringProject/Cluster4Results.csv")
  write.csv(Cluster5, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers -
Dissertation Items/AutomatedClusteringProject/Cluster5Results.csv")
  write.csv(Cluster6, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers -
Dissertation Items/AutomatedClusteringProject/Cluster6Results.csv")
  write.csv(Cluster7, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers -
Dissertation Items/AutomatedClusteringProject/Cluster7Results.csv")
  write.csv(Cluster8, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers -
Dissertation Items/AutomatedClusteringProject/Cluster8Results.csv")
  write.csv(Cluster9, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers -
Dissertation Items/AutomatedClusteringProject/Cluster9Results.csv")
}

if (c == 10) {
  write.csv(Cluster1, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers -
Dissertation Items/AutomatedClusteringProject/Cluster1Results.csv")
  write.csv(Cluster2, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers -
Dissertation Items/AutomatedClusteringProject/Cluster2Results.csv")
  write.csv(Cluster3, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers -
Dissertation Items/AutomatedClusteringProject/Cluster3Results.csv")
  write.csv(Cluster4, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers -
Dissertation Items/AutomatedClusteringProject/Cluster4Results.csv")
  write.csv(Cluster5, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers -
Dissertation Items/AutomatedClusteringProject/Cluster5Results.csv")
  write.csv(Cluster6, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers -
Dissertation Items/AutomatedClusteringProject/Cluster6Results.csv")
  write.csv(Cluster7, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers -
Dissertation Items/AutomatedClusteringProject/Cluster7Results.csv")
  write.csv(Cluster8, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers -
Dissertation Items/AutomatedClusteringProject/Cluster8Results.csv")
  write.csv(Cluster9, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers -
Dissertation Items/AutomatedClusteringProject/Cluster9Results.csv")
```

```
  write.csv(Cluster10, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers -
Dissertation Items/AutomatedClusteringProject/Cluster10Results.csv")
}

if (c == 11) {
  write.csv(Cluster1, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers -
Dissertation Items/AutomatedClusteringProject/Cluster1Results.csv")
  write.csv(Cluster2, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers -
Dissertation Items/AutomatedClusteringProject/Cluster2Results.csv")
  write.csv(Cluster3, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers -
Dissertation Items/AutomatedClusteringProject/Cluster3Results.csv")
  write.csv(Cluster4, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers -
Dissertation Items/AutomatedClusteringProject/Cluster4Results.csv")
  write.csv(Cluster5, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers -
Dissertation Items/AutomatedClusteringProject/Cluster5Results.csv")
  write.csv(Cluster6, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers -
Dissertation Items/AutomatedClusteringProject/Cluster6Results.csv")
  write.csv(Cluster7, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers -
Dissertation Items/AutomatedClusteringProject/Cluster7Results.csv")
  write.csv(Cluster8, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers -
Dissertation Items/AutomatedClusteringProject/Cluster8Results.csv")
  write.csv(Cluster9, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers -
Dissertation Items/AutomatedClusteringProject/Cluster9Results.csv")
  write.csv(Cluster10, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers -
Dissertation Items/AutomatedClusteringProject/Cluster10Results.csv")
  write.csv(Cluster11, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers -
Dissertation Items/AutomatedClusteringProject/Cluster11Results.csv")
}

if (c == 12) {
  write.csv(Cluster1, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers -
Dissertation Items/AutomatedClusteringProject/Cluster1Results.csv")
  write.csv(Cluster2, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers -
Dissertation Items/AutomatedClusteringProject/Cluster2Results.csv")
  write.csv(Cluster3, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers -
Dissertation Items/AutomatedClusteringProject/Cluster3Results.csv")
  write.csv(Cluster4, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers -
Dissertation Items/AutomatedClusteringProject/Cluster4Results.csv")
  write.csv(Cluster5, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers -
Dissertation Items/AutomatedClusteringProject/Cluster5Results.csv")
  write.csv(Cluster6, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers -
Dissertation Items/AutomatedClusteringProject/Cluster6Results.csv")
  write.csv(Cluster7, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers -
Dissertation Items/AutomatedClusteringProject/Cluster7Results.csv")
  write.csv(Cluster8, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers -
Dissertation Items/AutomatedClusteringProject/Cluster8Results.csv")
```

```
  write.csv(Cluster9, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers - 
Dissertation Items/AutomatedClusteringProject/Cluster9Results.csv")
  write.csv(Cluster10, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers - 
Dissertation Items/AutomatedClusteringProject/Cluster10Results.csv")
  write.csv(Cluster11, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers - 
Dissertation Items/AutomatedClusteringProject/Cluster11Results.csv")
  write.csv(Cluster12, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers - 
Dissertation Items/AutomatedClusteringProject/Cluster12Results.csv")
}

if (c == 13) {
  write.csv(Cluster1, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers - 
Dissertation Items/AutomatedClusteringProject/Cluster1Results.csv")
  write.csv(Cluster2, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers - 
Dissertation Items/AutomatedClusteringProject/Cluster2Results.csv")
  write.csv(Cluster3, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers - 
Dissertation Items/AutomatedClusteringProject/Cluster3Results.csv")
  write.csv(Cluster4, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers - 
Dissertation Items/AutomatedClusteringProject/Cluster4Results.csv")
  write.csv(Cluster5, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers - 
Dissertation Items/AutomatedClusteringProject/Cluster5Results.csv")
  write.csv(Cluster6, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers - 
Dissertation Items/AutomatedClusteringProject/Cluster6Results.csv")
  write.csv(Cluster7, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers - 
Dissertation Items/AutomatedClusteringProject/Cluster7Results.csv")
  write.csv(Cluster8, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers - 
Dissertation Items/AutomatedClusteringProject/Cluster8Results.csv")
  write.csv(Cluster9, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers - 
Dissertation Items/AutomatedClusteringProject/Cluster9Results.csv")
  write.csv(Cluster10, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers - 
Dissertation Items/AutomatedClusteringProject/Cluster10Results.csv")
  write.csv(Cluster11, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers - 
Dissertation Items/AutomatedClusteringProject/Cluster11Results.csv")
  write.csv(Cluster12, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers - 
Dissertation Items/AutomatedClusteringProject/Cluster12Results.csv")
  write.csv(Cluster13, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers - 
Dissertation Items/AutomatedClusteringProject/Cluster13Results.csv")
}

if (c == 14) {
  write.csv(Cluster1, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers - 
Dissertation Items/AutomatedClusteringProject/Cluster1Results.csv")
  write.csv(Cluster2, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers - 
Dissertation Items/AutomatedClusteringProject/Cluster2Results.csv")
  write.csv(Cluster3, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers - 
Dissertation Items/AutomatedClusteringProject/Cluster3Results.csv")
```

```
  write.csv(Cluster4, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers -
Dissertation Items/AutomatedClusteringProject/Cluster4Results.csv")
  write.csv(Cluster5, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers -
Dissertation Items/AutomatedClusteringProject/Cluster5Results.csv")
  write.csv(Cluster6, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers -
Dissertation Items/AutomatedClusteringProject/Cluster6Results.csv")
  write.csv(Cluster7, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers -
Dissertation Items/AutomatedClusteringProject/Cluster7Results.csv")
  write.csv(Cluster8, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers -
Dissertation Items/AutomatedClusteringProject/Cluster8Results.csv")
  write.csv(Cluster9, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers -
Dissertation Items/AutomatedClusteringProject/Cluster9Results.csv")
  write.csv(Cluster10, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers -
Dissertation Items/AutomatedClusteringProject/Cluster10Results.csv")
  write.csv(Cluster11, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers -
Dissertation Items/AutomatedClusteringProject/Cluster11Results.csv")
  write.csv(Cluster12, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers -
Dissertation Items/AutomatedClusteringProject/Cluster12Results.csv")
  write.csv(Cluster13, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers -
Dissertation Items/AutomatedClusteringProject/Cluster13Results.csv")
  write.csv(Cluster14, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers -
Dissertation Items/AutomatedClusteringProject/Cluster14Results.csv")
}

if (c == 15) {
  write.csv(Cluster1, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers -
Dissertation Items/AutomatedClusteringProject/Cluster1Results.csv")
  write.csv(Cluster2, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers -
Dissertation Items/AutomatedClusteringProject/Cluster2Results.csv")
  write.csv(Cluster3, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers -
Dissertation Items/AutomatedClusteringProject/Cluster3Results.csv")
  write.csv(Cluster4, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers -
Dissertation Items/AutomatedClusteringProject/Cluster4Results.csv")
  write.csv(Cluster5, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers -
Dissertation Items/AutomatedClusteringProject/Cluster5Results.csv")
  write.csv(Cluster6, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers -
Dissertation Items/AutomatedClusteringProject/Cluster6Results.csv")
  write.csv(Cluster7, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers -
Dissertation Items/AutomatedClusteringProject/Cluster7Results.csv")
  write.csv(Cluster8, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers -
Dissertation Items/AutomatedClusteringProject/Cluster8Results.csv")
  write.csv(Cluster9, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers -
Dissertation Items/AutomatedClusteringProject/Cluster9Results.csv")
  write.csv(Cluster10, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers -
Dissertation Items/AutomatedClusteringProject/Cluster10Results.csv")
```

```
  write.csv(Cluster11, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers -
Dissertation Items/AutomatedClusteringProject/Cluster11Results.csv")
  write.csv(Cluster12, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers -
Dissertation Items/AutomatedClusteringProject/Cluster12Results.csv")
  write.csv(Cluster13, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers -
Dissertation Items/AutomatedClusteringProject/Cluster13Results.csv")
  write.csv(Cluster14, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers -
Dissertation Items/AutomatedClusteringProject/Cluster14Results.csv")
  write.csv(Cluster15, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers -
Dissertation Items/AutomatedClusteringProject/Cluster15Results.csv")
}


#Stage V - Assigning All Records to clusters based on Training Model
#Only executed if model is based on a sample from the data set
#Create new data frames and variables for clustering operations
if (r1 > 40000) {
  TestData <- data.frame(x1.pc)
  View(TestData)
  ClusteringTestDataPC <- data.frame(ClusteringDataPC)
  View(ClusteringTestDataPC)

  c10 <- ncol(StageOneModel)
  FitModel <- data.frame(ClusteringDataPCS, StageOneModel[ , c10])
  c11 <- ncol(FitModel)
  names(FitModel)[c11] <- "fit.cluster"
  View(FitModel)

  #Compute training model cluster reps and calculate distances from all reps to
test records
  c3.1 <- ncol(ClusteringTestDataPC)
  c12 <- max(StageOneModel[ , "fit.cluster"])
  r4 <- nrow(ClusteringTestDataPC)
  for (a in 1:c12) {
    g1 <- subset(FitModel, fit.cluster==a)
    v <- vector(mode="numeric", length=c3.1)
    for (b in 1:c3.1) {
      v[b] <- mean(g1[ , b])
    }
    v <- rbind(v)
    v1 <- vector(mode="numeric", length=r4)
    for (c1.1 in 1:r4) {
      v1[c1.1] <- rdist(ClusteringTestDataPC[c1.1, 1:c3.1], v)
    }
    head <- paste("EucDist", a, sep = "")
    ClusteringTestDataPC[head] <- v1
  }
  View(ClusteringTestDataPC)

  #Assign test records to clusters
  c13 <- ncol(ClusteringTestDataPC)
  z3 <- c3.1 + 1
  v2 <- vector(mode="numeric", length=r4)
  v3 <- vector(mode="numeric", length=r4)
  for (h in 1:r4) {
    t1 <- min(ClusteringTestDataPC[h, z3:c13])
    v2[h] <- t1
    t2 <- which.min(apply(ClusteringTestDataPC[h, z3:c13], MARGIN=2, min))
    v3[h] <- t2
  }
  ClusteringTestDataPC["EuclideanDistance"] <- v2
  ClusteringTestDataPC["Test.Cluster"] <- v3
  TestData["fit.cluster2"] <- v3
  TestModel <- data.frame(TestData)
  View(TestModel)

  #Create variables
  c14 <- c1.2 + 1
  c15 <- c1.2 + c3.1
  c16 <- ncol(TestModel)

  if (c9 == 3) {
    #3D Plot of PCS 1, 2, and 3 with cluster assignments
    p6 <- data.frame(TestModel[ , c14:c15])
    ddd <- data.frame(p6, TestModel[ , c16])
    names(ddd)[1] <- "PC1"
    names(ddd)[2] <- "PC2"
    names(ddd)[3] <- "PC3"
    names(ddd)[4] <- "Cluster"
    scatterplot3d(ddd, col.axis = "green", col.grid = "green", main = "Principal
Components Scatterplot", angle=30, pch = 1)
    scatterplot3d(ddd, col.axis = "green", col.grid = "green", main = "Principal
Components Scatterplot", angle=60, pch = 1)
    scatterplot3d(ddd, col.axis = "green", col.grid = "green", main = "Principal
Components Scatterplot", angle=120, pch = 1)
    scatterplot3d(ddd, col.axis = "green", col.grid = "green", main = "Principal
Components Scatterplot", angle=150, pch = 1)
    scatterplot3d(ddd, col.axis = "green", col.grid = "green", main = "Principal
Components Scatterplot", angle=210, pch = 1)
    scatterplot3d(ddd, col.axis = "green", col.grid = "green", main = "Principal
Components Scatterplot", angle=240, pch = 1)
```

```
    scatterplot3d(ddd, col.axis = "green", col.grid = "green", main = "Principal
Components Scatterplot", angle=300, pch = 1)
    scatterplot3d(ddd, col.axis = "green", col.grid = "green", main = "Principal
Components Scatterplot", angle=330, pch = 1)
  }

  if (c3 > 3) {
    #3D Plots of PCs 1, 2, and 3 with cluster assignments
    p6.1 <- data.frame(TestModel[ , c14:c15])
    ddd1 <- data.frame(p6.1, TestModel[ , c16])
    names(ddd1)[1] <- "PC1"
    names(ddd1)[2] <- "PC2"
    names(ddd1)[3] <- "PC3"
    names(ddd1)[4] <- "Cluster"
    scatterplot3d(ddd1, col.axis = "green", col.grid = "green", main = "Principal
Components Scatterplot", angle=50, pch = 1)
    scatterplot3d(ddd1, col.axis = "green", col.grid = "green", main = "Principal
Components Scatterplot", angle=140, pch = 1)
    scatterplot3d(ddd1, col.axis = "green", col.grid = "green", main = "Principal
Components Scatterplot", angle=230, pch = 1)
    scatterplot3d(ddd1, col.axis = "green", col.grid = "green", main = "Principal
Components Scatterplot", angle=320, pch = 1)

    #3D Plots of PCs 1, 2, and 4 with cluster assignments
    c14.1 <- c14 + 1
    p6.2 <- data.frame(TestModel[ , c14:c14.1], TestModel[ , c15])
    ddd2 <- data.frame(p6.2, TestModel[ , c16])
    names(ddd2)[1] <- "PC1"
    names(ddd2)[2] <- "PC2"
    names(ddd2)[3] <- "PC4"
    names(ddd2)[4] <- "Cluster"
    scatterplot3d(ddd2, col.axis = "green", col.grid = "green", main = "Principal
Components Scatterplot", angle=50, pch = 1)
    scatterplot3d(ddd2, col.axis = "green", col.grid = "green", main = "Principal
Components Scatterplot", angle=140, pch = 1)
    scatterplot3d(ddd2, col.axis = "green", col.grid = "green", main = "Principal
Components Scatterplot", angle=230, pch = 1)
    scatterplot3d(ddd2, col.axis = "green", col.grid = "green", main = "Principal
Components Scatterplot", angle=320, pch = 1)

    #3D Plots of PCs 2, 3, and 4 with cluster assignments
    p6.3 <- data.frame(TestModel[ , c14.1:c15])
    ddd3 <- data.frame(p6.3, TestModel[ , c16])
    names(ddd3)[1] <- "PC2"
    names(ddd3)[2] <- "PC3"
    names(ddd3)[3] <- "PC4"
    names(ddd3)[4] <- "Cluster"
    scatterplot3d(ddd3, col.axis = "green", col.grid = "green", main = "Principal
Components Scatterplot", angle=50, pch = 1)
    scatterplot3d(ddd3, col.axis = "green", col.grid = "green", main = "Principal
Components Scatterplot", angle=140, pch = 1)
    scatterplot3d(ddd3, col.axis = "green", col.grid = "green", main = "Principal
Components Scatterplot", angle=230, pch = 1)
    scatterplot3d(ddd3, col.axis = "green", col.grid = "green", main = "Principal
Components Scatterplot", angle=320, pch = 1)
  }

  #Generate data frames for subsequent analysis
  c7 <- c1+1
  c8 <- (c1*2)-1
  x7 <- data.frame(ClusteringTestDataPC[ , 1:c3.1])
  x8 <- data.frame(TestModel[ , c7:c8])
  x9 <- data.frame(TestModel[ , 2:c1])

  #Get cluster means for normalized pcs, normalized original variables, and
original variables
  aggregate(x7,by=list(ClusteringTestDataPC[ , "Test.Cluster"]),FUN=mean)
  aggregate(x8, by=list(TestModel[ , c16]), FUN=mean)
  aggregate(x9, by=list(TestModel[ , c16]), FUN=mean)

  #Get cluster medians for normalized pcs, normalized original variables, and
original variables
  aggregate(x7,by=list(ClusteringTestDataPC[ , "Test.Cluster"]),FUN=median)
  aggregate(x8, by=list(TestModel[ , c16]), FUN=median)
  aggregate(x9, by=list(TestModel[ , c16]), FUN=median)

  #Stage VI - OUtlier Detection Via Mahalanobis Distance (test model- all objects)
  #Set initial variables for subsequent usage
  r5 <- nrow(TestModel)
  b2 <- r5*.01
  x10 <- max(TestModel[ , c16])
  p7 <- data.frame(ClusteringTestDataPC[ , 1:c3.1], TestModel[ , c16])
  names(p7)[ncol(p7)] <- "fit.cluster2"
  p8 <- data.frame(arrange(p7, fit.cluster2), row.names=NULL)
  p9 <- data.frame(arrange(TestModel, fit.cluster2), row.names=NULL)

  #Separate data by cluster, compute medians, calculate mahalanobis distances,
and create outlier plots for each cluster
  for (c in 1:x10) {
    nam1 <- paste("Cluster", c, sep = "")
    z <- assign(nam1, subset(p9, fit.cluster2==c))
    z1 <- subset(p8, fit.cluster2==c)
    med <- vector(mode="numeric", length=c3.1)
    for (cc in 1:c3.1) {
      med[cc] <- median(z1[ , cc])
    }
    z2 <- data.frame(z1[ ,1:c3.1])
    z3 <- cov(z2, use = "everything" )
    md <- mahalanobis(z2, center = med, cov = z3, inverted = TRUE)
    nam4 <- paste("Cluster", c, sep = "")
    z4 <- assign(nam4, data.frame(z, md, row.names=NULL))
    y2 <- nrow(z4)
    row.names(z4) <- 1:y2
    df.2 <- c(1:y2)
    if (y2 >= b2) {
      Out.999 = quantile(md, .999)
      plot(df.2, md, xlab="Object", ylab="Mahalanobis Distance",
main=c("Cluster",c,"Outlier Plot"))
      abline(h=Out.999, col="red", lwd=2)
      cat("The following rows in cluster",c,"data appear very
suspicious:",which(md>Out.999),"\n")
      qqplot(md, df.2, xlab="Mahalanobis Distance", ylab="Ranked Observation",
main=c("Cluster",c,"QQ Plot"))
      abline(v=Out.999, col="red", lwd=2)
    }
    if (y2 %in% c(3:b2)) {
      Out.999 = quantile(md, .999)
      plot(df.2, md, xlab="Object", ylab="Mahalanobis Distance",
main=c("Cluster",c,"Outlier Plot"))
      abline(h=Out.999, col="red", lwd=2)
      cat("The following rows in cluster",c,"data appear very
suspicious:",which(md>Out.999),"\n")
      qqplot(md, df.2, xlab="Mahalanobis Distance", ylab="Ranked Observation",
main=c("Cluster",c,"QQ Plot"))
      abline(v=Out.999, col="red", lwd=2)
      cat("Cluster",c,"has small relative membership and this warrants investigation
of all associated objects","\n")
    }
    if (y2 < 3) {
      cat("Cluster",c,"has an extremely small membership and this clearly warrants
investigation of all associated objects","\n")
    }
  }

  #Export processed information, segregated by clusters (if more than 15
clusters, add info as needed)
  #User specified parameter n:  Set path name for each of the following line items
  #This path name is the location where the processed output files are to be sent
  if (c == 3) {
    write.csv(Cluster1, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers -
Dissertation Items/AutomatedClusteringProject/Cluster1TestResults.csv")
    write.csv(Cluster2, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers -
Dissertation Items/AutomatedClusteringProject/Cluster2TestResults.csv")
    write.csv(Cluster3, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers -
Dissertation Items/AutomatedClusteringProject/Cluster3TestResults.csv")
  }

  if (c == 4) {
    write.csv(Cluster1, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers -
Dissertation Items/AutomatedClusteringProject/Cluster1TestResults.csv")
    write.csv(Cluster2, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers -
Dissertation Items/AutomatedClusteringProject/Cluster2TestResults.csv")
    write.csv(Cluster3, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers -
Dissertation Items/AutomatedClusteringProject/Cluster3TestResults.csv")
    write.csv(Cluster4, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers -
Dissertation Items/AutomatedClusteringProject/Cluster4TestResults.csv")
  }

  if (c == 5) {
    write.csv(Cluster1, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers -
Dissertation Items/AutomatedClusteringProject/Cluster1TestResults.csv")
    write.csv(Cluster2, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers -
Dissertation Items/AutomatedClusteringProject/Cluster2TestResults.csv")
    write.csv(Cluster3, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers -
Dissertation Items/AutomatedClusteringProject/Cluster3TestResults.csv")
    write.csv(Cluster4, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers -
Dissertation Items/AutomatedClusteringProject/Cluster4TestResults.csv")
    write.csv(Cluster5, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers -
Dissertation Items/AutomatedClusteringProject/Cluster5TestResults.csv")
  }

  if (c == 6) {
    write.csv(Cluster1, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers -
Dissertation Items/AutomatedClusteringProject/Cluster1TestResults.csv")
    write.csv(Cluster2, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers -
Dissertation Items/AutomatedClusteringProject/Cluster2TestResults.csv")
    write.csv(Cluster3, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers -
Dissertation Items/AutomatedClusteringProject/Cluster3TestResults.csv")
    write.csv(Cluster4, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers -
Dissertation Items/AutomatedClusteringProject/Cluster4TestResults.csv")
    write.csv(Cluster5, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers -
Dissertation Items/AutomatedClusteringProject/Cluster5TestResults.csv")
    write.csv(Cluster6, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers -
Dissertation Items/AutomatedClusteringProject/Cluster6TestResults.csv")
```

```
  }

  if (c == 7) {
    write.csv(Cluster1, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers -
Dissertation Items/AutomatedClusteringProject/Cluster1TestResults.csv")
    write.csv(Cluster2, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers -
Dissertation Items/AutomatedClusteringProject/Cluster2TestResults.csv")
    write.csv(Cluster3, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers -
Dissertation Items/AutomatedClusteringProject/Cluster3TestResults.csv")
    write.csv(Cluster4, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers -
Dissertation Items/AutomatedClusteringProject/Cluster4TestResults.csv")
    write.csv(Cluster5, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers -
Dissertation Items/AutomatedClusteringProject/Cluster5TestResults.csv")
    write.csv(Cluster6, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers -
Dissertation Items/AutomatedClusteringProject/Cluster6TestResults.csv")
    write.csv(Cluster7, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers -
Dissertation Items/AutomatedClusteringProject/Cluster7TestResults.csv")
  }

  if (c == 8) {
    write.csv(Cluster1, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers -
Dissertation Items/AutomatedClusteringProject/Cluster1TestResults.csv")
    write.csv(Cluster2, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers -
Dissertation Items/AutomatedClusteringProject/Cluster2TestResults.csv")
    write.csv(Cluster3, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers -
Dissertation Items/AutomatedClusteringProject/Cluster3TestResults.csv")
    write.csv(Cluster4, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers -
Dissertation Items/AutomatedClusteringProject/Cluster4TestResults.csv")
    write.csv(Cluster5, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers -
Dissertation Items/AutomatedClusteringProject/Cluster5TestResults.csv")
    write.csv(Cluster6, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers -
Dissertation Items/AutomatedClusteringProject/Cluster6TestResults.csv")
    write.csv(Cluster7, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers -
Dissertation Items/AutomatedClusteringProject/Cluster7TestResults.csv")
    write.csv(Cluster8, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers -
Dissertation Items/AutomatedClusteringProject/Cluster8TestResults.csv")
  }

  if (c == 9) {
    write.csv(Cluster1, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers -
Dissertation Items/AutomatedClusteringProject/Cluster1TestResults.csv")
    write.csv(Cluster2, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers -
Dissertation Items/AutomatedClusteringProject/Cluster2TestResults.csv")
    write.csv(Cluster3, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers -
Dissertation Items/AutomatedClusteringProject/Cluster3TestResults.csv")
```

```
    write.csv(Cluster4, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers - 
Dissertation Items/AutomatedClusteringProject/Cluster4TestResults.csv")
    write.csv(Cluster5, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers - 
Dissertation Items/AutomatedClusteringProject/Cluster5TestResults.csv")
    write.csv(Cluster6, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers - 
Dissertation Items/AutomatedClusteringProject/Cluster6TestResults.csv")
    write.csv(Cluster7, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers - 
Dissertation Items/AutomatedClusteringProject/Cluster7TestResults.csv")
    write.csv(Cluster8, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers - 
Dissertation Items/AutomatedClusteringProject/Cluster8TestResults.csv")
    write.csv(Cluster9, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers - 
Dissertation Items/AutomatedClusteringProject/Cluster9TestResults.csv")
  }

  if (c == 10) {
    write.csv(Cluster1, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers - 
Dissertation Items/AutomatedClusteringProject/Cluster1TestResults.csv")
    write.csv(Cluster2, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers - 
Dissertation Items/AutomatedClusteringProject/Cluster2TestResults.csv")
    write.csv(Cluster3, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers - 
Dissertation Items/AutomatedClusteringProject/Cluster3TestResults.csv")
    write.csv(Cluster4, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers - 
Dissertation Items/AutomatedClusteringProject/Cluster4TestResults.csv")
    write.csv(Cluster5, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers - 
Dissertation Items/AutomatedClusteringProject/Cluster5TestResults.csv")
    write.csv(Cluster6, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers - 
Dissertation Items/AutomatedClusteringProject/Cluster6TestResults.csv")
    write.csv(Cluster7, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers - 
Dissertation Items/AutomatedClusteringProject/Cluster7TestResults.csv")
    write.csv(Cluster8, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers - 
Dissertation Items/AutomatedClusteringProject/Cluster8TestResults.csv")
    write.csv(Cluster9, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers - 
Dissertation Items/AutomatedClusteringProject/Cluster9TestResults.csv")
    write.csv(Cluster10, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers - 
Dissertation Items/AutomatedClusteringProject/Cluster10TestResults.csv")
  }

  if (c == 11) {
    write.csv(Cluster1, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers - 
Dissertation Items/AutomatedClusteringProject/Cluster1TestResults.csv")
    write.csv(Cluster2, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers - 
Dissertation Items/AutomatedClusteringProject/Cluster2TestResults.csv")
    write.csv(Cluster3, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers - 
Dissertation Items/AutomatedClusteringProject/Cluster3TestResults.csv")
    write.csv(Cluster4, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers - 
Dissertation Items/AutomatedClusteringProject/Cluster4TestResults.csv")
```

```
    write.csv(Cluster5, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers - 
Dissertation Items/AutomatedClusteringProject/Cluster5TestResults.csv")
    write.csv(Cluster6, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers - 
Dissertation Items/AutomatedClusteringProject/Cluster6TestResults.csv")
    write.csv(Cluster7, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers - 
Dissertation Items/AutomatedClusteringProject/Cluster7TestResults.csv")
    write.csv(Cluster8, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers - 
Dissertation Items/AutomatedClusteringProject/Cluster8TestResults.csv")
    write.csv(Cluster9, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers - 
Dissertation Items/AutomatedClusteringProject/Cluster9TestResults.csv")
    write.csv(Cluster10, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers - 
Dissertation Items/AutomatedClusteringProject/Cluster10TestResults.csv")
    write.csv(Cluster11, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers - 
Dissertation Items/AutomatedClusteringProject/Cluster11TestResults.csv")
  }

  if (c == 12) {
    write.csv(Cluster1, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers - 
Dissertation Items/AutomatedClusteringProject/Cluster1TestResults.csv")
    write.csv(Cluster2, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers - 
Dissertation Items/AutomatedClusteringProject/Cluster2TestResults.csv")
    write.csv(Cluster3, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers - 
Dissertation Items/AutomatedClusteringProject/Cluster3TestResults.csv")
    write.csv(Cluster4, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers - 
Dissertation Items/AutomatedClusteringProject/Cluster4TestResults.csv")
    write.csv(Cluster5, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers - 
Dissertation Items/AutomatedClusteringProject/Cluster5TestResults.csv")
    write.csv(Cluster6, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers - 
Dissertation Items/AutomatedClusteringProject/Cluster6TestResults.csv")
    write.csv(Cluster7, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers - 
Dissertation Items/AutomatedClusteringProject/Cluster7TestResults.csv")
    write.csv(Cluster8, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers - 
Dissertation Items/AutomatedClusteringProject/Cluster8TestResults.csv")
    write.csv(Cluster9, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers - 
Dissertation Items/AutomatedClusteringProject/Cluster9TestResults.csv")
    write.csv(Cluster10, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers - 
Dissertation Items/AutomatedClusteringProject/Cluster10TestResults.csv")
    write.csv(Cluster11, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers - 
Dissertation Items/AutomatedClusteringProject/Cluster11TestResults.csv")
    write.csv(Cluster12, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers - 
Dissertation Items/AutomatedClusteringProject/Cluster12TestResults.csv")
  }

  if (c == 13) {
    write.csv(Cluster1, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers - 
Dissertation Items/AutomatedClusteringProject/Cluster1TestResults.csv")
```

```
    write.csv(Cluster2, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers - 
Dissertation Items/AutomatedClusteringProject/Cluster2TestResults.csv")
    write.csv(Cluster3, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers - 
Dissertation Items/AutomatedClusteringProject/Cluster3TestResults.csv")
    write.csv(Cluster4, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers - 
Dissertation Items/AutomatedClusteringProject/Cluster4TestResults.csv")
    write.csv(Cluster5, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers - 
Dissertation Items/AutomatedClusteringProject/Cluster5TestResults.csv")
    write.csv(Cluster6, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers - 
Dissertation Items/AutomatedClusteringProject/Cluster6TestResults.csv")
    write.csv(Cluster7, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers - 
Dissertation Items/AutomatedClusteringProject/Cluster7TestResults.csv")
    write.csv(Cluster8, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers - 
Dissertation Items/AutomatedClusteringProject/Cluster8TestResults.csv")
    write.csv(Cluster9, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers - 
Dissertation Items/AutomatedClusteringProject/Cluster9TestResults.csv")
    write.csv(Cluster10, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers - 
Dissertation Items/AutomatedClusteringProject/Cluster10TestResults.csv")
    write.csv(Cluster11, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers - 
Dissertation Items/AutomatedClusteringProject/Cluster11TestResults.csv")
    write.csv(Cluster12, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers - 
Dissertation Items/AutomatedClusteringProject/Cluster12TestResults.csv")
    write.csv(Cluster13, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers - 
Dissertation Items/AutomatedClusteringProject/Cluster13TestResults.csv")
  }

  if (c == 14) {
    write.csv(Cluster1, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers - 
Dissertation Items/AutomatedClusteringProject/Cluster1TestResults.csv")
    write.csv(Cluster2, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers - 
Dissertation Items/AutomatedClusteringProject/Cluster2TestResults.csv")
    write.csv(Cluster3, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers - 
Dissertation Items/AutomatedClusteringProject/Cluster3TestResults.csv")
    write.csv(Cluster4, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers - 
Dissertation Items/AutomatedClusteringProject/Cluster4TestResults.csv")
    write.csv(Cluster5, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers - 
Dissertation Items/AutomatedClusteringProject/Cluster5TestResults.csv")
    write.csv(Cluster6, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers - 
Dissertation Items/AutomatedClusteringProject/Cluster6TestResults.csv")
    write.csv(Cluster7, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers - 
Dissertation Items/AutomatedClusteringProject/Cluster7TestResults.csv")
    write.csv(Cluster8, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers - 
Dissertation Items/AutomatedClusteringProject/Cluster8TestResults.csv")
    write.csv(Cluster9, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers - 
Dissertation Items/AutomatedClusteringProject/Cluster9TestResults.csv")
```

```
    write.csv(Cluster10, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers - 
Dissertation Items/AutomatedClusteringProject/Cluster10TestResults.csv")
    write.csv(Cluster11, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers - 
Dissertation Items/AutomatedClusteringProject/Cluster11TestResults.csv")
    write.csv(Cluster12, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers - 
Dissertation Items/AutomatedClusteringProject/Cluster12TestResults.csv")
    write.csv(Cluster13, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers - 
Dissertation Items/AutomatedClusteringProject/Cluster13TestResults.csv")
    write.csv(Cluster14, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers - 
Dissertation Items/AutomatedClusteringProject/Cluster14TestResults.csv")
  }

  if (c == 15) {
    write.csv(Cluster1, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers - 
Dissertation Items/AutomatedClusteringProject/Cluster1TestResults.csv")
    write.csv(Cluster2, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers - 
Dissertation Items/AutomatedClusteringProject/Cluster2TestResults.csv")
    write.csv(Cluster3, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers - 
Dissertation Items/AutomatedClusteringProject/Cluster3TestResults.csv")
    write.csv(Cluster4, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers - 
Dissertation Items/AutomatedClusteringProject/Cluster4TestResults.csv")
    write.csv(Cluster5, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers - 
Dissertation Items/AutomatedClusteringProject/Cluster5TestResults.csv")
    write.csv(Cluster6, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers - 
Dissertation Items/AutomatedClusteringProject/Cluster6TestResults.csv")
    write.csv(Cluster7, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers - 
Dissertation Items/AutomatedClusteringProject/Cluster7TestResults.csv")
    write.csv(Cluster8, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers - 
Dissertation Items/AutomatedClusteringProject/Cluster8TestResults.csv")
    write.csv(Cluster9, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers - 
Dissertation Items/AutomatedClusteringProject/Cluster9TestResults.csv")
    write.csv(Cluster10, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers - 
Dissertation Items/AutomatedClusteringProject/Cluster10TestResults.csv")
    write.csv(Cluster11, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers - 
Dissertation Items/AutomatedClusteringProject/Cluster11TestResults.csv")
    write.csv(Cluster12, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers - 
Dissertation Items/AutomatedClusteringProject/Cluster12TestResults.csv")
    write.csv(Cluster13, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers - 
Dissertation Items/AutomatedClusteringProject/Cluster13TestResults.csv")
    write.csv(Cluster14, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers - 
Dissertation Items/AutomatedClusteringProject/Cluster14TestResults.csv")
    write.csv(Cluster15, row.names=TRUE, "C:/Users/Eric/Documents/Rutgers - 
Dissertation Items/AutomatedClusteringProject/Cluster15TestResults.csv")
  }
}
```