ABSTRACT

Securities can be classified in multiple ways. One of the most common ways to classify securities is using Morningstar's label system. For instance a fund can be Large-Value or Small-Growth. However, these systems have inherit limitations that makes it hard to properly maximize diversification in the selection of a portfolio. In this thesis, we present a method to classify securities that allows flexibility in the number of groups as well as the underlying data on which we classify. The goal is to have a complete system, able to take any universe of securities as input and produce a high-quality classification of that universe of securities.

SAMMANFATTNING

Tillgångar kan klassificeras på många olika sätt. Ett av de vanliga sätten att klassificera tillgångar är med Morningstar's så kallade label system. Till exempel kan en fond vara Large-Value eller Small-Growth. De här systemen har dock medförda begränsningar som gör det svårt att maximera diversifieringen av en portfolio som väljs med de här systemen. I det här arbetet presenterar vi en metod för att klassificera tillgångar som tillåter flexibilitet i antalet grupper så väl som datan som används som grund för klassificeringen. Målet är att ha ett komplett system som kan ta ett universum av tillgångar som indata och producera en klassificering av alla tillgångarna i det universumet som är av hög kvalité.

ACKNOWLEDGEMENTS

First I would like to thank my parents for inspiring me to pursue higher education. I would like to thank my grandfather for inspiration and contemplation over the fact that we live in a society where people can go from bricklayer to "Master of Engineering Physics" in two generations. I would also like to thank my girlfriend for spending many evenings with me in the library throughout my education as well as the support during the writing of this thesis.

Finally, I would like to acknowledge the contributions of Richard C. Dunne, CEO of Bdellium, to this project. Without his vision and help this project would never have taken place.

CONTENTS

1 BACKGROUND

Bdellium is a company that develops and provides products and services mainly for the financial industry. In its portfolio of services are such things as a retirement plan optimizer and a way to assess Qualified Default Investment Alternatives (QDIAs) for the American retirement plan market.

This thesis is part of a research effort from Bdellium targeted at developing a new way to classify securities and subsequently choose an investment portfolio.

This thesis will lay the foundations for this project in implementation and research development. The aim of the project is twofold.

- To document and theoretically define which algorithms to use for grouping a universe of securities intended for use in creating an investment portfolio.
- To implement the methods described, in Haskell, such that given a set of data for securities and parameters such as which similarity measure to use and how many groups to have, the program outputs a clustering for the securities.

Upon the work of this thesis project you can continue to build multiple interfaces as well as different kinds of outputs, like desktop applications, interactive graphs or PDF reports detailing the results. These are however not central to the theory being developed and are much more tied together with product development. As such, no work will be put towards those areas in this thesis.

This report will take you (the reader) through the motivation for building a tool like this, including how investment classification is done traditionally. It will present the raw data used as input for the method and of course the main bulk of the report will be explanation of the methodology itself.

Since a major part of this project is software development, there will be a section on why Haskell has been chosen as the implementation language as well as some general discussion about application modelling learnt throughout the development of this tool. Unfortunately, since the code for this thesis will go towards a proprietary product, it is not possible to review all the source code for the implementation of the various algorithms throughout this thesis.

We start by introducing some background knowledge to finance, basic things that we need in order to efficiently reason about what we want to achieve in our methods.

1.1 RISK AND DIVERSIFICATION

Risk in the financial industry can be defined in many ways. However, when talking about risk it is generally defined in terms of the variance (or standard deviation) of the returns of the security. The consistency of returns of a security is obviously extremely important in determining any possible future for that security. Any security available in the market carries some amount of risk. Academics and industry professionals alike are arguing the correct way

to calculate risk and many different measures have been developed. There is Beta, from the Capital Asset Pricing Model, Value at Risk, simple variance and a wide array of different measures of different mathematical complexity.

One thing that both academics and professionals can agree on is that in order to minimize risk, we should diversify our holdings between multiple securities. In other words, diversification is *one* way to reduce risk.

This concept was mathematically introduced in 1952 by Harry Markowitz [Markowitz, 1952], the theory is now called Modern Portfolio Theory (MPT) or Mean Variance Optimization. MPT has been widely adopted in finance, and while there are still discussions and research on the topic, the basics are simple.

As a simplified example, say that you have two businesses, Company A and Company B, that have seasonal performance. One does well in summer and bad in winter, the other does bad in summer and well in winter. Even if they have the exact same risk profiles overall, investing in both of them as opposed to only one will reduce your risk to 0. The two companies have a correlation of -1. In reality, you're unlikely to find correlation of -1, but we *do* wish to find securities that are as un-correlated as possible, if we want to reduce risk as much as possible. The real essence of this theory is that it says by adding securities to our portfolio we can reduce volatility without reducing expected return.

MPT shows that you don't even need negatively correlated securities to reduce your risk, as long as you add a security that is not perfectly correlated with one you already hold, you will reduce your variance by some amount. As the name Mean Variance Optimization implies, there is an optimization process involved as well that will create weights between your list of securites such that you get the maximum amount of returns for the minimum amount of risk. If we allow negative weights (but they must still sum to 1) that is called holding a security short. It can be shown that the optimal portfolio will always hold half of the securities short as the number of securities goes up [Levy and Ritov, 2011].

The problem with MPT is that you need some input list of securities to create your portfolio which you run the optimization on and there is no method of control for the concentration of the allocations to the funds in that input list. In other words, the objective function of mean-variance optimization is to minimize portfolio volatility, not optimize diversification. Ideally, the input list of securities should be as different from each other as possible, so that we control diversification as a separate process to mean-variance optimization.

This is a very hard problem to solve because of the sheer amount of securities available.

With modern computing technology however, we can calculate pairwise dissimilarity between all securities and then analyze these results to pick a shortlist of investments that are as different from each other as possible. This is what clustering securities is about.

We can define dissimilarity between securities in many ways, it could for example be simple correlation. However, the more risk factors we describe in the dissimilarity, the better our resulting clustering will become. It is also possible for an investment analyst to input subjective measures of dissimilarity into the clustering, should they wish to utilize their own wisdom in the selection of securities.

1.1.1 Definition of Value at Risk

In order to clarify a little bit what financial Risk is about and to activate the finance mindset, we will look at a simple example of VaR, which is one of the easiest risk measurements.

VaR is a measure of market risk that a security or portfolio is exposed to. The purpose is to measure how much value we could loose on any given day if we have money invested in the security.

The VaR calculation is based on the fact that you know the distribution of returns. It is often assumed that security returns follow a Normal distribution, that is, that they are completely random whose values are spread out along a bell-shaped curve. In the following examples the assumption of a Normal distribution is made. VaR will be the 99th percentile of the returns, because it is a normal distribution, we know that the 99th percentile will be at 2.32 times the standard deviation for the distribution. The VaR is then defined as

$$VaR_T \approx 2.32\sigma_T. \tag{1.1}$$

Where T is the time period used for calculating the standard deviation. Usually you calculate the standard deviation of the returns for a single day. This is then the value that you have a 1% chance of loosing every day.

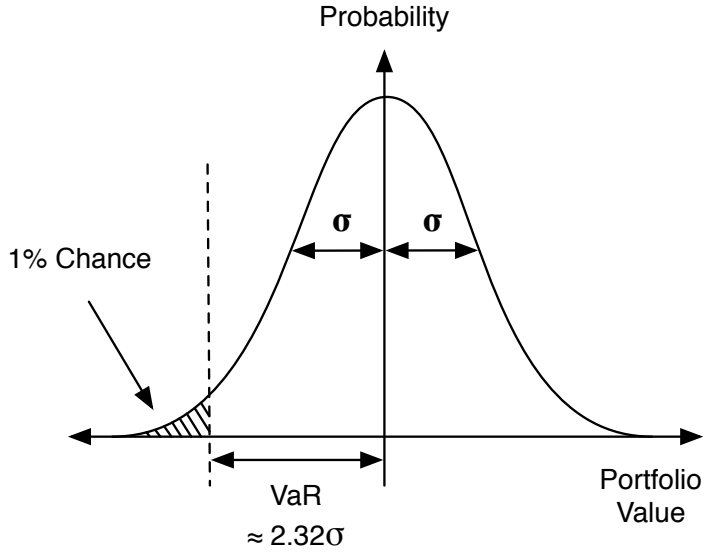


Figure 1: Illustration of the VaR on the normal distribution.

Example 1.1.1. Example VaR for an arbitrary investment.

Lets say you have invested in Company A that has had the following hourly dollar returns over the previous day, assuming the stock market was open for 8 hours.

$$\begin{bmatrix} -5.139 & 3.765 & 5.929 & -3.509 & 5.281 & 5.921 & 2.096 & -7.376 \end{bmatrix}$$

The hourly standard deviation σ_{hour} for those returns are $\sigma_{hour} = 5.395$. If you had invested your money to buying 10 stocks of Company A, then the standard deviation for your investment portfolio would be 10 times the standard deviation of Company A. However, to calculate the daily VaR we need the standard deviation for the daily returns, not the hourly. Since the daily return is the sum of 8 hourly returns

$$r_{day} = \sum_{n=1}^{8} r_{hour,n}$$

we get that the variance for the daily returns is $\sigma^2_{day} = 8 \cdot \sigma^2_{hour}$ and thus $\sigma_{day} = \sqrt{8} \cdot \sigma_{hour} = 15.259$

So your daily VaR would be $VaR = 2.32 \cdot 10 \cdot 15.259 = 354.001$. In other words, there would be a 1% chance that you will loose approximately $354 the coming day.

This methodology can be expanded to an investment portfolio with multiple investments, and then the correlation between investments come into play. If one stock goes up, another might go down and thus offset the standard deviation. Showing how to do this is out of scope for this report. But it is one of the most valuable tools in use by banks today to make sure that they are not taking too large risks [Marrison, 2002].

1.2 TRADITIONAL CLASSIFICATION

Our main motivation for wanting to build a system that automatically groups securities such that the groups are as diverse as possible is twofold. There is simply too much data for a human being to observe and manage themselves, so they need some sort of classification system to partition the set of available securities such that they can focus their efforts on picking good investments from the different classes.

There exists several of these classification systems in use today, the most popular one being invented by Morningstar. However, these systems usually come with several limitations or flaws, so the second part of our motivation is to come up with something to replace or improve the existing systems.

As mentioned, the most prevalent classification system in use today was designed by Morningstar, Inc. Morningstar is a provider of investment research, investment data and a whole host other financial products. They supply several major news network with real-time stock data as well as provide their famous 5-star rating system for funds. They have historical data and classification data on 385,000 investment offers as well as real-time market data on more than 8 million investment instruments in total.

Since their system of classification is so widespread, we will focus on using their system as the basis for comparison to what is developed in this report. Morningstar can classify securities into many categories, but for comparison we will look at what they call their Style Box[Morningstar, 2012]. This method is what they use to classify US domestic stocks and US domestic stock mutual funds.

The style box categorizes every security into one of 9 groups. An illustration of the grouping can be seen in Figure 2.

What it represents is a categorization on two factors. The first is company size, with the range Large, Medium and Small. The second factor is investment valuation, or investment style. The investment style is based on a growth score and a value score, calculated by Morningstar. Growth and Value are common terms in finance and so this categorization is based on a mix of both of them. The growth score is based on long-term projected earnings relative to the other securities in its size, it also based on a combination of historical earnings growth, sales growth, cash-flow growth and book-value growth relative to the stocks in its size range.

The value score is based on price divided by projected earnings relative to stocks in its size, as well as a combination of price-to-book, price-to-sales, price-to-cash flow and dividend yield, all relative to stocks in its size.

Both the growth score and the value score lie in the range of 0 to 100. To then determine the category for a stock, they take the growth score minus the value score. If the result is strongly

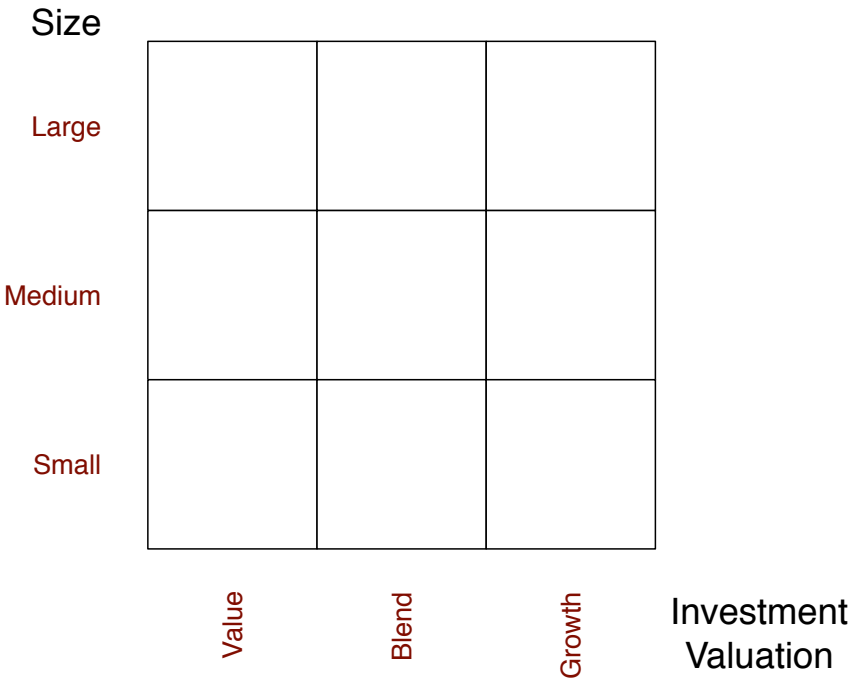


Figure 2: Illustration of Morningstar Style Box with labels.

negative, it is classified as a Value stock, one with strongly positive score is assigned growth stock. Finally, the remainder of the stocks will be classified as a blend.

This brings us to the first problem with this classification structure. "The remainder" between strongly negative and strongly positive is very ill-defined. Another very strong ambiguity is how to define company size. In fact Morningstar has chosen rather arbitrary limits where large-cap is the top 70% of the market, mid-cap is the next 20% and small-cap is the next 3%. Whatever falls outside is called micro-cap and isn't a part of the normal style box. For anyone interested in digging the methodology further, there is a full description of this system, which is essentially a holdings-based categorization of securities, provided by Morningstar [Kaplan, 2003].

While their underlying methods for calculating investment style and company size might be good in the sense of producing diversification and could yield groups with diversified results, they are ultimately forced to make compromises in the grouping such that they are able to fit everything within the grid you see in Figure 2.

When selecting investments from the Style Box, in order to diversify, it is common to try to choose investments from the different boxes. However, another problem arises when you want more or less than 9 securities to invest in. The 9 groups are set up to be as diverse as possible within their scheme, but if you only want three investments, which three groups are the most diverse? If you want 12 investments, which groups do you pick two securities from? It is easy to see that we loose the simplicity of diversifying when we want any number other than 9 securities.

With the method proposed by this thesis, you can choose the number of groups you want to maximize diversification between as well as let the algorithm choose an optimal number of groups.

1.3 CREATING AN INVESTMENT PORTFOLIO

How to create an investment portfolio is probably as widely debated as religion. There is no fixed method or optimal way to do it, but not for lack of trying to find one.

If you are to generalize at all, there seems to be one approach which is quite common. It is a two-step process, although these steps are thought out by the author of this thesis and are perhaps not explicitly thought of.

Step one is to derive some list of investments in which we want to put our money. How people select investments are usually a very personal thing, it could be from word-of-mouth alone or they could be selected in some manner from the Morningstar Style Box or any other of an infinite amount of ways.

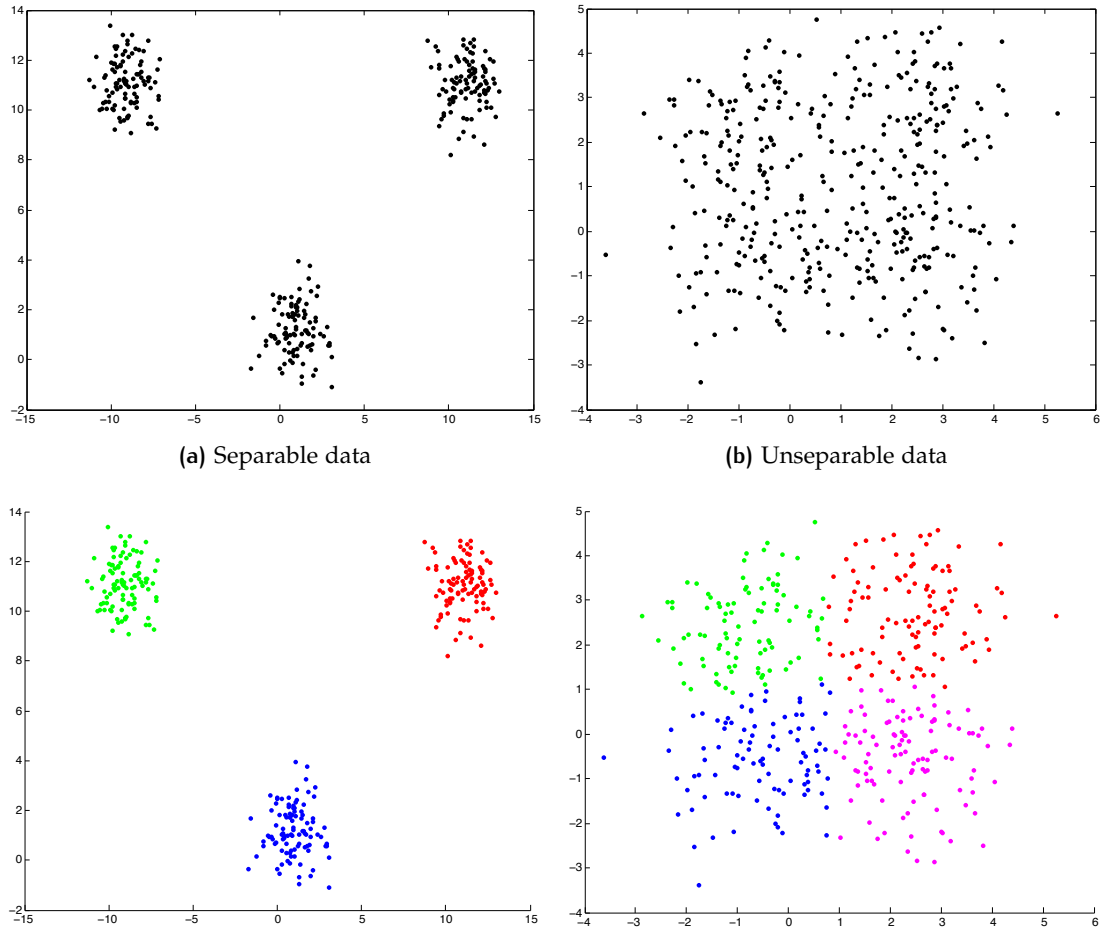
When one has a list of investments one believes in, in order to decide how much money to put in each, you perform a Mean Variance Optimization in order to get the best return for the amount of risk you want to take. There might be some other processes involved in finding the weights for the different securities, as Mean Variance Optimization might not be thought of as the end-all optimal way to create your portfolio. But in general, there are these two steps.

The problem with the situation today is that the financial world has grown up with categorization of securities like the Morningstar Style Box. So little to no consideration is taken to optimize your selection of securities in terms of their pairwise correlations. The Mean Variance Optimization will try to minimize risk, but it can only work with what has been given. If you give better input, you can get lower risk with perhaps the same returns.

Our method of clustering aims to make an optimization process a part of the selection of securities as well. Optimizing for correlation and all other dissimilarity factors we wish to consider.

2 CLUSTER ANALYSIS

The goal of cluster analysis is to, by some means, segment a set of data into groups such that the members of a group is more related to each other than to the elements of a different group.



(a) Separable data

(b) Unseparable data

(c) Grouped with three groups

(d) Grouped with four groups

Figure 3: Example data plotted as ungrouped points and grouped clusters.

Figure 3 shows the extremes of two situations that we might run in to. Figure 3a shows three clearly separable groups where if we saw this illustration we could clearly circle the clusters and find an optimal grouping by hand. We want our algorithm to also find the optimal grouping in such a case. Figure 3b shows a much more difficult scenario where we could not create a good clustering by hand because the data is not clearly separable. It is also unclear as to exactly how many groups we should have. Finding the optimal number of groups is a separate problem and is discussed more in Section 3.

Figure 3c and Figure 3d shows the results of applying our grouping algorithm PAM (2.5) to the ungrouped example data.

There are multiple ways to segment data into groups. The most common method is called K-means, which requires us to have data-points available in a numeric format (vector) such that we can calculate distances between a data-point and some other given point in the vector-space. K-means is discussed further in Section 2.3.

A different way to create clusters from data is with a method called K-medoids. In K-means an arbitrary point in the vector-space is selected as the center point for the group. In K-medoids we choose one of the data-points, which we call object, as the center point and then instead of calculating distances to the center point we calculate dissimilarity. This means that in order to find a grouping we only need one measure of dissimilarity between each pair of objects. This is explained further in Section 2.4.

2.1 SIMILARITY AND DISSIMILARITY

Throughout this thesis there are references to both similarity and dissimilarity. In essence, one is the inverse of the other. However, similarities and dissimilarities might have different value ranges in some cases.

In this thesis the most common measure of similarity is correlation. Taking that as an example, correlation can range between $[-1, 1]$. However, whether 0 is the most dissimilar or -1 (and perhaps 1) is the most dissimilar could depend on the application. One could imagine a situation where 0 correlation means the most dissimilar and -1 still hints at some similarity since they are in fact correlated (albeit negatively). However, when talking about securities, -1 means that when one security fails the other succeeds, and that is as dissimilar as possible.

When comparing similarity values to dissimilarity values there needs to be some qualitative assessment on how to convert between the two values. In the example of correlation for securities, we want -1 similarity to be 1 in dissimilarity and we want 1 in similarity to be 0 in dissimilarity, as they are exactly the same. We see that we can achieve that with the following formula.

$$d = (1 - c)/2 \tag{2.1}$$

This directly maps all correlation similarity values to dissimilarity values.

2.2 MULTIDIMENSIONAL SCALING

The goal of Multidimensional Scaling (MDS) is to represent a set of points $x_i \in \mathbb{R}^p$ as a set of points z_i in lower-dimensional space $\mathbb{R}^k$ $(k < p)$.

The distance between the observations x_i and x_j is called d_{ij} and this can be defined as any distance. If the observations are known Euclidean distance is often taken, however often the observations themselves are not known, only the distances. This is the case for our dissimilarity matrices used in the K-medoids method of clustering.

Multidimensional Scaling now seeks to minimize the stress function

$$S_M(z_1, z_2, ..., z_N) = \sum_{i \neq j} \left(d_{ij} - ||z_i - z_j||\right)^2 \tag{2.2}$$

This is known as *least squares* scaling and this is what is implemented in MATLAB's function `mdscale`. The problem is generally solved with a gradient descent algorithm.

It is essentially a minimization problem that attempts to find points z_i such that the Euclidean distances between them most closely resembles the real distances d_{ij}.

For very small distances and for a large number of points, this becomes a quite computationally heavy method of mapping higher-dimensional points to lower dimensions. If the distances between the points are smaller we need higher accuracy in order to properly represent them, likewise if there are more points it becomes more difficult to calculate the gradient in a gradient descent algorithm.

In this thesis MDS is used in several places to derive a set of 2D-points that could represent a dissimilarity matrix. While accuracy is important to display the groups correctly, the calculations are not the foundation of any analysis and are only for visualization purposes. Thus, no effort has been put into creating an implementation of MDS and the results achieved with MATLAB are good for this purpose.

2.3 K-MEANS

K-means together with Gaussian Mixture Models (not covered) are among the most popular methods of choice when it comes to cluster analysis.

The objective of K-means is to place k points in space that define the centres of the clusters. Every object is then grouped based on which centre point, or centroid, is the closest.

We thus want to find a clustering such that the total distance from every point to its centroid is minimized [Hastie, Tibshirani, and Friedman, 2011]. The overall objective function is

$$\min_{C,\{m_k\}_1^K} \sum_{k=1}^{K} \sum_{C(i)=k} \|x_i - m_k\|^2 . \tag{2.3}$$

The outer sum goes over all $k = \{1, ..., K\}$ clusters. The inner sum adds up the distances between an object x_i and its centroid for all objects in cluster k.

An iterative descent algorithm for solving 2.3 works by first guessing initial cluster centres, an arbitrary point in the vector-space defined by the problem, and then performing the following two steps until convergence is reached. Note that this is not guaranteed to find an optimal solution and may get stuck in local optima. Solving 2.3 optimally would indeed be an NP-hard problem and requires exponential time in general.

1. Given a cluster assignment C, replace the current centroids $\{m_1, ...m_k\}$ by the coordinate-wise mean of all clusters in C. Yielding k new means. This minimizes the total cluster variance with respect to the centroids.

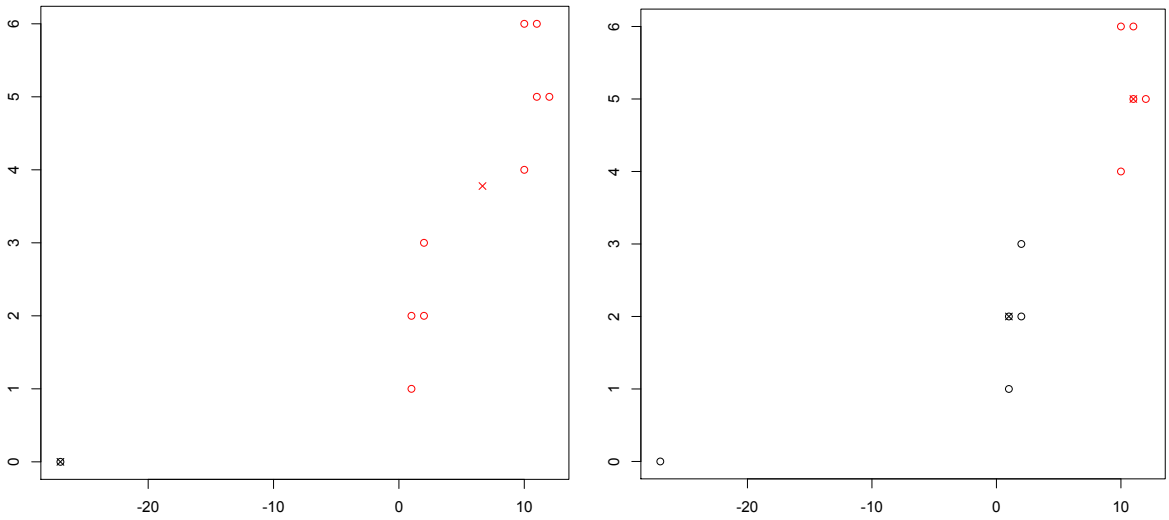
2. Assign every object to a cluster based on its closest centroid. Yielding a new clustering C where
$$C(i) = \arg \min_{1 \leqslant k \leqslant K} \|x_i - m_k\|^2 \tag{2.4}$$

In the second step, calculating the coordinate-wise average of all data-points, we need the actual coordinates of every point. It is easy seeing this applied to a 2D or 3D problem and it can be extended to multiple dimensions as well. However, we can also imagine a type of data that does not in fact exist in such a space at all.

Say we wanted to group countries based on peoples perception of them. Every country can not be given an objective vector of scalars that sets them apart from other countries. What we have to do is to compare countries pair-wise. We can say that country A is definitely different from country B, but not very different at all from country C. This would yield a totally different type of raw data for clustering.

To deal with this kind of non-coordinate based data we need a separate method. One such method is called K-medoids, and the example stated above will be treated in Section 2.4.

Another drawback of K-means compared to K-medoids is that we are using squared Euclidean distances in our calculations. Squaring the distance means we put more weight on outliers and the result is a lack in robustness for large outliers. For instance if we have two clearly defined groups and add a large outlier, one of the groups will be moved to that outlier and all the other data will be put into one group as seen in Figure 4.



(a) Using K-means with outlier

(b) Using K-medoids with the same data

Figure 4: The difference in robustness using K-means and K-medoids on data with large outliers.

2.4 K-MEDOIDS

In K-medoids we aim to solve the same objective function, Equation 2.3. However, instead of minimizing for the coordinate-wise cluster means $\{m_1, ..., m_k\}$ we now want to find the minimal cluster *medoids*.

A medoid is an object itself that is selected as a representative object of the cluster. The aim is to find a medoid that is as central to the cluster as possible.

By using an object from the actual observations we are not creating any new points in the space and thus we don't really need the coordinates at all, we only need a measure of similarity, or dissimilarity, between each pair of objects.

If the dissimilarity is taken to be squared Euclidean distance, we should get the same results as K-means with exception that the mean point and the medoid will not be in the exact same location. However, to achieve higher robustness the dissimilarity is often taken to be Euclidean distance without squaring it, if we are using distances at all.

As mentioned earlier, the data we apply K-medoids to is often not arising from calculating distances at all. As seen in Example 2.4.1.

Example 2.4.1. A good example from [Kaufman and Rousseeuw, 1990] is if we want to cluster countries based on similarity as perceived by people in a questionnaire. The data then becomes a similarity matrix, or dissimilarity matrix, depending on how we pose the question. Each data point in the matrix is the average of the result of the question "How dissimilar is country X to country Y on a scale from 1 to 10". Such a matrix can be seen in Table 1.

Table 1: Dissimilarity matrix arising from questionnaire data.

Index		0 BEL	1 BRA	2 CHI	3 CUB	4 EGY	5 FRA	6 IND	7 ISR	8 USA	9 USS	10 YUG
1	BRA	5.58										
2	CHI	7.00	6.50									
3	CUB	7.08	7.00	3.83								
4	EGY	4.83	5.08	8.17	5.83							
5	FRA	2.17	5.75	6.67	6.92	4.92						
6	IND	6.42	5.00	5.58	6.00	4.67	6.42					
7	ISR	3.42	5.50	6.42	6.42	5.00	3.92	6.17				
8	USA	2.50	4.92	6.25	7.33	4.50	2.25	6.33	2.75			
9	USS	6.08	6.67	4.25	2.67	6.00	6.17	6.17	6.92	6.17		
10	YUG	5.25	6.83	4.50	3.75	5.75	5.42	6.08	5.83	6.67	3.67	
11	ZAI	4.75	3.00	6.08	6.67	5.00	5.58	4.83	6.17	5.67	6.50	6.92

Say you now wanted to identify three groups of countries which are similar to each other, it is not immediately obvious how to do this from looking at the data in Table 1, even for this small of a sample.

Employing PAM, described in Section 2.5, the result we get is a set of medoids and a clustering vector, saying which cluster an object belongs to.

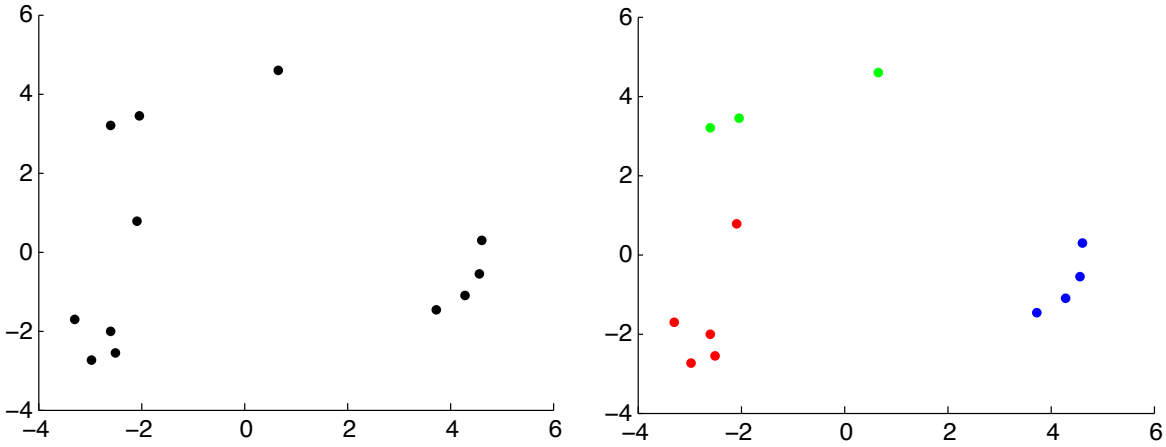
$$\begin{aligned} \mathbf{m} &= \{8, 11, 3\} \\ \mathbf{C} &= [2, 3, 1, 1, 2, 2, 3, 2, 2, 1, 1, 3] \end{aligned} \tag{2.5}$$

Though this might not be very informing in this case without employing some visualization tool to help us see what this grouping looks like. However, with nothing but a dissimilarity matrix it is not entirely obvious how to visualize this. Luckily, we can apply Multidimensional Scaling [Hastie, Tibshirani, and Friedman, 2011], which is easily obtained through the `MATLAB` function `mdscale`.

In Figure 5a we can see how the countries are visualized in two dimensions through MDS. In Figure 5b we see that the results in Equation 2.5 are perfectly valid groupings of the given dissimilarity matrix, the only country that is not very clearly defined is the one between the red and green groups, which if we dig in to the data little bit we can find out is Egypt.

2.4.1 K-medoids complexity

When trying to solve Equation 2.3 for K-medoids we can essentially see it as if we are given k arbitrary initial medoids and then have to move them around until we have found an optimal



(a) MDS visualization of the countries

(b) MDS visualization of the final grouping

Figure 5: Visualizing the results of Example 2.4.1.

configuration. Since the medoids fully define the clustering, we really only need to care about the medoids.

Now let's say you have a given set of medoids $\mathbf{m}$ of size k and there are N objects in total. You can move any medoid to any non-medoid and either get a better or a worse result. At each turn then, you can move the medoid label from any one of k objects to any one of $N-k$ objects. Or in other words, at each turn there are $k(N-k)$ possible moves. Let's call one of these moves a swap. We immediately see that this turns into a search-tree with a branching factor of $k(N-k)$.

Looking at every possible swap and expanding the tree for every swap would be a naive search strategy to find an optimal solution that amounts to a Breadth-First Search. BFS has a time complexity of $\mathcal{O}\left(b^d\right)$ where b is the branching factor and d is the depth of the optimal solution. In our case then $b = k(N-k)$ and d is unknown but will obviously grow with problem size, so it's proportional to both N and k. In other words, it is a terribly bad exponential-time search strategy.

In fact, solving K-medoids or K-means optimally is an NP-hard problem [Hastie, Tibshirani, and Friedman, 2011], meaning that there is no known solution that is less than exponential in time-complexity that solves the problem optimally.

For this reason, we need to apply an approximating search strategy, finding locally optimal but "good enough" solutions.

2.5 PARTITIONING AROUND MEDOIDS

Partitioning Around Medoids, or PAM, is a particular method used to find a good approximate solution to the K-medoids method of grouping. PAM will be the main clustering algorithm used in this project, it was introduced by [Kaufman and Rousseeuw, 1990].

To continue on the previous sections' comparison to search trees, PAM essentially amounts to a Best-First Search. It expands the tree only for the swap with the highest possible gain at the moment. If we want to find an optimal solution, we sometimes have to make "bad" swaps

in order to lead to the even better ones. Since we are only going to the best at every turn, we are absolutely not guaranteed to find the optimal arrangement of medoids.

However, not having the optimal arrangement of medoids does not necessarily mean that we have sub-optimal groupings. If the groups are well separated, the medoids can shift quite a lot within the cluster and still maintain the same ultimate grouping. That is, of course, not to say that the groupings will always be optimal either, but the situation is not as bleak as saying whenever we have sub-optimal medoids we have sub-optimal groups.

PAM is a two-phase solution algorithm, where in the first phase we find good initial medoids and in the second phase we try to move these medoids around such that we find a locally optimal solution.

An explanation of the two phases and what they entail follows in Subsection 2.5.1 and 2.5.2.

2.5.1 Phase I - Build

The goal of the build phase is to find good initial medoids.

First we define the contribution C_{ij} as

$$C_{ij} = \max\left(D_j - d(i,j)\right), 0\right). \tag{2.6}$$

Where D_j is the dissimilarity between j and its most similar medoid. $d(i,j)$ is the dissimilarity between object i and j.

This means that the contribution is positive if and only if j's most similar medoid is more dissimilar to j than i is to j.

In other words, if we added i as a new medoid, j would be more similar to i and we would lower the value of the objective function 2.3.

That means if we can first select *one* initial medoid, we can recursively add on new medoids until we have found as many as we want. This is what we do in the following two steps, which make up the Build phase of the algorithm.

1. The first medoid is the object for which the sum of dissimilarities to all other objects is as small as possible. This is computed by simple enumeration, calculating the sum of dissimilarities for all objects and then finding the minimum. This means we pick one medoid which would minimize 2.3.

2. Choose an object i which has not yet been selected as a medoid such that i maximises the sum of contributions C_{ij} over all non-selected objects j.

$$i = \arg\max_i \sum_j C_{ji} \tag{2.7}$$

   Add this object i to the list of medoids.

   If we have the desired amount of medoids then stop, otherwise repeat step 2.

2.5.2 Phase II - Swap

The swap phase attempts to find a swap between a selected object i and a non-selected object h.

In order to determine what swap is the best to perform, we need some measure of quality of a swap. We can define C_{jih} as the contribution for object j to the swap between i and h. We will define the contribution in terms of the amount it will change the objective function, and as such a good contribution is a negative contribution.

We again define D_j as the dissimilarity to the most similar medoid. We also add E_j which is the dissimilarity to the second-most similar medoid.

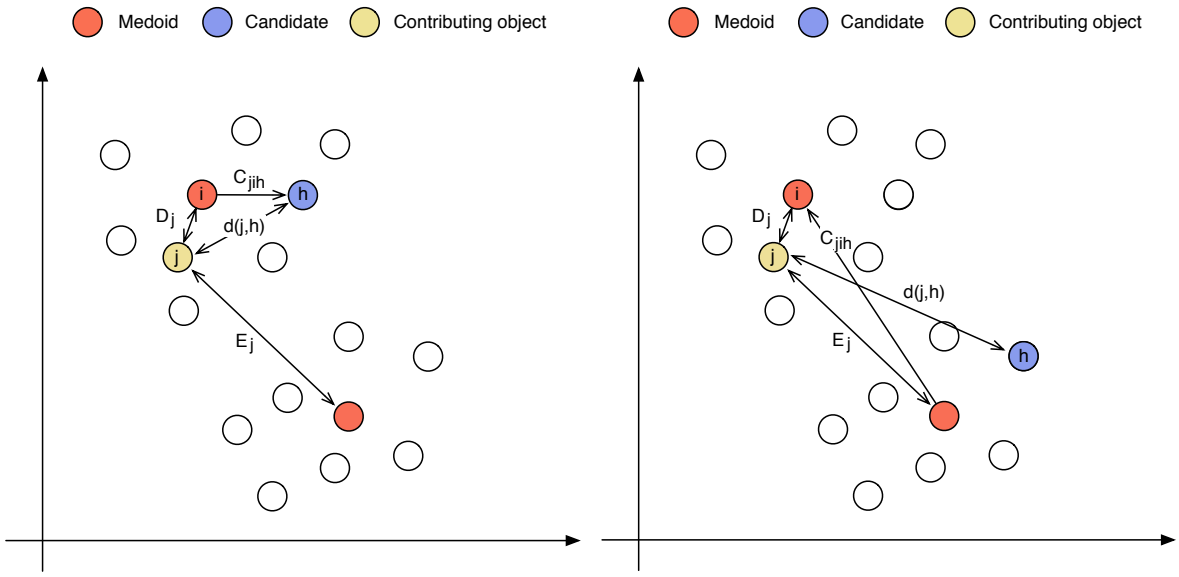
There can now arise three situations with regards to the contribution.

1. Object i is the most similar medoid, $d(j,i) = D_j$.

   If the dissimilarity from j to h is less than to the second most similar medoid, $d(j,h) < E_j$, then the contribution is the difference in dissimilarity, $d(j,h) - d(j,i)$, which might be negative or positive.

   If $d(j,h) \geqslant E_j$ the situation is guaranteed to be worse for j since it amounts to removing D_j and assigning E_j as the most similar medoid, so the contribution becomes $E_j - D_j$, which is always positive.

   These two situations are shown in Figure 6.



(a) h is *closer* than the second-closest medoid.

(b) h is *farther away* than the second-closest medoid.

Figure 6: Illustration of the case where i is the closest medoid.

2. When the $d(j,i) > D_J$ *and* $d(j,h) < D_j$ h, then the swap is guaranteed to be beneficial. We are then swapping a medoid that is more dissimilar to j than the current best for a medoid that is better than the current best, essentially moving the medoid "belonging" to j closer. The contribution will thus be the always negative value $d(j,h) - D_j$. This situation is shown in Figure 7.

3. The third situation is where none of the above applies, which means that both i and h are more dissimilar than D_j. When this is the case, j will not be affected by the swap at all and $C_{jih} = 0$. This situation is shown in Figure 8.

The above list can be summarized as in Equation 2.8.

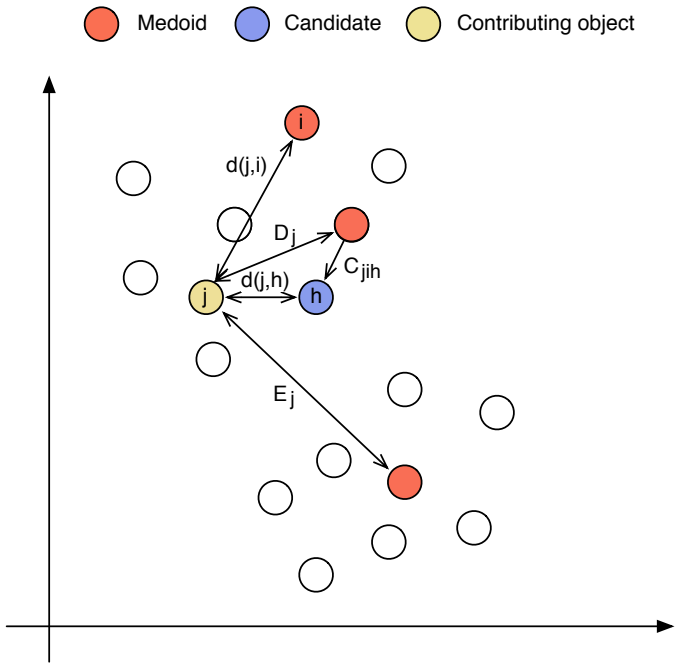


Figure 7: Illustration where i is farther away than the closest medoid, but h is closer.

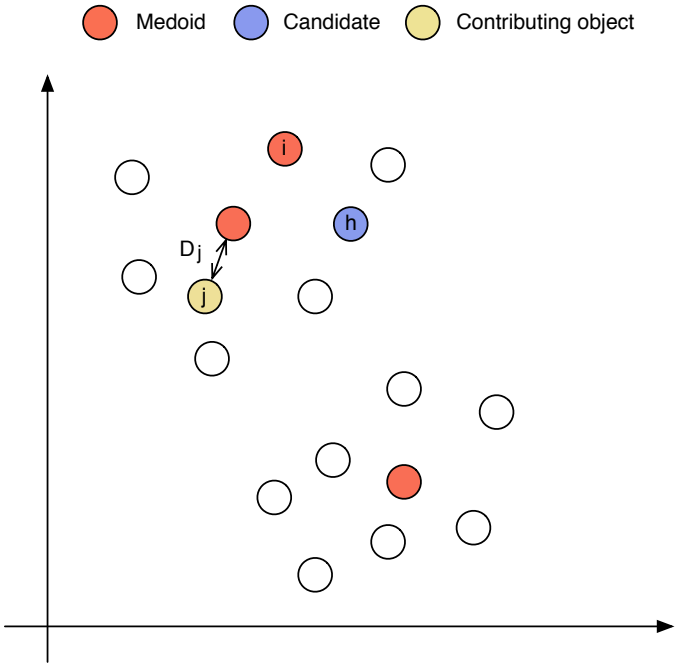


Figure 8: Illustration where both i and h are farther away than the closest medoid.

$$C_{jih} = \begin{cases} \begin{cases} d(j,h) - d(j,i) & \textbf{if } d(j,h) < E_j \\ E_j - D_j & \textbf{otherwise} \end{cases} & \textbf{if } d(j,i) = D_j \\ d(j,h) - D_j & \textbf{if } d(j,i) > D_j \textbf{ and } d(j,h) < D_j \\ 0 & \textbf{otherwise} \end{cases} \tag{2.8}$$

Now that we have defined the contribution, we can find the best swap by simply calculating the contribution for every j and summing up the values to get the total contribution for the swap

$$T_{ih} = \sum_j C_{jih}. \tag{2.9}$$

The swap phase is then to calculate T_{ih} for all possible swaps and select the swap for which this total contribution has the minimum value. The swap is only beneficial if the total contribution is *less than zero*, and thus the algorithm continues to recurse until we can not find any swap that has a total contribution lower than zero.

We have now found a final solution to our grouping problem.

2.6 FAMES

Described first by [Paterlini, Nascimento, and Traina Junior, 2011], FAMES represents the latest developments in speeding up k-medoids. FAMES stands for FAst MEdoid Selection and is a method for selecting a new medoid given a cluster. This means it can be integrated into any existing k-medoids algorithm and produce significant speedups. For instance we could replace calculating the contributions for all possible swaps in the Swap Phase of PAM. Instead we would calculate a new medoid using this much more direct approach and use that as the medoid to swap to.

According to the original article, both speed and quality are improved by using FAMES over PAM. However, my findings say that the quality of the clustering produced by FAMES is slightly worse or equal to PAM, but the speedup is indeed tremendous.

FAMES is a highly intuitive selection method for new medoids. Imagine drawing a line between two points at the outskirts of a cluster that are as far away from each other as possible. Essentially "one at each end". Then draw a hyperplane perpendicular to this line such that it divides the cluster in half, meaning half of the elements are on each side of the plane. Now pick the object most similar to the intersecting point as the new medoid, as it should be central to the cluster.

However, it is complicated somewhat by the fact that we can not calculate dissimilarity to any point in space, we can only calculate dissimilarity to other objects. Therefore, this is an approximate method as well, though you can easily imagine a situation where it is exact, for instance given a cluster in 2D where all the data points are on perfectly concentric circles with a point in the centre.

We start off the FAMES algorithm by selecting two points at the edges as far away from each other as possible. We do this by first randomly selecting any object. The first point is then the object furthest away from this object. The second point is the object furthest away from the point that we selected as the first point.

Next, the dataset is divided in half by comparing their differences to points one and two. We do this by projecting each element over the line between point one and two using the cosine law [Paterlini, Nascimento, and Traina Junior, 2011], where p_1, p_2 and p are point one, point two and the point to be projected respectively.

$$x_i(p) = \frac{d(p_1,p)^2 + d(p_1,p_2)^2 - d(p_2,p)^2}{2 \cdot d(p_1,p_2)} \tag{2.10}$$

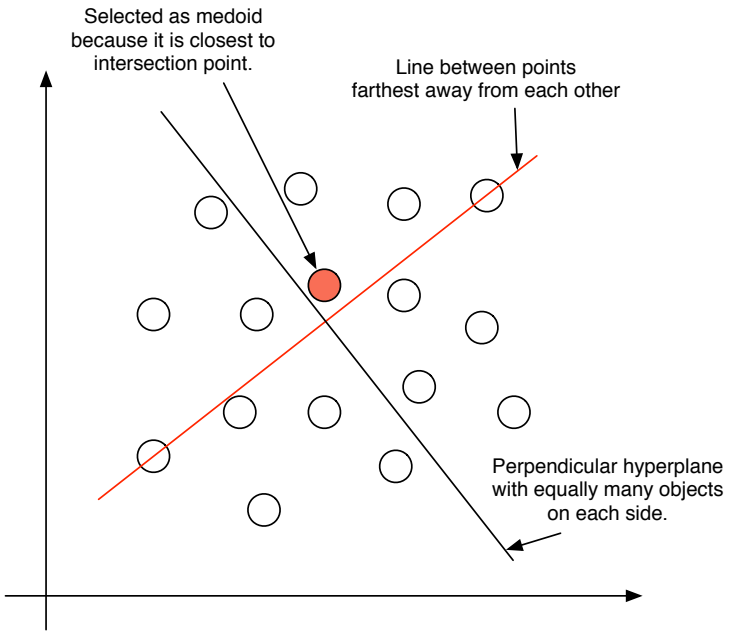


Figure 9: Intuitive illustration of the FAMES algorithm.

X is the vector of $x_i(p)$ applied to all objects in the cluster. We sort X and take the median value m, as our midpoint that splits the elements in half.

To find the new medoid r now, we choose the object that minimizes the sum of differences from p and m to p_1 and p_2.

$$r = \arg\min_p |d(p_1, p) - m| + |d(p_2, p) - (d(p_1, p_2) - m)| \tag{2.11}$$

Essentially we are trying to find the point where the four different distances shown in Figure 10 are as equal as possible.

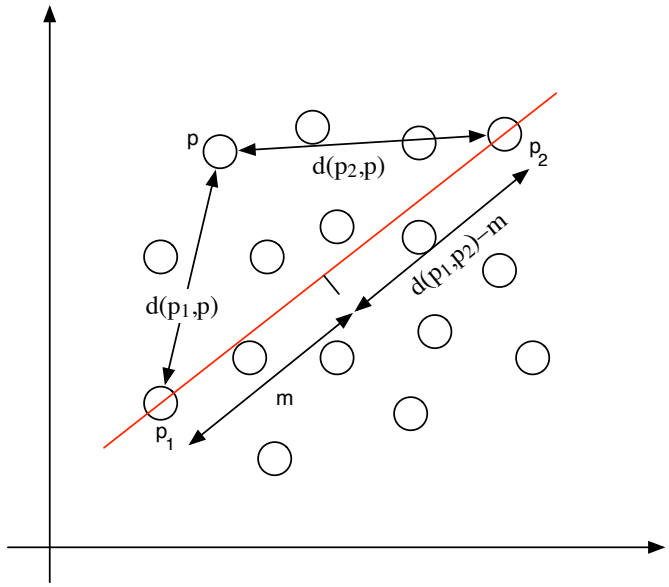


Figure 10: Showing the distances relevant for the minimization problem of Equation 2.11.

2.6.1 Comparing FAMES with PAM

Having implemented both PAM and FAMES in our Haskell platform we can compare the results of the two methods both in terms of end value of the objective function and the run-speed.

In order to have some significance to the test, we will need a few relevant datasets to test on. The parameters that are interesting to test are the size of the data set as well as the separability of different groups.

We create four different datasets to run our tests on. A set with 1000 objects and 10 relatively separable groups, a set with 1000 objects and 40 very dense groups as well as datasets with 2500 objects and 10 separable and 50 non-separable groups respectively. An illustration of these can be seen in Figure 11.

In running these tests every clustering was performed 5 times and the objective function and time has been averaged. Since FAMES can iterate possibly infinitely many times, the number of iterations was capped at 200. Of course, quality of results do not increase linearly with number of iterations and the improvement above 200 iterations was minimal. The same limit was imposed on PAM, but PAM usually finishes in about 20 to 50 iterations on the sample datasets.

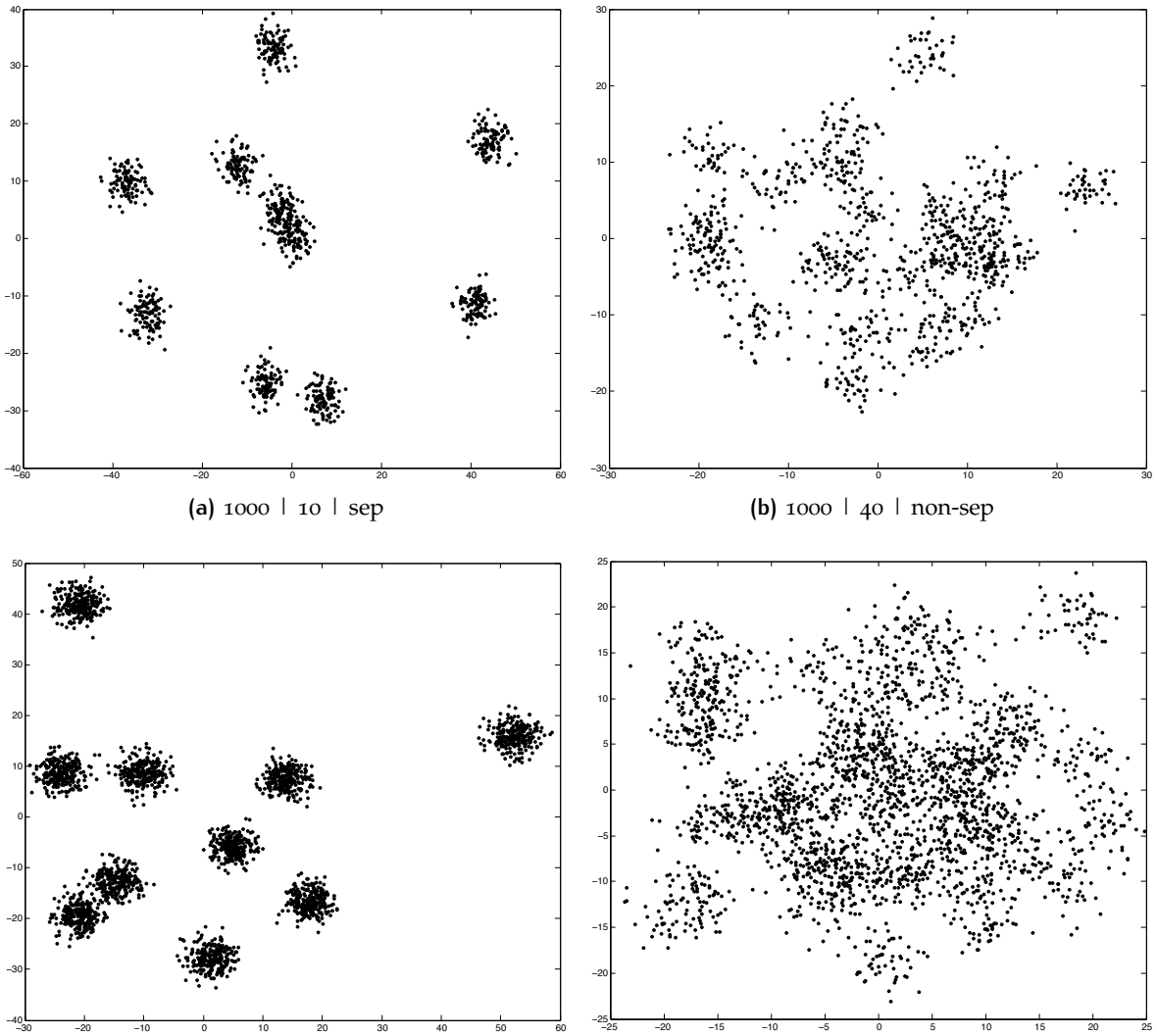
As we can see in Figure 12, the speed of FAMES is drastically much better than PAM. If we run FAMES with random initial medoids, which is the default, the average time to run on 2500 objects is around 5 seconds for 40 groups, while it is around 300 seconds for PAM.

The speed difference between separable and non-separable data seems statistically insignificant, meaning we can expect roughly the same execution time no matter what data we input.

We can also see that PAM Build is linear in the increase of k and PAM Swap is more than linear in the increase of k.

When looking at the quality of the clustering, a lower number for the objective function is better. In Figure 13 we can see that FAMES consistently under-performs in comparison to PAM. The error is particularly bad for small numbers of groups. And we can see that in the perfectly separable groups, picking random initial medoids and doing 200 iterations of FAMES produces horrible results. Though if we pick initial medoids with the BUILD phase of PAM, we get down to roughly the same difference as with the non-separable groups.

A qualitative study of the impact of this difference in objective function is hard to do. The difference could potentially become irrelevant when applying the final adjustment to the clustering as seen in Section 2.7.



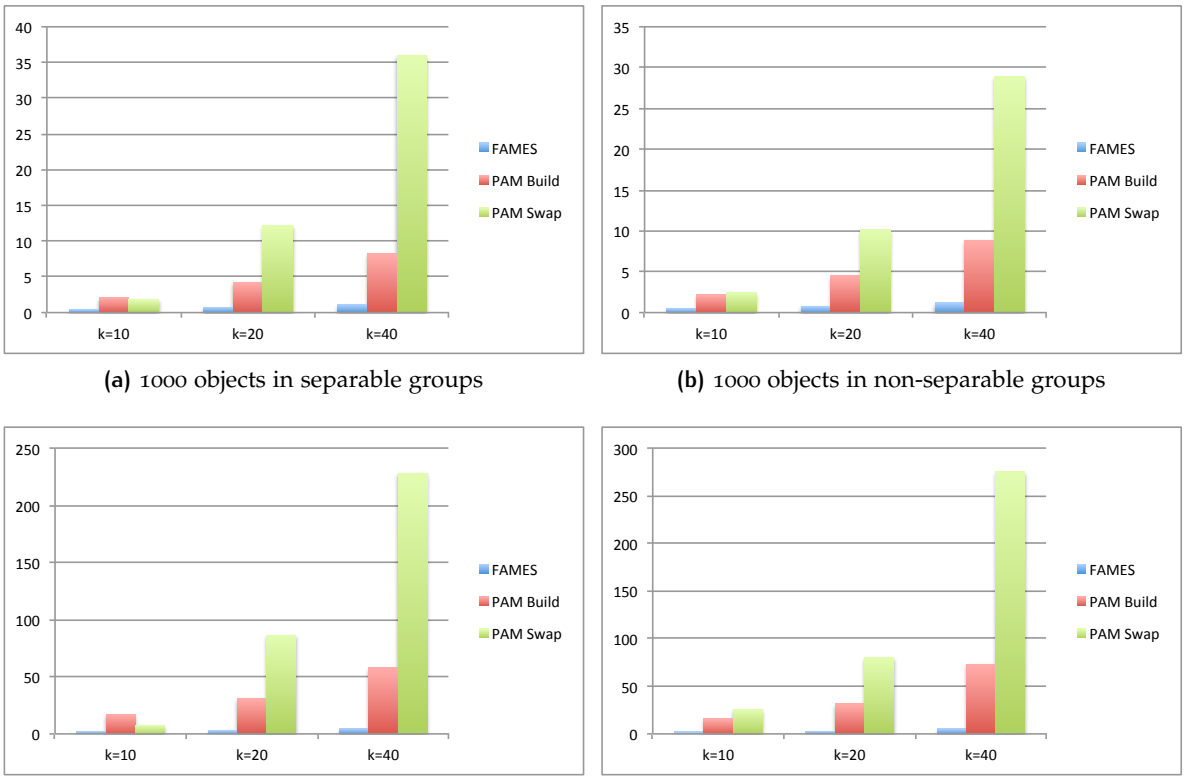
(a) 1000 | 10 | sep

(b) 1000 | 40 | non-sep

(c) 2500 | 10 | sep

(d) 2500 | 50 | non-sep

Figure 11: Different datasets. Notation is; Number of objects | real number of groups | separable or non-separable



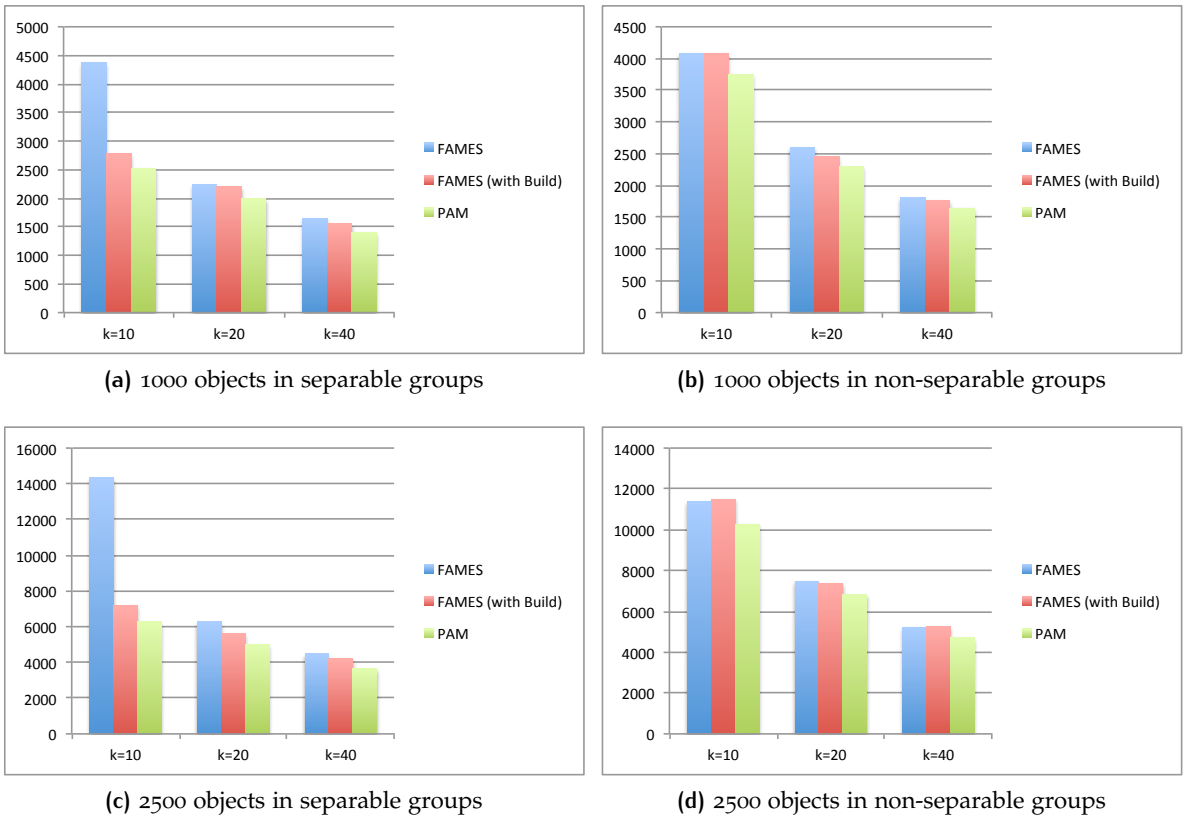
(a) 1000 objects in separable groups

(b) 1000 objects in non-separable groups

(c) 2500 objects in separable groups

(d) 2500 objects in non-separable groups

Figure 12: Run time for different data sets. Y-axis is speed and X-axis is the number of groups into which we divide our data set.



(a) 1000 objects in separable groups

(b) 1000 objects in non-separable groups

(c) 2500 objects in separable groups

(d) 2500 objects in non-separable groups

Figure 13: Objective function for the result of clustering different data sets. Y-axis is objective function value (total dissimilarity) and X-axis is the number of groups into which we divide our data set.

2.7 ADJUSTING THE RESULTS OF CLUSTERING

The difference between classical clustering like what we have described in the previous chapters and what we want to achieve for this tool is the following.

In classical clustering we are trying to minimize the total distance from every object in a cluster to the cluster medoid.

Here, we want to make sure that the average distance to every other point in the cluster is less than the average distance between this point and every other point in any other group.

To achieve this, we apply a very simple adjustment algorithm to the results of the clustering produced by PAM or FAMES. The algorithm is described by the steps below.

- Select an object and calculate the average distance from this object to every other object in the same group.
- For all other groups, calculate the average distance from the object to every object in the group.
- If there is a different group which has lower average distance then move the object to that group and start over from the beginning. If the object is already in the correct group, perform the algorithm for the next object.

It is important that we do not treat more than one object at a time, since moving an object would affect the calculations for every subsequent object. Therefore, we start the algorithm at the first object and move down the list, when we find an object that needs to be moved we start over from the beginning again.

This continues on until we hit the end of the list and can confirm that all the objects are in the correct groups to match our criteria.

3 OPTIMAL NUMBER OF GROUPS

When determining the optimal number of groups, we need to define some measure of optimality. The optimality should consider the clustering as a whole and not just look at the optimality of a single group, as we're trying to optimize for diversity.

We define quality measurements in Section 3.1 for a given clustering. These quality measurements are largely developed by Bdellium. With the quality in hand we try to select the group with the highest quality. How to do that selection is described in Section 3.2.

3.1 CLUSTER QUALITY MEASUREMENTS

3.1.1 Classification Quality (CQ) Index

$$\text{CQ Index} = 25 \cdot [(1 + \text{Avg Group Correlation}) + (1 - \text{Avg Other-Group Correlation})] \quad (3.1)$$

Where Group Correlation is defined as the *Average Correlation Within Own Group* and Other-Group Correlation is the *Average Correlation to Other Groups*. Below is described how to calculate Group Correlation and Other-Group Correlation for some group C. The CQ Index measurement is then calculated with the average Group Correlation and Other-Group Correlation measurements for all groups in the clustering.

To calculate Group Correlation we calculate all possible correlations between securities in the same group C and take the average of that value. Where N_C is the number of items in group C.

$$\text{Group Correlation} = \frac{1}{N_C} \sum_{a \in C} \sum_{b \in C} \rho_{a,b}, a \neq b \quad (3.2)$$

The average correlation to other groups is then defined as calculating the average correlation between an item in the group we're looking at and all other groups. We do that for all items in the group and subsequently take the average of all those averages. Where N is the total number of items in consideration.

$$\text{Other-Group Correlation} = \frac{1}{N_C} \sum_{a \in C} \frac{1}{N - N_C} \sum_{b \notin C} \rho_{a,b} \quad (3.3)$$

Now the result of Equation 3.1 is that we map both of the correlation values from $[-1, 1]$ to $[0, 2]$. Then we get a score in the range $[0, 100]$ by adding them and multiplying by 25.

Thus, the best score of 100 means that the group correlation is 1 and the other-group correlation is -1. This is the best possible diversification we could get.

3.1.2 InCorr ⩽ ExCorr

This quality measurement tells us how many securities are "misplaced". I.E. placed in a group in which the correlation to the rest of the members of the group is lower than if it was placed in a different group. To calculate the InCorr ⩽ ExCorr score, we take the following approach.

We first select a security in a group. We calculate its average correlation to other securities in its own group. Then we calculate the average correlation for this security to all the items in a different group. We do this for all groups. The maximum of the correlation to other groups is called ExCorr, and if the correlation to it's own group, InCorr, is less than ExCorr that means we have a sub-optimal grouping. Remember we want InCorr to be as close to 1 as possible and ExCorr as close to -1 as possible.

To get the InCorr ⩽ ExCorr score, we calculate InCorr and ExCorr for all securities and count the instances where InCorr ⩽ ExCorr. Thus, the lower measurement the better. To contrast this with CQ Index, InCorr and ExCorr are like Group Correlation and Other-Group Correlation, but calculated for a single security instead of for the clustering as a whole.

3.1.3 Grouping Variance Effect

The Grouping Variance Effect (GVE) aims to study the overall effects on variance from doing the grouping. For the purposes of first describing the math of the GVE, we will consider a portfolio of single securities.

We need to define what we mean by change in variance, specifically reduction of variance. Assume that we have a security x and a portfolio, p, consisting of x and $N-1$ other securities. We have a variance σ_x^2 for security x and σ_p^2 for the portfolio. The change in variance from investing in x to investing in the portfolio is then

$$\frac{\sigma_p^2}{\sigma_x^2} - 1. \tag{3.4}$$

According to mean variance optimization, we can find a portfolio such that this ratio is guaranteed to be less than or equal to 1. Thus Equation 3.4 is guaranteed to be less than or equal to 0.

We subtract by one to obtain a percentage change, so if the factor of change is 0.98 then the *reduction in variance for portfolio* p is -2%.

We now want to study the effects of going from a single security to a portfolio of securities, but without taking into consideration the effect of the added securities individual variances.

In other words, going from one security to two securities can cause a drop in variance, both from the fact that the second securities variance is lower and from the fact that the two securities are not perfectly correlated. What we want to study now, is only the effect caused by the correlation.

If we start with an investment in a single security, the portfolio variance equals the individual security's variance σ^2.

The Grouping Variance Effect is then a measure of the reduction in portfolio variance from simply adding more securities to a portfolio, no matter what the variance of those securities are.

The variance of a portfolio consisting of multiple securities is defined as the following

$$\begin{aligned}\sigma_p^2 =& W_1^2\sigma_1^2 + W_2^2\sigma_2^2 + \ldots + W_N^2\sigma_N^2 \\ &+ 2(W_1\sigma_1)(W_2\sigma_2)\rho_{1,2} \\ &+ 2(W_1\sigma_1)(W_3\sigma_3)\rho_{1,3} \\ &+ 2(W_2\sigma_2)(W_3\sigma_3)\rho_{2,3} \\ &+ \ldots\end{aligned} \tag{3.5}$$

Where W are the weights of the securities in the portfolio. σ_k is the standard deviation (volatility) of returns of security k. $\rho_{1,2}$ is the correlation between the returns of security 1 and security 2.

We note that the number of terms in Equation 3.5 that include $\rho_{a,b}$ equals the number of ways we can take 2 securities from N securities, or in other words "N choose 2", or

$$\frac{N\,(N-1)}{2}. \tag{3.6}$$

We want to remove the effect of each individual securities variance on the total variance. To do that, we set every securities individual variance to some σ^2. We also want to remove any effect that weighting can give us, so we set equal weighting for all securities $W = 1/N$. What we have then for the portfolio variance is

$$\sigma_p^2 = NW^2\sigma^2 + 2W^2\sigma^2\,(\rho_{1,2} + \rho_{1,3} + \ldots) \tag{3.7}$$

The average pairwise correlation is defined as

$$\overline{\rho} = \frac{2}{N\,(N-1)}\,(\rho_{1,2} + \rho_{1,3} + \ldots) \tag{3.8}$$

and thus we can rewrite Equation 3.7 using the average correlation as

$$\begin{aligned}\sigma_p^2 &= NW^2\sigma^2 + 2W^2\sigma^2\frac{N\,(N-1)}{2}\overline{\rho} \\ &= \frac{N}{N^2}\sigma^2 + \frac{N\,(N-1)}{N^2}\sigma^2\overline{\rho} \\ &= \frac{\sigma^2}{N}\,(1 + (N-1)\,\overline{\rho})\end{aligned} \tag{3.9}$$

Now if we start with a single security, we know that its variance is σ_s^2

We know that a portfolio with more than one security will have a σ_p^2 equal to Equation 3.9 above if the variances are fixed and the securities are equal-weighted.

This means that the reduction in portfolio variance between the original single security and the portfolio equals

$$\begin{aligned}\text{GVE} &= \frac{\sigma_p^2}{\sigma_s^2} - 1 \\ &= \frac{1}{N}\,(1 + \overline{\rho}\,(N-1) - N) \\ &= \frac{N-1}{N}\,(\overline{\rho} - 1)\,.\end{aligned} \tag{3.10}$$

Now to bring this back into the realm of grouping. What we're saying is that we treat each group as a security, where the pairwise correlation between groups is the average correlation of every item in group A to every item in group B. And the average correlation in calculating

the GVE is the average other-group correlation for all groups that we find in calculating the CQ Index as well.

What we're left with in the end is a measure of how much grouping the securities in N groups will reduce the variance, regardless of the variance of the individual securities that are put into the N groups.

3.1.4 Diversification Efficiency (DE) Ratio

The Diversification Efficiency ratio aims to measure the extent to which any portfolio achieves greater (negative) Grouping Variance Effect (GVE) than the grouping that delivers the largest GVE, after adjusting for the difference in the number of groups.

To calculate this, we want to take the ratio GVE_x/GVE_m where GVE_x is the GVE for our target group x and GVE_m is the maximum GVE found in all the different number of groups we have tested. This ratio measures how the GVE has increased or decreased in the target grouping compared to the max. Now we want to relate this to the *expected* ratio if the correlation between all the groups were the same regardless of number of groups. In other words, we assume that the only difference between the two scenarios is the number of groups and not the contents of the groups.

Then, the expected ratio becomes

$$\frac{\frac{N_x-1}{N_x}(\bar{\rho}-1)}{\frac{N_m-1}{N_m}(\bar{\rho}-1)} = \frac{N_m(N_x-1)}{N_x(N_m-1)} \tag{3.11}$$

We now want to see how the actual change in Grouping Variance changed in relation to the expected change in Grouping Variance if the only factor had been the number of groups.

$$DE = \frac{\frac{GVE_x}{GVE_m}}{\frac{N_m(N_x-1)}{N_x(N_m-1)}} = \frac{GVE_x}{GVE_m}\frac{N_x(N_m-1)}{N_m(N_x-1)} \tag{3.12}$$

This is our measure of how well group x performed in terms of increasing diversification compared to the maximum measured increase in diversification, when taking into consideration the number of groups.

In other words, one grouping might not have achieved the maximum diversification, but when taking into consideration the number of groups, it could have been more efficient in achieving what it did achieve by using a different number of groups.

3.2 SCORING AND RANKING DIFFERENT GROUPINGS

Now that we have defined a number of quality measurements for clusters, there remains the question of how to use these to determine the optimal number of groups. To do so, we apply a scoring method that Bdellium has patented called BdScore.

In order to determine the optimal number of groups we need the quality measurements for all possible groups we want to consider. In other words, if we want to find the optimal number of groups in the range 10 to 100, then we need to cluster the data with $k = 10, 11, 12, \ldots, 100$. We save the quality measurements for every clustering and then apply the scoring method.

The method is in principle very simple. We calculate normalized values, then compose those by some weighting function and finally score the composite normalized values from 0

to 100 where 100 is the best group and 0 is the worst. The groups in between will have a score proportional to the quality compared to the best, i.e. a grouping with score 50 is half as good, compared to the worst quality grouping, as one with score 100.

The scoring process is as follows

1. Calculate normalized values for all measurements. If we have a measurement x drawn from a set of measurements X (that are the measurements for all numbers of groups), then the normalized value

$$x_{norm} = \frac{x - \text{average}(X)}{\text{stddev}(X)}. \tag{3.13}$$

   This results in a value that is negative if it is worse than average and positive if it is greater than average.

2. Now that we have normalized measurements, we simply compose them by addition or subtraction, with some weight applied. whether to use addition or subtraction is a question of whether a large measurement or a small measurement is desirable. When it comes to the cluster quality measurements, CQ Index and DE Ratio are positive measurements (we want them to be large) while InCorr <= ExCorr and GVE are negative measurements (we want them to be as small as possible). Thus the composed value is $\mathrm{CQIndex} \cdot W_1 + \mathrm{DERatio} \cdot W_2 - \mathrm{InCExC} \cdot W_3 - \mathrm{GVE} \cdot W_4$. The weightings can really be chosen as desired, equal weighting is a good choice, but if you, for instance, want to try to maximize the diversification effect it makes sense to put some extra weighting on GVE.

3. Now that we have a range of composite values, we can turn these into scores. If y is a composite value drawn from all the composite values Y then the score is

$$\text{BdScore}(y) = 100 \cdot \frac{y - \min(Y)}{\max(Y) - \min(Y)}. \tag{3.14}$$

   After calculating the score for all groupings, we can trivially turn the score into a ranking.

   Example 3.2.1. Lets look at an example of calculating the normalized values and scores for 4 fictive measurements. We have the data as in Table 2. Measurements X_3 and X_4 are considered negative measurements and are subsequently subtracted when calculating the Composite Value.

Table 2: Example of applying scoring method to fictive data.

X_1	X_2	X_3	X_4	$X_{1,norm}$	$X_{2,norm}$	$X_{3,norm}$	$X_{4,norm}$	Comp. Val.	BdScore
0.82	0.16	741	-4.96	0.85	-0.32	0.57	-0.21	0.04	57
0.79	0.03	627	-9.83	0.77	-0.86	0.04	-1.71	0.39	100
0.81	0.26	954	-4.36	0.83	0.08	1.54	-0.02	-0.15	33
0.17	0.70	387	-0.19	-1.09	1.88	-1.05	1.26	0.14	70
0.09	0.05	377	-2.11	-1.35	-0.78	-1.10	0.67	-0.43	0

   We can see the power of the method as we can use any sort of measurement of any range, normalize the values and end up with an intuitively interpretable and comparable score.

When we apply this scoring process to our example of clustering for $k = 10, 11, \ldots, 100$, we would simply select the grouping with BdScore 100 as the most optimal grouping.

4 MULTI-PERIOD ANALYSIS

Multi-Period Analysis is a method developed and patented by Bdellium, used to serve as an improved way to calculate time-dependent variables such as return on investment.

Traditionally, to measure the success of a security one might look at the return on investment over a past period of time. It is however not particularly well-defined what period to look at and this poses a problem.

Looking back at a 1-year return can be vastly different from looking at 3-, 5- or 10-year returns. Usually you choose to look at one or possibly all of those intervals.

What if you're looking to invest only 3 or 6 months? Or you want the stock with the best 7-year return. Which 3-month period do you look at to decide the outcome? The latest 3 months is perhaps not the most representative. This is where Multi-Period Analysis comes in. We will see more examples of how returns are sensitive to the period chosen in Section 4.3.

The solution to the problem posed above is simple, let's look at all the 3-month periods possible, including overlapping periods, and use those to select or calculate a return that is representative of the multiple possible outcomes, this can be done by picking the median, calculating the average or to calculate some weighted average. That way we have all possible investment periods starting at any time represented. Taking this reasoning one step further leads to the full Multi-Period Analysis, which calculates a weighted average of all periods for all period lengths. This would give us a well-defined measurement that you can compare between securities.

4.1 MULTI-PERIOD ALGORITHM

Before we decide what we want to calculate we need to know which periods to use. The foundation of the Multi-Period Analysis algorithm is to find all possible periods to look at.

The only inherit minimum period length for Multi-Period Analysis is one period. However, Bdellium has chosen to base all analysis on a minimum of 12 months of historical data. As such, the minimum period we use in practice is 12 months. There is no maximum period length, we are merely limited to our own restrictions or to the maximum number of months available in the data.

Imagine now that we have 24 months of returns. The first period length is 12, and the first period is the first twelve months. To get the second period we move the entire time-interval one step down such that the period becomes the 12 consecutive months *after* the first month. This process is continued until we find the period of 12 months containing the last available data-point (the 24th month).

Now we have all 12-month periods, so we increase the period length to 13 and repeat the process. This is repeated until the period length is the same as the maximum length and for this period length there is only one period consisting of the entire data set. An illustration of this can be seen in Figure 14, where we start out with 12 orange bars of 24, and we see that there are 13 different periods of length 12. As we increase the period length, the number of

periods for that length goes down. The final period represented at the bottom of the figure covers the whole dataset.

What we now do with the data in each of the periods depends on our application. We could calculate the annualized monthly compounded return for each period and get a Multi-Period rate of return for a security as shown in Section 4.3. We could also do a period-by-period analysis, comparing every period in a security to the corresponding period for some other security.

The major application of Multi-Period Analysis in this project is using it to calculate a weighted average correlation between two securities that takes all data into account and is intended to produce a better comparative measurement than simple correlation. This is explained further in Section 4.4.

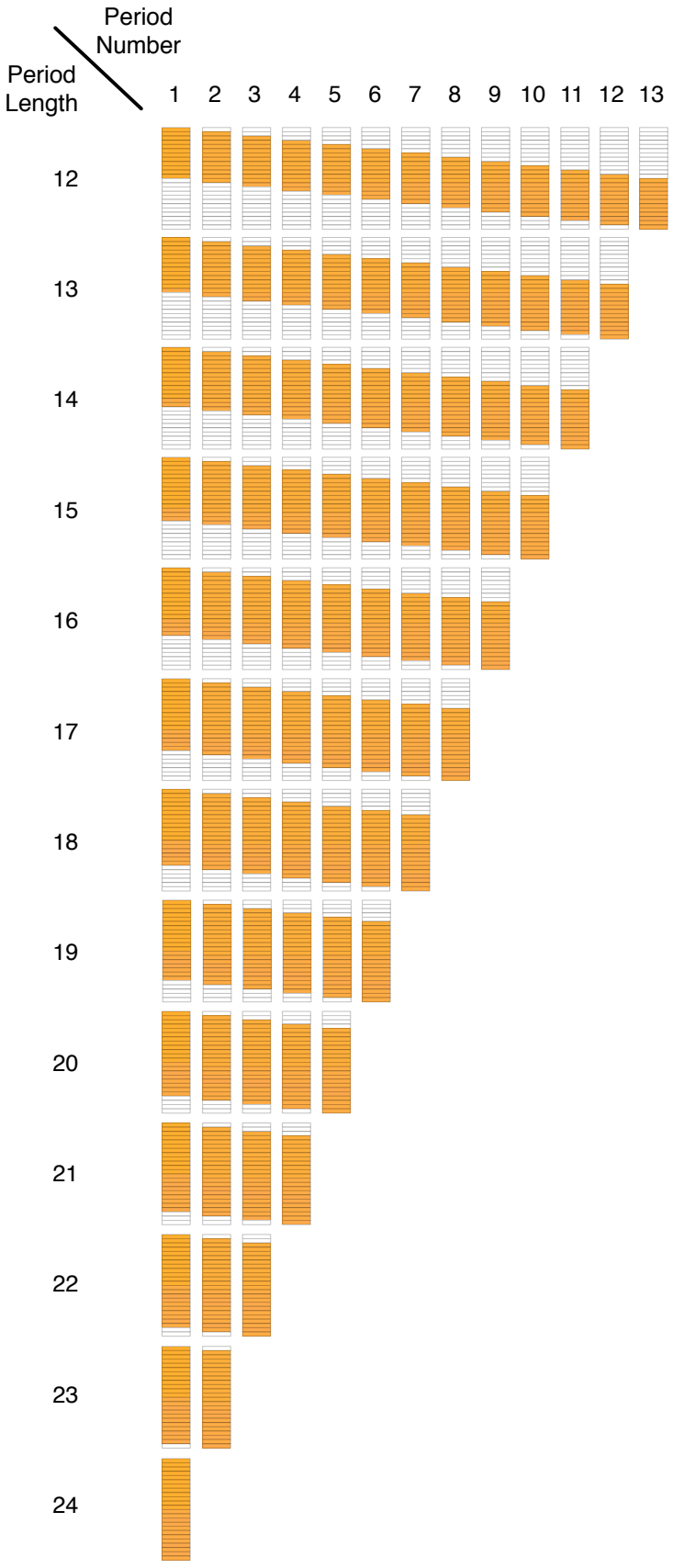


Figure 14: Illustrating all the possible periods with a minimum period length of 12 and a maximum period length of 24.

4.2 CHOOSING WEIGHTING FUNCTION

When combining the results of all the different periods we want to calculate a weighted average. For instance if we calculate an annualized return for every period we want to end up with an average yearly return.

There are many different weighting functions we can apply that all have different meanings and can be customized fully to the users requirements. The choice of weighting function is non-trivial and very subjective, and it will be left up to the analyst using the software to create the weighting function they feel is appropriate for their application.

However, in the absence of user requirements we want to have a default weighting function that produces a good result.

The default weighting function applied by Bdellium uses the logic that a period with greater length will produce a more stable and thus "better" result. The resulting weighting function is then

$$W(x_p) = \frac{p}{\sum_{l \in P} l \cdot n_l} \tag{4.1}$$

where W is the weighting function, x_p is the value to be weighted and corresponds to a period length of p. P is the set of all period lengths and n_p is the number of periods existing for period length p. With this weighting function, the weighting factor only depends on the period length, not on which exact period we are weighting. Note also that the weights must sum to 1.

Now the final weighted average value we want to reach through Multi-Period Analysis R is

$$R = \sum_{p \in P} x_p \cdot W(x_p) \tag{4.2}$$

4.3 MULTI-PERIOD ANALYSIS FOR RETURNS

As an example application of Multi-Period Analysis, we can calculate the weighted average return produced by our method for a sample security, the Google stock.

We have 36 months of returns listed in Table 4 that we will use as the data set for this example. The period lengths are thus 12 to 36 months in increments of one month.

We can divide the data set up into all the possible periods according to the algorithm described in Section 4.1. For each data set we then want to calculate the Annualized monthly compounded return, which we call $r_{k,p}$ for specific period k of period length (in months) p. The data (the returns) in the set for the period k is defined as R_k. The reason we are using monthly compounded returns instead of continuously compounded returns is simply that it is easier for people to understand if an external resource reviews the analysis methodology. There is no reason one can not use continuously compounded returns.

$$r_{k,p} = \left(\prod_{r \in R_k} r + 1 \right)^{12/p} - 1 \tag{4.3}$$

Applying this formula to each of the possible periods and then calculating the weighted average using the default weighting function described in the previous section, we get a Multi-Period return for Google displayed in Table 4.

In Table 3 there is also a comparison of our number and the 1-, 2- and 3-year rate of returns. As we can see, they yield very different (one could say unstable) results.

Table 3: Comparison of Multi-Period analysis looking at fixed periods for returns.

Multi-Period result considering 3 years of data	7.45%
1-year returns	26.64%
2-year returns	23.38%
3-year returns	14.06%

As we can see, the result from Multi-Period Analysis yields a significantly lower number than simply looking back 1, 2 or 3 years. The 1- and 2-year returns are inflated from the fact there there are 4 months, two in 2011 and two in 2010 where most of the returns are concentrated. We have four months of good returns mixed in with generally poor returns. Multi-Period return gives a better indication of this fact and it takes into consideration periods where these four months are not present. Our number also indicates that there have been bad periods, for instance if you were to invest for 19 months starting April 2010, you would have lost 12.62% of your money.

It is important to note that Multi-Period return is not intended to be an estimator of expected rate of return. When calculating an estimator, you assume that the values follow some (usually a normal) distribution and then you use the returns to calculate a mean and a standard deviation such that you can fully define the distribution. This requires the returns to be independently and identically distributed. In other words that the values in the sequence are assumed to be randomly selected from the target distribution. However, with Multi-Period returns, since we have overlapping periods, they are clearly not independent and as such can't be used to build an estimator.

Thus, the purpose of Multi-Period returns is not to create an estimator of returns. It is to achieve a more accurate understanding of what actually happened in the past, irrespective of time period. The primary use of Multi-Period returns should be to conduct a comparative analysis of two investments over multiple *corresponding* periods. The single multi-period average is used to compare overall results, while doing a period-by-period comparison between the two investments is a more detailed approach. One such period-by-period comparison could be to say that "Investment A outperformed Investment B in 68 out of 100 periods." But of course one could go much deeper than that.

Table 4: Three years of monthly returns for the Google stock.

Google	
Date	Return
8/1/12	0.082341975
7/2/12	0.09119589
6/1/12	-0.001360052
5/1/12	-0.039662726
4/2/12	-0.056749423
3/1/12	0.037185605
2/1/12	0.065746152
1/3/12	-0.101857873
12/1/11	0.077595555
11/1/11	0.011389714
10/3/11	0.150667909
9/1/11	-0.047914818
8/1/11	-0.103910948
7/1/11	0.192167937
6/1/11	-0.042796114
5/2/11	-0.027715493
4/1/11	-0.072704342
3/1/11	-0.043430062
2/1/11	0.021720301
1/3/11	0.010758119
12/1/10	0.06884886
11/1/10	-0.094492423
10/1/10	0.167196029
9/1/10	0.168370295
8/2/10	-0.071836651
7/1/10	0.089672997
6/1/10	-0.083767477
5/3/10	-0.07622218
4/1/10	-0.073035689
3/1/10	0.076537585
2/1/10	-0.005925199
1/4/10	-0.145230491
12/1/09	0.063430532
11/2/09	0.08744311
10/1/09	0.081214077
9/1/09	0.074035567

4.4 CORRELATION USING MULTI-PERIOD ANALYSIS

Instead of calculating the Multi-Period return for a single security, we can use the Multi-Period analysis algorithm to calculate correlation between two different securities.

The goal of the Multi-Period correlation is twofold, we want to be able to supply some subjective input to the process of calculating the correlation. One would want to fine-tune the similarity measure such that we can try to reflect our own knowledge of the particular application, such as finance where it might be more important what companies have done recently rather than what they did 10 years ago.

The second motivation is a basic assumption that we don't want to put a lot of weight on temporary changes in otherwise correlated data. In other words, if two stocks are perfectly correlated except for a short period, they should still be considered highly correlated despite a temporary change that was relatively large.

First, if we apply Multi-Period Analysis to the monthly returns data for two securities, assuming they both cover the same time-periods, there will be equally many periods.

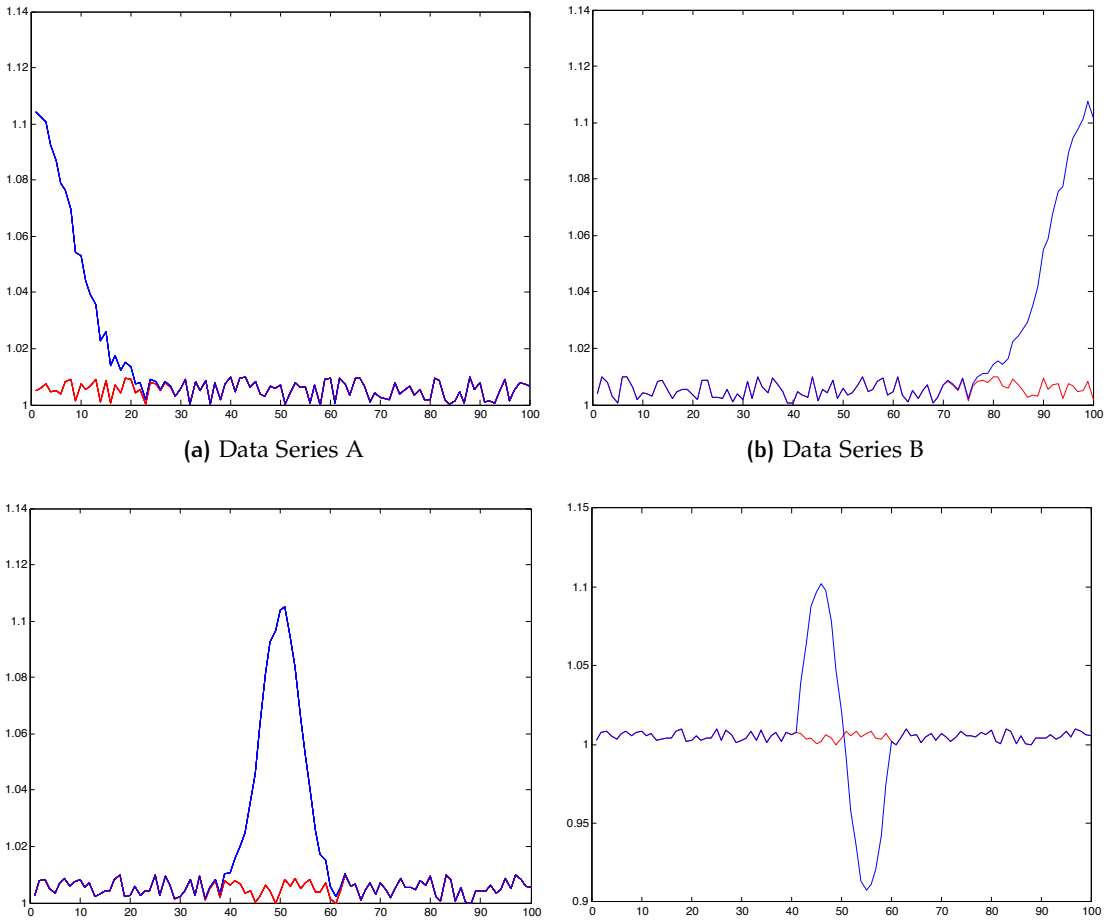
Now instead of calculating the compounded return for each period as we did in the previous section, we calculate the correlation between the two periods representing the same period time and period length for the two securities. In other words, the correlation between the first twelve months, the correlation between the twelve months after the first month and so forth.

If we do this and then take the weighted average according to some weighting function described in Section 4.2, we will get a correlation number between two stocks that describes the similarity between the two stocks and applies a sort of smoothing in the process.

Lets look at some examples.

Example 4.4.1. If we generate a baseline data series using randomly generated numbers from 1 to 1.01 (this interval has been chosen to reflect small changes but the actual range is arbitrary) and then overlay a normal distribution in the beginning, middle or end, we will get pairs of data-series that look like the ones in Figure 15. Additionally, to see the effect of a psychologically affected marketplace (for instance large upswing on rumours and downfall on revelation of the truth), we add an additional overlay of a sine-like form in the middle.

The important thing about these datasets are that they are different for a part of the series and then exactly the same. The purpose of this is to show how multi-period correlation "evens out" the data.



(a) Data Series A

(b) Data Series B

(c) Data Series C

(d) Data Series D

Figure 15: Example data series with piece-wise differing data.

The interesting part is to now compare calculating regular correlation and Multi-Period correlation on the different series to see what differences we can see.

When we do so, we get results as seen in Table 5.

Table 5: Correlation and Multi-Period Correlation for the different Data Series.

Data Series	Correlation	MP Correlation	Increase
Data Series A	0.1772	0.7083	299.72%
Data Series B	0.1324	0.6809	414.27%
Data Series C	0.0559	0.1903	240.43%
Data Series D	-0.1365	-0.0584	-57.22%
Two Random Series	0.0475	0.0924	94.53%

We can see that we achieve significant increases in the correlation when using Multi-Period correlation with the default weighting. We want this in almost all cases because in our application of finance, there could exist differences between two stocks that are large in value but small in time-period. For instance, Data Series A could represent a company that performs very differently in its start-up phase but then falls in line with other companies in its industry. Just because this company was very different 5 years ago does not mean that its

different today, and so it makes sense to put greater weight to the similarities rather than the dissimilarities. Or in other words, we want to see the fact that the pairs in Data Series A are very similar except for this short period.

This could pose a problem when looking at Data Series B though. Where if we again imagine two companies, the performance has become significantly different in recent time. Using the default weighting would take into account that they've been similar for so long they should be similar in the future as well. If one wants to take this into consideration one could use a weighting function that puts greater weight on recent time-periods, then we could maintain the low correlation or even lower it further depending on how much priority we put on the recent data.

An important thing to note is that Multi-Period correlation does not destroy the indication that uncorrelated data is indeed uncorrelated. If we generate two "flat" random data series and test the correlation, the MP correlation should still indicate that they are uncorrelated. The results of a test like this are included in Table 5. As we can see, while for Data Series A, B and C, MP correlation raises the correlation significantly, while for the two random series it raises it a little bit, while the relative change is 95%, the absolute change is only 0.0449, which would not be considered a large absolute change.

The MP correlation algorithm does pose a problem for the kind of sine-like data that we see in Data Series D. The change is actually not indicating that the two series are highly correlated, but indicating that they are less correlated than they actually are.

It might therefore be desirable to look at other correlation functions that aim to achieve the same results as Multi-Period correlation and then test the different similarity functions based on real data. Other measurements of correlation are for instance Kendall τ or Spearman correlation. We will look at these later, and of course our clustering algorithm support using any similarity measure.

4.4.1 A Note On Valid Correlation Matrices

A valid correlation matrix reflects whether the correlations are consistent with each other or not. The requirements for a valid correlation matrix are the following.

1. All the diagonal elements are ones.
2. The matrix is real symmetric. This is always true if the correlations are calculated and not estimated.
3. The matrix is positive semi-definite.

Looking at these requirements, we can see that a valid correlation matrix is not about whether the actual correlation values are correct or not, it is about whether or not they are consistent with each other.

In the world of finance, there is an emphasis on using valid correlation matrices for calculations. For instance, when calculating Value-at-Risk (VaR), a correlation matrix is used when calculating VaR for a portfolio with multiple investments with a parametric approach.

To highlight the importance of valid correlation matrices, consider the following example [Rebonato and Jackel, 1999]. Let's say an analyst has retrieved correlations between securities A, B and C with some tool. The following matrix M represents those correlations.

$$M = \begin{bmatrix} 1 & 0.9 & 0.7 \\ 0.9 & 1 & 0.4 \\ 0.7 & 0.4 & 1 \end{bmatrix} \tag{4.4}$$

We can see that this is a valid correlation matrix as there are ones on the diagonal, it is real symmetric and the eigenvalues are $eig(M) = [0.0303, 0.6160, 2.3536] \geqslant 0$ (positive semi-definite).

So if we were to use this matrix to calculate the VaR then we would get some result. Now the analyst wants to simulate what happens if the correlation between B and C dropped from 0.4 to 0.2. We could simply input these values in the matrix and get

$$M_{sim} = \begin{bmatrix} 1 & 0.9 & 0.7 \\ 0.9 & 1 & 0.2 \\ 0.7 & 0.2 & 1 \end{bmatrix}. \tag{4.5}$$

However, this is no longer a valid correlation matrix. It's eigenvalues are $[-0.0487, 0.8065, 2.2421]$, the one negative eigenvalue tells us we no longer have a positive semi-definite matrix.

This comes from the fact that we can not simply change the correlation between B and C, as this means the underlying data for either B or C or both would have changed. This change would result in a change in correlation between B and A or C and A as well, so the other values need to be adjusted accordingly.

Calculating the VaR with the modified matrix would yield incorrect results. There are several methods to adjust M_{sim} such that we get a valid correlation matrix in some optimal sense [Higham, 2001]. It is slightly out of scope to present those methods in this thesis and we shall only focus on proving that our correlation matrix is already valid.

4.4.2 Multi-Period Correlation and Validity

It is not as clearcut of a case to present why an invalid correlation matrix would yield incorrect results in the case of clustering. However, one can imagine that having inconsistent correlations would yield inconsistent clustering results. In other words, if we were to use M_{sim} from the previous Subsection, then we would say that B is more similar to C than it could theoretically be.

To re-assure ourselves of the fact that Multi-Period correlation matrices are indeed valid, we only need to consider the requirements for validity and to think a bit about what Multi-Period correlation calculations actually do.

When creating the Multi-Period correlation matrix, for each correlation value, we calculate multiple correlations and take a weighted average of those values. This is equivalent to saying we calculate multiple correlation matrices and take a weighted average of all of them.

Now going through the requirements for a valid correlation matrix, the first is clearly satisfied if the weights used add up to 1, which they should. So we still have ones on the diagonal. The weighted average of real symmetric matrices is still real symmetric, this is quite clear to see as well. The only question that is a little bit tougher to answer straight off the top of your head is the third requirement that the final correlation matrix stays positive semi-definite. However, it can be very easily shown. We remind the reader of the defining attribute of a positive semi-definite matrix; $x^T Ax \geqslant 0$ for all $x \in \mathbb{R}^n$.

We have the weighted average $M = w_1 A_1 + w_2 A_2 + ... + w_k A_k$ with weights $w_1, ..., w_k$ where $\sum_j w_j = 1$.

$$x^T M x = x^T (w_1 A_1 + ... + w_k A_k) x = w_1 x^T A_1 x + ... + w_k x^T A_k x \geqslant 0 \tag{4.6}$$

Since $A_1, ..., A_k$ are all positive semi-definite (they are valid correlation matrices for different periods) the resulting matrix M must also be positive semi-definite.

We have shown that Multi-Period correlation matrices are still valid correlation matrices. Which strengthens the idea of using them as the basis for our clustering.

4.5 ALTERNATIVE CORRELATION-BASED SIMILARITY MEASUREMENTS

As we discussed in the previous section about Multi-Period correlation, one of the primary goals was to try to smooth the data and put less weight on outliers. There are other correlation measurements that might capture this effect as well, although they might have problems of their own. Among other things they don't provide a way to give a subjective input on the calculation. Still, they might provide a good way to achieve our goal in certain situations so it makes sense to include them here and include them in our tests on the real data.

Both Kendall's τ and Spearman correlation, detailed below, are so-called rank correlation coefficients. That means that instead of comparing the actual value of the data, they compare the rank. The rank is the order in which a value would appear if they were sorted. So for instance, the rank of the values in the vector $[-1, 2, 0.5, 10]$ would be $[1, 3, 2, 4]$.

Because we are working with ranks instead of values, we will achieve a more robust correlation measurement as an outliers' value will not affect the calculation, only it's position in the series (rank) will.

In our application of finance, it is sensible to say that we don't care about the amount two stocks are rising with, if they are both always rising (but with different amounts) then they should be considered correlated. In that sense, using rank instead of values makes sense.

4.5.1 Kendall's τ rank correlation coefficient

In calculating Kendall's correlation, one first pairs up all observations $(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$, there are $\frac{1}{2} n (n-1)$ such pairs. The variables x_i, y_i are the in this case the ranks of the actual values X_i, Y_i.

Two of these pairs are considered *concordant* if they move in the same direction, if both $x_i > x_j$ and $y_i > y_j$ or both $x_i < x_j$ and $y_i < y_j$ hold true. Similarly, they are considered *discordant* if they are not moving in the same direction, if the previous conditions are false.

Kendall's τ is then defined as following.

$$\tau = \frac{(\text{number of concordant pairs}) - (\text{number of discordant pairs})}{\frac{n}{2}(n-1)} \tag{4.7}$$

There is a third case in which two values might be exactly the same, the rank would then be considered tied, and there are three variations on Kendall's τ where two make adjustments to account for ties and one does not. It is out of scope to explain the adjustment process, but in this thesis MATLABs implementation of Kendall τ will be used through the function `corr`, the algorithm used there will account for ties.

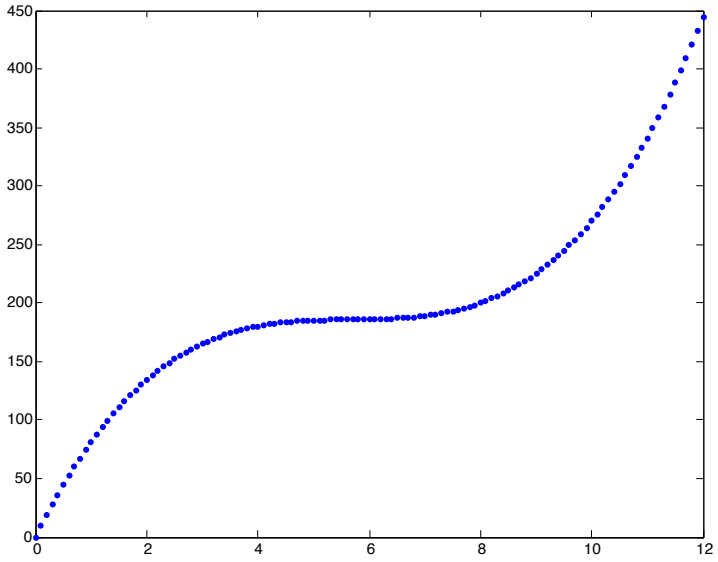


Figure 16: Pearson (normal) correlation = 0.91, Kendall's $\tau = 1$

In Figure 16 we see a plot of two data series, on the X-axis are values increasing linearly from 0 to 13. On the Y-axis we have values increasing according to a cubic function. The important thing to note is that both series have continuously increasing values. So in our financial example of two stocks, they are both constantly going up in value.

Normal Pearson correlation between these two series would say that they are not perfectly correlated, but has a correlation value of 0.91. However, Kendall's τ yields a perfect correlation of 1.

To be able to compare to Multi-Period correlation and normal correlation, lets produce the same comparison table as we saw earlier, but for Kendall's τ correlation coefficient instead.

Table 6: Correlation and Kendallr Correlation for the different Data Series.

Data Series	Correlation	Kendall Correlation	Increase
Data Series A	0.1772	0.6470	265.12%
Data Series B	0.1324	0.6477	389.20%
Data Series C	0.0559	0.5913	957.78%
Data Series D	-0.1365	0.5630	312.45%
Two Random Series	0.0475	0.0435	-8.42%

It is interesting to see that the Kendall τ produces more consistent results than MP correlation, and that the τ value roughly corresponds to the percentage of values that are different in the two data-series (which of course is to be expected when only considering the rank). It also produces a smaller change for the truly uncorrelated samples than MP correlation. For Data Series D it is also worth nothing that it actually turns a normally negatively correlated series into a positively correlated series.

4.5.2 Spearman's rank correlation coefficient

Spearman's correlation is just like Kendall's a measurement based on the rank of the values in the input data series. Often when talking about Spearman's correlation one talks about monotonically related data.

If one can perfectly describe one data series through a monotonous function applied to the other, then they are perfectly Spearman rank correlated (correlation 1). By contrast, the same thing can be said about normal Pearson correlation for data series related by a *linear* function.

In this way, Spearman is a more general approach than Pearson. Like we said before, in finance one could argue that we are more interested in monotonically related data, and don't just want to test for linear relation. If one is going up as long as the other one is, then that is a strong indication of similarity even if they are not increasing linearly with each other.

Spearman's correlation does not generally deal with ties, but one can find algorithms for Spearman's correlation that are tie-adjusting. If one can guarantee that there are no ties, the definition is the following.

$$\rho = 1 - \frac{6 \sum d_i^2}{n\,(n^2 - 1)} \tag{4.8}$$

In this equation $d_i = x_i - y_i$ where x_i and y_i are the ranks of the corresponding values X_i and Y_i.

If we look at the same table as for the other correlation values, we see that the results are very similar to that of Kendall's τ for the same data sets.

Table 7: Correlation and Spearman Correlation for the different Data Series.

Data Series	Correlation	Spearman Correlation	Increase
Data Series A	0.1772	0.6973	293.51%
Data Series B	0.1324	0.6477	389.20%
Data Series C	0.0559	0.6219	1012.52%
Data Series D	-0.1365	0.5203	281.17%
Two Random Series	0.0475	0.0525	10.53%

5 MEASURING DISSIMILARITY

As mentioned in Section 1.2 there are currently many ways to group securities together and try to choose investments from the different groups in order to try to increase the diversification of our portfolio.

However, none of these current methods provides a good way to take multiple sources of information into account and easily combine them. Neither is there a fully automatic way to group securities into a desired number of groups.

What we aim to provide in this thesis is a new way to group securities, one of the cornerstones of this method is the ability to calculate a number representing the dissimilarity between two securities.

There are multiple ways to get such numbers and in this chapter we will detail the ones used by Bdellium to achieve a high quality grouping of any securities for which there is data available.

5.1 CORRELATION AND MULTI-PERIOD CORRELATION

We can convert the correlation to a dissimilarity measure by Equation 5.1 [Kaufman and Rousseeuw, 1990], where d is the dissimilarity and c is the correlation number.

$$d = (1 - c)/2 \tag{5.1}$$

Since correlation is in the range $[-1, 1]$ where -1 means highly dissimilar for securities (if one goes down it is highly favorable if we choose a security with correlation -1 as we then know it will go up) and should result in a dissimilarity of 1. Correlation of 1 means highly similar, so the dissimilarity should be 0.

We can now pair-wise compare every security with every other security in order to produce a dissimilarity matrix ready for input to our clustering algorithm in Chapter 2.

Example 5.1.1. Comparing Google, Yahoo, Boeing and General Electric.

Gathering data equivalent to that for Google in Table 4 for three other companies, namely Yahoo, Boeing and General Electric, allows us to create a dissimilarity matrix using Multi-Period correlation as the way to measure dissimilarity.

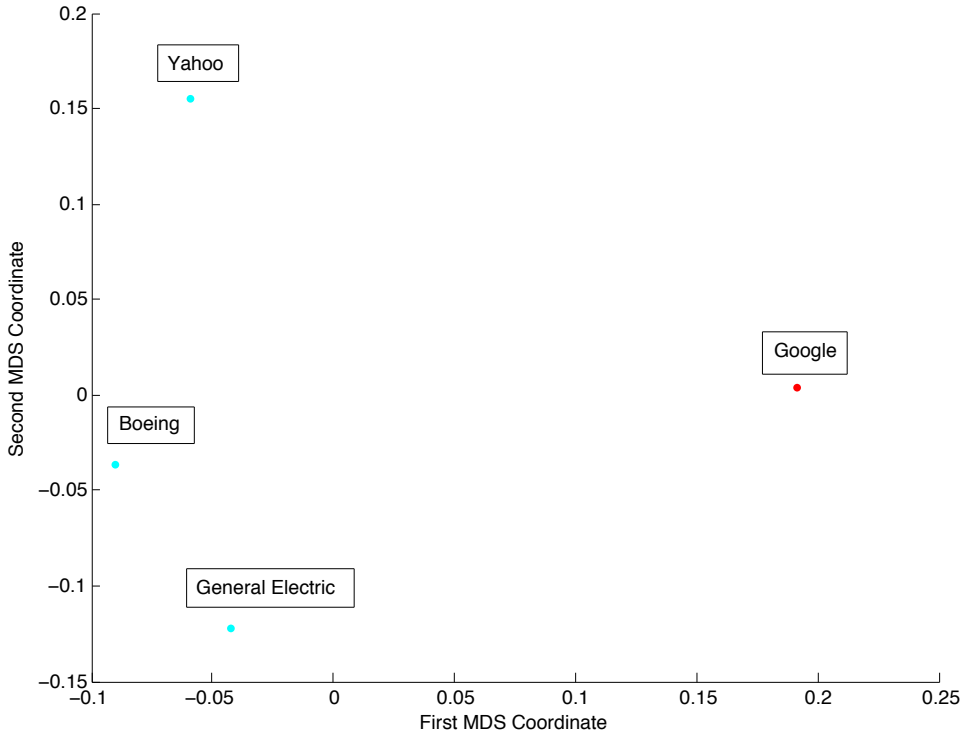
Table 8: Dissimilarity matrix for four stocks.

	Google	Yahoo	Boeing	General Electric
Google	0.0000	0.2931	0.2871	0.2698
Yahoo	0.2931	0.0000	0.2072	0.2771
Boeing	0.2871	0.2072	0.0000	0.1391
General Electric	0.2698	0.2771	0.1391	0.0000

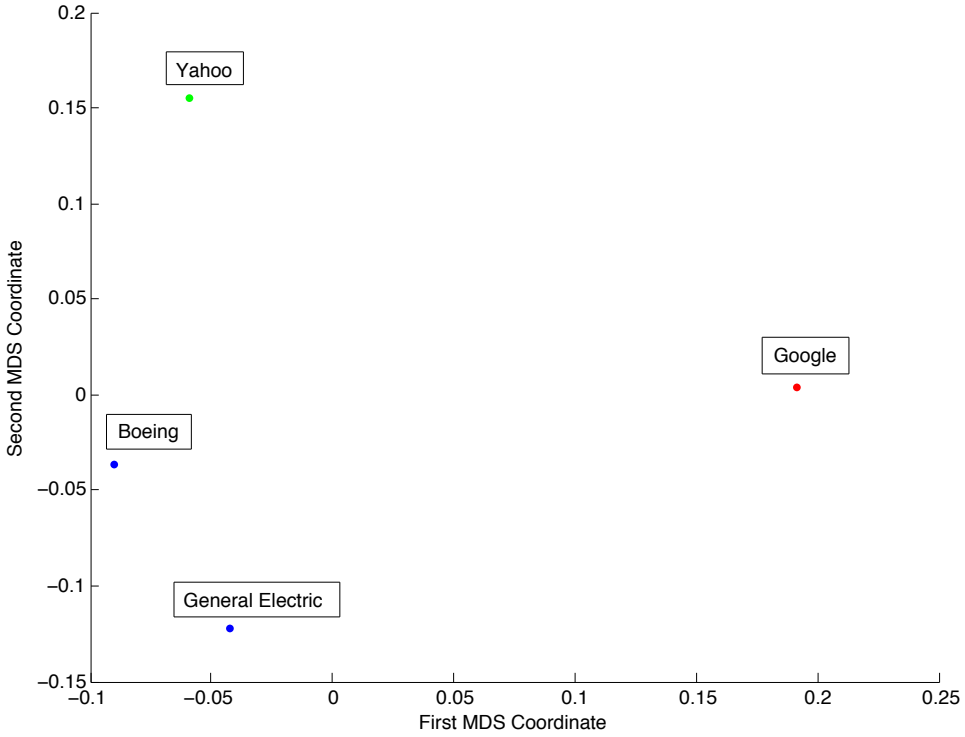
Using the dissimilarity matrix in Table 8 as input to our clustering algorithm, for clustering into 2 groups we get the clustering result $[1, 2, 2, 2]$ and for clustering into 3 groups we get the result $[1, 2, 3, 3]$. If we now illustrate these results using Multidimensional Scaling as described in Section 2.2 we get results as shown in Figure 17a and Figure 17b.

We would expect that Boeing and General Electric would be quite similar, being in the same Industry and of the same era. Likewise, one might expect that Yahoo and Google would be quite similar for the same arguments. However, looking at the results of our clustering it is quite clear that Google and Yahoo is not very similar at all when looking simply at the returns.

When clustering into 2 groups, our algorithm even places Yahoo into the same group as Boeing and GE instead of clustering it with Google. However, it is right on the verge and should really be in a group on it's own, as we can see when clustering with three groups.



(a) Four stocks in two groups.



(b) Four stocks in three groups.

Figure 17: Plotting the clusters resulting from analyzing four different stocks based on correlation.

Simply looking at returns and no other dissimilarity measurement might not give us the best results. Yahoo and Google are in the same industry and are clearly exposed to the same

type of risks (for example should a second IT-bubble burst). Therefore, we would want to impose some further similarity between Yahoo and Google because of this shared risk.

Fortunately, our method of grouping is indifferent to how the dissimilarity matrix is constructed and it is a simple matter to construct another dissimilarity matrix based on Industry and then combining the two, as we shall se in Sections 5.2 and 5.3.

5.2 OTHER MEASUREMENTS OF RISK AND DISSIMILARITY

As mentioned previously, there is a whole host of different risk factors we could take into consideration when clustering securities. The most obvious ones will be listed below, but there is essentially no limit as to what factors could be considered in the clustering as long as there is some comparative nature of the measurement.

5.2.1 Industry

One of the major risk factors identified when looking at securities is industry. It has the most intuitive clear effect on diversification and risk. If we were to invest only in technology companies working with search engines, we would not really be diversifying no matter how many of these companies we invest in. The logic here being that if there comes some major change to the search engine market that affects all companies negatively (like some new legislation), we will be in the negative for all our investments.

Diversifying on correlation between companies will help as securities within the same industry are generally highly correlated, but not always. It is not good enough to look only at the correlations as they will not catch all the nuances and will not account for extreme events such as a whole industry going down.

5.2.2 Country

Major national events can have positive or negative effects on the entire securities market of that country. It is therefore important to diversify across countries, such that if one countries market plunges, one can try to offset that loss by investing in a different countries market.

Diversification across countries is not as common as diversifying across industry, and banks are often recommended to invest with a bias toward their own country or continent [Marrison, 2002], the logic being that they have expertise in that market and can thus cover risks better with the extra knowledge instead of blindly going for more diversification.

In our clustering methods, country can easily be incorporated in the similarity measure between securities, however it should probably not be given as heavy weight as for instance Industry.

5.2.3 A worked example

Morningstar provides industry categorization data for all securities in their database, grouping all securities into 4 super sectors, 13 sectors and 148 industries [Morningstar, 2011]. They also provide a range of other information such as country, whether they are US securities,

Asian, European or other continent. Theres also the classical grouping into the so-called Morningstar Style-box.

Some of these measurements might provide an interesting grouping on its own, however most should probably be used in conjunction with another measurement such as the correlation. How to combine dissimilarity matrices is described in Section 5.3. Let's construct a dissimilarity matrix for our four stocks from the previous example.

Example 5.2.1. Dissimilarity matrix using Industry as dissimilarity measurement.

The four stocks are divided into two different industries as is shown in Table 9.

Table 9: Industry classification for our four stocks.

Company	Industry
Google	Technology
Yahoo	Technology
Boeing	Industrial
General Electric	Industrial

Now to describe the dissimilarity, we want a dissimilarity of 1 if the two securities are *not* in the same industry and 0 if they *are* in the same industry. For example, the dissimilarity between Google and Yahoo is 0 and between Google and Boeing it is 1.

This leads to a dissimilarity matrix which is very easy to calculate.

Table 10: Dissimilarity matrix for four stocks based on industry.

	Google	Yahoo	Boeing	General Electric
Google	0	0	1	1
Yahoo	0	0	1	1
Boeing	1	1	0	0
General Electric	1	1	0	0

Of course, grouping only based on this dissimilarity matrix is tantamount to simply assigning each industry a group number. However, combining it with the dissimilarity matrix from Example 5.1.1 yields useful results as we will see in Section 5.3.

5.3 CONSTRUCTING THE DISSIMILARITY MATRIX

We've seen how we can produce dissimilarity matrices for lots of different measurements. How to construct the final matrix to be used for input into the clustering algorithm is an important step and one that is different from constructing the individual matrices for each measurement.

The difference is that we have to input some amount of subjectivity. The way to combine the different matrices is simply by calculating a weighted average [Kaufman and Rousseeuw, 1990].

When we're calculating the Multi-Period Analysis correlation we are also inputting some amount of subjectivity through the weighted average, it is even clearer when combining the different matrices.

Using an evenly distributed weighting function (same weight for every matrix) one might think we are being "fair" and assigning equal importance to every category. Though this

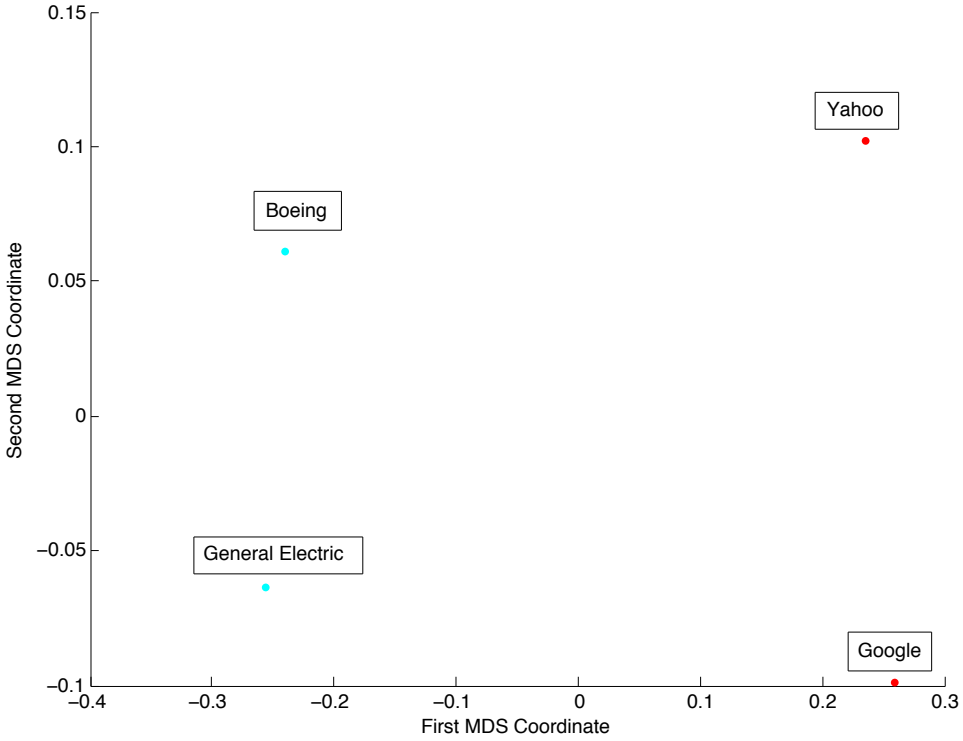
might be a skewed view in some cases. Assigning equal importance to correlation and industry is probably not a good idea from the start as they are probably not equally important in determining risk and optimizing diversification.

Ultimately, the weighting function is a subjective factor and needs to be input by the user of the system.

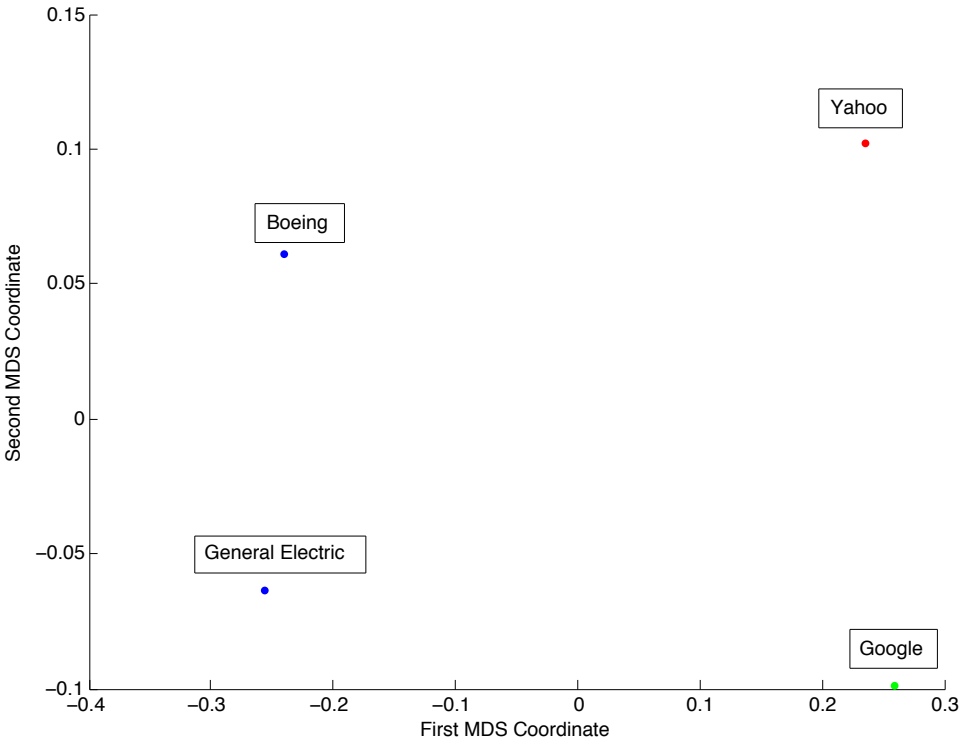
To continue on Example 5.2.1 and combine the correlation matrix and the industry matrix for our four stocks, we can arbitrarily choose a weighting of $w_1 = 1/3$ for industry and $w_2 = 2/3$ for correlation. Then the combined dissimilarity matrix D becomes

$$\begin{aligned} D &= \sum_{i=1}^{4}\sum_{j=1}^{4} w_1 d^i_{i,j} + w_2 d^c_{i,j} \\ &= \begin{bmatrix} 0.0000 & 0.1954 & 0.4715 & 0.5181 \\ 0.1954 & 0.0000 & 0.5247 & 0.5132 \\ 0.4715 & 0.5247 & 0.0000 & 0.0927 \\ 0.5181 & 0.5132 & 0.0927 & 0.0000 \end{bmatrix}. \end{aligned} \tag{5.2}$$

Using this as our basis for clustering we get the clustering $[1, 1, 2, 2]$ for two groups and $[1, 2, 3, 3]$ for three groups. Again we see that the grouping with three groups is exactly the same. However, importantly, the clustering for two groups has changed to reflect the fact that Yahoo and Google are in the same industry. The results can be displayed using MDS as seen in Figure 18.



(a) Four stocks in two groups.



(b) Four stocks in three groups.

Figure 18: Plotting the clusters resulting from analyzing four different stocks based on correlation and industry.

6 SOFTWARE DEVELOPMENT

As a majority of the time spent on this thesis project will have been time spent programming, this chapter will not be proportional to the time spent on software development. It is however important to make note of the contributions of the programming language to this project.

Many of the algorithms discussed in this thesis such as K-medoids or FAMES are not new discoveries made by the author. They are rather well-established techniques of clustering. What is new in regards to those particular algorithms is the implementation.

The implementation language is Haskell, which shall be covered further in the coming section, and Haskell has a public repository of open-source packages. This repository is known as Hackage. The Haskell community is a very open-source driven one and there are essentially three camps of developers. Those who work for large institutions such as banks, those that work for open-source projects, and academics. The open-source projects will invariably be available in Hackage, the code from the institutional camp will never get to see the light of day as it is highly protected by the institution. The academics rarely, if ever, focus on implementation and rarely release their code.

Despite various research efforts and discussions with the Haskell community through different channels, it seems as though there is no available package for clustering. K-medoids seems to never have been implemented in the language (though K-means pops up here and there as a programming exercise). It is really a shame though, because as we shall see, Haskell is in the authors opinion very suitable for this kind of work.

6.1 WHAT HASKELL IS

6.1.1 Purely functional

Haskell is a purely functional language. Lets try to break down that statement a little bit. First, a functional language is a language where functions are considered *first-class*. In other words, that functions are the same as any other possible value in the language. We can create a function inside a function and return that. We can receive a function as an argument, apply it to something and then send back the result or send back a new function. In Haskell, all expressions are a function. If we write "`a = 3`", then we have just declared a function with the name `a` that returns the value `3`. This function `a` takes no parameters, so when we require the value of `a` anywhere in our code the `a` function is evaluated and the value `3` is returned.

The "pure" part is what really makes Haskell shine. Haskell's functions are pure by default. The opposite of a pure language is a language that produces side-effects. Often languages such as Java, Python and C are programmed such that they mutate global state in some way. For instance you may create global variables that are changed by several functions in different places in your code. In a pure language, this is not allowed. You are not allowed to change any global state at all.

A pure function is a function where there are no side-effects. Side-effects are any effects that affect anything outside the function.

For instance, this C function is impure.

Listing 1: Example impure function

```
int a = 0;

int addOne(int b) {
        a += 1;
        return b+=1;
}
```

Even though given the same input `b` and the same output `b+1` every time we run the function, we have a side-effect of increasing `a` by one.

The "same" code in Haskell is pure. This is a contrived example, though it illustrates the point of purity as well as gives an introduction to the next subsection.

Listing 2: Example pure function

```
a :: Int
a = 0

addOne :: Int -> Int
addOne b = let a = a + 1 in b + 1
```

We define the function `addOne` in a way that calculates a "new" `a` with the old `a`, then we return `b+1`. This is pure because in Haskell, variable assignment is not possible. What we are doing in the `let` statement is declaring a new function `a` that returns the value of the function `a` plus one. We can see that this causes an infinite loop if it were to be evaluated.

This function is not only pure, but it still compiles and runs. The reason that we do not get stuck in this infinite loop is because Haskell is lazily evaluated. Since we always just return `b + 1`, the Haskell compiler realizes that we never need to calculate `a` and, at run time, it never does.

In short, a purely functional programming language is much closer to mathematics than it is to mechanical processing. This probably stems from the fact that Haskell was originally developed by academics for academics.

6.1.2 Lazy

One of the major features for Haskell is its laziness. The main purpose of introducing pure functions is because it de-couples time and execution. In an imperative language like C or Java, *when* we execute a function can be as relevant to the results as what arguments it has. For instance in Listing 1 the effect of calling `addOne` can only be determined if we already know how many times it has been executed. In Haskell, a pure function will always return the same value given the same input arguments. Therefore, we can execute a Haskell expression at any time we like and still have the same results. This is the foundation of laziness.

What now happens under the hood of a running Haskell program is that all computations are deferred until the values of those computations are needed. For instance, if we created

a program that performed a massive computation that would take hours to finish and then in Haskell call that function but *without* doing anything with the results, the program would run instantly. If we printed the results, it would execute the full computation and take the necessary time.

In reality, this provides us with a lot of high-level programming abilities and possible abstractions, but it can also provide for some difficult optimization questions as it is hard to know exactly when a computation will be performed.

An example of the benefits are infinite lists. Lets say we wanted to attach a numbered index to every item of a list. In an imperative language we might find out the length of the list and then iterate through them in a for-loop. In Haskell this is reduced to the zip function, which takes two lists and pairs up the items in those lists until one of the lists are spent. So to accomplish our task pairing up a number with every item of a list, we simply zip the list with the infinite list of numbers from 1 and upwards and no matter how long the input list, we will have the desired result.

Listing 3: Example function to pair every item with an index

```
pairWithIndex :: [a] -> [(a, Int)]
pairWithIndex unknownList = zip unknownList [1..]
```

6.1.3 Strongly and statically typed

Haskell is a strongly and statically typed language. That means that types are required, though in Haskell's case they may be inferred by the compiler. It also means that the types are checked at compile time, so that it is guaranteed to not have any type errors at run time.

The strength of Haskell lies in its type system. The language has implicit type declaration, so while types are required for all expressions, the compiler can conclude the type of expressions and automatically assign any expression a type. It is usually common practice to always define types for top-level function declarations, but not for every expression inside a function.

Generally, the way you would write a function is to define the type of the function first. If it is too complex perhaps you need to introduce a new data-type or move the computation into what is called a Monad (a description of which is outside the scope of this chapter). When the function type is defined, you try to write an implementation of the function that conforms to the constraints of the already defined type.

A simple sample here could be in order. Let's say we want a data type that describes two different kinds of glasses, Empty and Full. We would declare a Glass data type like this.

Listing 4: Example data declaration

```
data Glass = Empty | Full
```

Then we can write a function that takes a Glass as a parameter and returns an Action, which is a new data type we can declare in the same manner.

Listing 5: Example data declaration

```
data Glass  = Empty | Full
data Action = Increase Double | Decrease Double

adjustLevel :: Glass -> Action
adjustLevel Empty = Increase 2.8
adjustLevel Full  = Decrease 3.2
```

We can see that data declarations are not just simple enumerations of words, they can be much more complex. In this example, we added a Double parameter to the Increase or Decrease datatype, essentially storing dynamic information about how much to Increase or Decrease. Then in the function, first of all, we see that the type declaration is on top, this tells us what this function does. It takes a Glass and returns an Action. Type declarations of functions are often seen as part of the documentation of a function, since a lot of information can be gained from just reading the type.

Another thing we see in example Listing 5 is the very important pattern matching.

Haskell has what is called Pattern Matching. Meaning that when executing, the computer matches the argument to what is on the left-hand-side of the equal sign to determine which righ-hand-side to execute. Haskell's pattern matching system is obviously heavily tied together with the type system. What makes Haskell a pleasure to program is the combination of the powerful type system and the ability to pattern match those types.

In the case of the Glass data type, there are only two possible matches, Empty and Full, and thus we can enumerate all possible matches.

The pattern matching system is (of course) much more powerful than that. For example we can match Integers.

Listing 6: Example pattern matching

```
matchInteger :: Integer -> String
matchInteger 1 = "We got a 1"
matchInteger 2 = "We got a 2"
matchInteger x = "We didn't get 1 or 2, we got " ++ show x
```

In this case, we have a "catch-all" case where we match any, not already matched, arguments and assign the value to `x`. We can then print that value. This example shows that patterns are matched from top to bottom. If we are not matching every possible case we get a compilation warning, if we ignore that warning and execute the statement with an argument that is not matched, the program will crash.

6.2 APPLICATION MODELLING

Haskell has a very simple yet powerful module system. It is important to note that Functional programming in the case of Haskell means that it is not Object oriented. There are no instances of objects or instances of classes or even instances of types. There are only values.

In the language there are things such as type classes but they are very distinct from object oriented classes.

Therefore, application modelling does not follow the same principals as Object Oriented programming. We can set up a data type to store information for a User like this.

Listing 7: Example record data type

```
data User = User { username :: String,
                   fullName :: String,
                   phoneNumber: String,
                   accountBalance :: Money }
```

But when we create a User, it is not an object with methods that belongs to a particular class. It is simply a value with type User.

Since we do not have to define functions and types together like in traditional object oriented programming, we are free to organize our code as we want to.

Typically, this means that you have a module that contains all of your types so that you can easily browse them and see what a particular function should use as their type signature. Then we organize the actual code in terms of functionality instead of what kind of types they operate on.

6.2.1 Modelling BdTools

The application for this thesis project has been named BdTools, and I will describe here in short how it is modelled.

As mentioned, the typical way to organize code around the kind of function it performs.

In BdTools there are the following modules.

Listing 8: Module structure for BdTools

```
-- Parent module that contains all the modules
BdTools
  -- Contains all modules related to purely statistical measurements
  BdTools.Statistics
    -- Contains code to calculate different similarity measurements between
       securities
    BdTools.Statistics.Similarity
    -- Contains all code that calculates different Multi-period measurements
    BdTools.Statistics.Multiperiod
  -- Contains all modules that relate to different grouping algorithms
  BdTools.Grouping
    -- Contains all code for the different types used in the clustering
       algorithms
    BdTools.Grouping.Types
    -- Contains various utility functions that are general to all Grouping code
    BdTools.Grouping.Utils
    -- Contains all code for the PAM algorithm
    BdTools.Grouping.PAM
    -- Contains all code for the FAMES algorithm
    BdTools.Grouping.FAMES
    -- Contains all code for the group adjustment algorithm
    BdTools.Grouping.Adjustment
```

As we can see, it is very simple to write clear hierarchical code organization in Haskell. A new user to this system would only have to read through the relevant data types and they

can immediately dive into specific functions. Since all functions are pure, they never have to worry that one part of the code will affect a different part. As long as you don't change the type signature of a function, you are ensured by the compiler that your code will not break. Of course you may change a function to produce incorrect values, but having the type-level security allows for a very clear modelling procedure.

6.3 TESTING

A relatively recent trend in application development is what is called Test Driven Development, where you first define what you want by writing tests for the code. Essentially you write a specification of what input a piece of code should have and what output it should give. If you are doing web development you might define the ability to click a button named a certain thing and then measure the outcome of that click in some way. After having written the tests you start writing the implementation and you keep writing it until you pass the tests.

In Haskell, this is traditionally not exercised. There are two popular reasons cited for this, one is that the type system is itself a definition of input and output, and since we are in a pure setting; those are the only possible parameters to control. The second reason is that Haskell is typically tested with something called QuickCheck. A very powerful testing system that has been ported to many other languages.

QuickCheck is a system that is much more like testing mathematical properties of functions. You do generally not define an input value and test it against an output value (though that is possible in Haskell as well) but rather you write functions that take *any* input of the correct type and then test a property of the output given that input.

In Subsection 4.4.1 about Valid correlation matrices we discussed the properties of such a matrix. Let's say we wanted some empirical evidence that our Multi-Period Correlation matrices were indeed valid correlation matrices. We could write a test like the one in Listing 9.

Listing 9: Test that Multi-Period correlation matrices are valid

```
validCorrelationMatrix :: [[Double]] -> Bool
validCorrelationMatrix listReturns = diagIs1 && eigGTE0
  where
    diagIs1 = all (== 1) (diag mat)
    eigGTE0 = all (> (-1e-14)) (eig mat)
    mat = similarityMatrix returns
    returns = V.fromList (map U.fromList listReturns)

listOfListsLongerThan :: Int -> Int -> Gen [[Double]]
listOfListsLongerThan c l = suchThat
                              arbitrary
                              (\lol ->
                                length lol >= c &&
                                minimum (map length lol) >= l)

test_ValidCorrelMatrix :: IO ()
test_ValidCorrelMatrix = quickCheckWith
                           stdArgs { maxSuccess = 5000 } $
```

```
                    forAll (listOfListsLongerThan 2 12)
                      generatingValidCorrelationMatrix
```

This contains a function that generates lists of lists of Double values that are longer than a given set of arguments. This represents generating the returns for c different companies where every company has at least l months of returns in their history.

In the test `test_ValidCorrelMatrix`, we generate such lists with at least 2 companies and at least 12 months of returns. The argument maxSuccess makes sure that we generate 5000 different combinations of companies and months of returns. Then we have the `validCorrelationMatrix` function that tests the properties mentioned in Subsection 4.4.1. Namely, that all diagonal elements are 1 and all eigenvalues are greater than or equal to 0. In the test we have defined that as greater than -10^{-14} because of rounding errors produced by a 32-bit machine.

Running this test will generate 5000 random Multi-Period matrices and make sure that they follow the rules for valid correlation matrices. All 5000 tests pass and we have a strong indicator that Multi-Period matrices in fact are valid.

If we were to have a finitely enumerable data-type as input to our test, we could in fact construct proofs in this manner.

6.4 WHY HASKELL IS SUITABLE FOR FINANCE

Arguably, the two most desirable traits in financial tools are robustness and correctness. So how does Haskell help towards these two goals?

Robustness means that the application does not crash while executing. In Haskell there are several measures to control the possibility of a run-time crash. There are very few functions that can cause a crash (also known as a fatal error). The most important thing about Haskell is that fatal errors *can* be avoided. It is possible to know exactly which built-in functions can cause fatal errors and to write your own functions such that no fatal errors are produced. Of course, if you want to be sure that no crashes affect your service, you can also easily employ an Erlang-type mentality of accepting failure as a possibility and program to recover from it.

The type system is obviously fundamentally important to the robustness of an application. A common phrase in the Haskell community is "If it compiles, it works." Meaning that once a program compiles, it runs and will do what you want. I would want to modify this statement to "If it compiles, it is robust." The reason being that compilation does not provide any measure of correctness of the results.

That brings us to our next issue, which is correctness. This is a very hard thing to prove or statically test in an application. However, the combination of the type system, the purity and testing important pieces with QuickCheck makes it easier than in most other languages to end up with a correct program.

Example 6.4.1. An example of using Types to increase probability of a correct program.

Let's say we have two different functions to calculate how much money a person defers. They both take the same arguments, three floating point values and returns a floating point value. The arguments do not, however, represent the same things. What it does internally is to calculate the percentage of salary deferred and compares that to a given limit. If it is over the limit, it returns the dollar amount that represents the limit, otherwise return the actual dollar amount deferred.

The first function takes a dollar value for the amount the deferred, the second takes a percentage value of amount deferred. An example implementation as follows.

Listing 10: Example not using types

```
dollarDeferral :: Double -> Double -> Double -> Double
dollarDeferral salary deferral limit
  | deferral / salary > limit = limit * salary
  | otherwise = deferral

percentDeferral :: Double -> Double -> Double -> Double
percentDeferral salary deferral limit
  | deferral > limit = limit * salary
  | otherwise = deferral * salary
```

The change in logic here is extremely subtle. One can very easily mistake the two functions and the type system would in this case provide absolutely no help in using the right function. With a bit of Haskell knowledge, we can however remedy this situation.

Listing 11: Exampl using types

```
data Deferral = Money Double | Percent Double

deferral :: Double -> Deferral -> Double -> Double
deferral salary (Money def) limit
  | def / salary > limit = limit * salary
  | otherwise = def
deferral salary (Percent def) limit
  | def > limit = limit * salary
  | otherwise = def * salary
```

By declaring a Deferral data type that is either Money or Percent, we can pattern match on that argument and be sure to always execute the correct equations. Moreover, it is a trivial matter (actually it is easier than the alternative) to now concatenate these two functions into one. Using the type system, we have improved probability of correctness as well as simplified the situation for ourselves as we only need to remember one function name.

This can then be carried through the entire application, the compiler will know whether a function is getting a percentage value or a money value as argument. So if a mistake is made, it is caught at compile time instead of at run time.

As one becomes more proficient with Haskell, it becomes very natural to try to describe things as clearly as possible with the type system. Refactorizations such as the one in Example 6.4.1 becomes second nature, and in general, from the authors experience and from the opinions of those who choose Haskell above other languages, it is *easier* to write correct code in Haskell than incorrect code. While in other languages there is no tangible difference between the two, I.E. there are generally no syntactic or language motivators to writing correct code.

7 RESULTS

Now that we have the theory in place and the code in place to test the theory; it is time to run the application on a sample data-set and see what results we get out of it.

The sample data set chosen is a set of 1000 US equity funds, for each set we have the ticker, the Morningstar category label and 3 years of monthly returns (in USD).

In this chapter we will first see a little bit of how the application is used, an example of a grouping illustrated with `MATLAB`'s `mdscale` function, and finally we will look at some more quantitative quality studies for the different measures of similarity (Multi-Period, Pearson, Kendall and Spearman).

Finally we run the quality measurements for the grouping provided by Morningstar's Style-box categorization so that we can compare that to clustering the same securities using PAM and some similarity measurement.

7.1 APPLICATION USAGE

The application in its current state is entirely a command-line application, in other words, designed to be run by other systems or to be run in something like Terminal on Mac OS X. The executable name is `BdTools`, so to run any command you would on the terminal type `BdTools <command> <parameters>`. All the valid commands can be seen in Listing 13. At the end of this section.

The application supports reading raw securities data from a CSV file and generating a correlation matrix from the raw data, it can then use the matrix to run the grouping or you can choose to print it to a different file. On the command-line, to read CSV data and print the matrix you would run `BdTools printMatrixFromCSV path/to/csv_file.csv path/to/output_matrix.txt`. For example, below is a sample table with 10 securities along with 3 months of return data.

Table 11: Sample CSV data.

Ticker	Style Box Category	Return 2009-10	Return 2009-11	Return 2009-12
AAAGX	Large Growth	-2.06897	5.39906	3.58387
AADDX	Large Blend	-1.20482	5.22648	3.09795
AADEX	Large Value	-1.50875	5.82108	1.59872
AAGOX	Large Growth	-1.6479	6.14354	2.0784
AALGX	Large Blend	-2.27861	4.87546	2.8473
AALVX	Small Blend	-5.53691	2.39787	7.8912
AAMOX	Mid Growth	-5.35456	3.66972	6.93215
AASCX	Mid Blend	-5.13937	3.76492	5.9292
AASMX	Small Blend	-6.81199	4.2885	6.72897
AASOX	Small Growth	-6.12161	3.80744	7.84148

Two other commands to highlight are `pamWithAdjust` and `qualityTable`. `pamWithAdjust` takes a path to a file that contains a dissimilarity matrix and a number, `k`, that represents the number of clusters we should run PAM with. A sample execution of the command can be seen in Listing 12, abbreviated slightly in the output.

Listing 12: Sample execution of `pamWithAdjust`

```
$ BdTools pamWithAdjust ../1000_us_equities_mp.txt 10
Running PAM for 1000 by 1000 matrix with 10 groups.
Resulting grouping found after running PAM:
[1,7,5,1,7,3,9,8,8,8,7,5,3,5,1,4,2,10,7,7,5,7,5,2,3,1,4,2,2,7,9,5,5,1,5,1,9,1,...

After running adjustment algorithm for 1430 iterations, final grouping is:
[7,5,5,7,5,3,9,9,3,9,5,5,3,5,7,8,5,5,5,5,5,5,5,5,3,7,9,9,5,5,9,5,5,7,5,7,9,7,...
```

The second command mentioned, `qualityTable`, is used to generate Quality Tables that are used to determine the optimal number of groups for the data-set in question. It takes a path to a dissimilarity matrix and automatically groups the data with 2 to 25 groups, then prints the different quality measurements for each resulting grouping before and after adjustment. The output is the equivalent of the tables we will see later in Section 7.2.

Listing 13: Executable commands for BdTools

```
main :: IO ()
main = do
  args <- getArgs
  case args of
    ["test"]                              -> runTestsForAllFiles
    ["bench"]                             -> matrixBench
    ["sample"]                            -> print $ pam sampleMatrix 50
    ["run"              , path, k]        -> runPamFromFile path k
    ["fames"]                             -> runFamesOnSample
    ["fames"            , path, k]        -> runFames path (read k)
    ["compare"          , path, k]        -> compareFAMES path (read k)
    ["adjust"           , path, grouping] -> adjustMatrixFromFileWithGrouping
        path grouping
    ["pamWithAdjust"    , path, k]        -> runPamWithAdjustFromFile path k
    ["famesWithAdjust"  , path, k]        -> runFamesWithAdjustFromFile path k
    ["objective"        , path, grouping] -> calculateObjective path grouping
    ["matrixFromRawCSV" , path]           -> createMatrixFromCSV path
    ["printMatrixFromCSV", rpath, wpath]  -> printMatrixFromCSV rpath wpath
    ["pamAdjustWithData" , path, k, sim]  -> runPamWithAdjustFromData path (read
        k) (read sim)
    ["qualityTable"     , path]           -> measureQuality path
    [path               , k]              -> runFile path (read k)
    _                                     -> putStrLn "Unrecognized command"
```

7.2 QUALITY OF GROUPINGS

The result of running the `qualityTable` function of the application is what we call a Quality Table. The table contains the four quality measurements discussed in Section 3.1 as well as the average within-group correlation and the average correlation to other groups. With the four quality measurements we can calculate normalized values as well as a score and finally a rank, showing which number of clusters should be the optimal choice for the given data-set.

On the following pages are shown the quality tables for the different types of similarity measurements, Multi-Period-, Pearson-, Kendall- and Spearman-correlation. The reader is asked to look at the tables and note what ranges we observe for in-group and other-group correlation, the quality measurements and most importantly the ranking for the different number of groups.

The most interesting thing to note that while 2 groups might be the most common result for the optimal number of groups (discussed later), it is not the case that we get worse results with increasing number of groups. Most notably, 25 groups is ranked very high for Pearson correlation.

In the next section we shall look more closely at one particular grouping so that we can see some more qualitative results. In order to compare those results with the traditional Morningstar Style-box classification, we choose to look closer at the Multi-Period 9-group clustering with the adjustment algorithm applied, which is ranked 9th best of the 25.

Using Multi-Period Correlation

Unadjusted grouping

										NORMALIZED VALUES				
										15.00%	15.00%	60.00%	10.00%	**100.00%**
Number of Groups	Composite Rank	Composite BdScore	Composite Value	Average Correlation Within Group	Average Correlation to Other Groups	Classification Quality (CQ) Index	Diversification Efficiency (DE) Ratio	Grouping Variance Effect	InCorr <= ExCorr	Classification Quality (CQ) Index	Diversification Efficiency (DE) Ratio	Grouping Variance Effect	InCorr <= ExCorr	Normalized Composite Value
2	1	100.00	3.16	0.96	0.47	62.22	1.00	-26.66%	22	3.99	4.00	2.78	2.93	3.31
3	2	76.50	2.05	0.96	0.62	58.35	0.71	-25.07%	104	1.88	1.94	2.15	1.88	2.23
4	3	54.45	1.02	0.96	0.70	56.39	0.55	-22.17%	132	0.82	0.89	1.02	1.52	1.21
5	4	40.31	0.36	0.96	0.75	55.24	0.47	-20.02%	113	0.20	0.30	0.18	1.76	0.56
6	6	30.45	-0.11	0.96	0.77	54.64	0.43	-18.99%	211	-0.13	0.00	-0.23	0.51	0.10
7	16	20.32	-0.58	0.96	0.80	54.06	0.38	-17.43%	222	-0.44	-0.32	-0.84	0.37	-0.37
8	20	11.71	-0.98	0.96	0.81	53.65	0.35	-16.25%	268	-0.66	-0.55	-1.30	-0.21	-0.76
9	23	6.69	-1.22	0.96	0.82	53.39	0.33	-15.59%	298	-0.80	-0.68	-1.56	-0.60	-0.99
10	24	0.00	-1.53	0.96	0.84	53.13	0.31	-14.66%	338	-0.95	-0.85	-1.92	-1.11	-1.30
11	17	18.22	-0.68	0.97	0.81	54.01	0.36	-17.69%	330	-0.47	-0.43	-0.74	-1.00	-0.46
12	19	13.16	-0.92	0.97	0.82	53.75	0.34	-16.85%	330	-0.61	-0.57	-1.07	-1.00	-0.70
13	21	9.86	-1.07	0.97	0.83	53.52	0.33	-16.08%	279	-0.73	-0.70	-1.37	-0.35	-0.85
14	22	6.99	-1.21	0.97	0.83	53.33	0.31	-15.60%	273	-0.84	-0.78	-1.56	-0.28	-0.98
15	18	15.66	-0.80	0.97	0.82	53.77	0.34	-17.03%	269	-0.60	-0.59	-1.00	-0.23	-0.58
16	15	21.05	-0.55	0.97	0.81	54.03	0.36	-17.89%	259	-0.46	-0.48	-0.66	-0.10	-0.33
17	11	27.11	-0.26	0.97	0.80	54.34	0.38	-18.89%	256	-0.29	-0.35	-0.27	-0.06	-0.05
18	7	29.77	-0.14	0.97	0.79	54.53	0.39	-19.53%	297	-0.18	-0.27	-0.02	-0.58	0.07
19	5	31.57	-0.05	0.98	0.79	54.63	0.39	-19.83%	296	-0.13	-0.24	0.10	-0.57	0.15
20	8	29.16	-0.17	0.98	0.80	54.52	0.38	-19.46%	304	-0.19	-0.30	-0.04	-0.67	0.04
21	14	25.37	-0.34	0.98	0.80	54.33	0.37	-18.79%	296	-0.29	-0.40	-0.31	-0.57	-0.13
22	10	27.76	-0.23	0.98	0.80	54.45	0.38	-19.17%	290	-0.23	-0.35	-0.16	-0.49	-0.02
23	12	26.91	-0.27	0.98	0.80	54.39	0.37	-18.99%	282	-0.26	-0.38	-0.23	-0.39	-0.06
24	9	28.86	-0.18	0.98	0.80	54.44	0.38	-19.28%	271	-0.23	-0.34	-0.11	-0.25	0.03
25	13	25.86	-0.32	0.98	0.80	54.32	0.37	-18.88%	292	-0.30	-0.40	-0.27	-0.52	-0.11

Figure 19: Quality table for grouping using Multi-Period correlation without adjustment algorithm.

Using Multi-Period Correlation

Adjusted grouping

										NORMALIZED VALUES				
										15.00%	15.00%	60.00%	10.00%	100.00%
Number of Groups	Composite Rank	Composite BdScore	Composite Value	Average Correlation Within Group	Average Correlation to Other Groups	Classification Quality (CQ) Index	Diversification Efficiency (DE) Ratio	Grouping Variance Effect	InCorr <= ExCorr	Classification Quality (CQ) Index	Diversification Efficiency (DE) Ratio	Grouping Variance Effect	InCorr <= ExCorr	Normalized Composite Value
2	1	100.00	2.22	0.96	0.47	62.22	1.00	-26.67%	0	3.99	3.78	1.71	0.30	2.46
3	4	72.57	1.10	0.96	0.62	58.38	0.71	-25.25%	0	1.63	1.54	0.98	0.30	1.21
4	3	74.07	1.16	0.95	0.65	57.37	0.65	-25.98%	0	1.01	1.07	1.36	0.30	1.28
5	2	74.70	1.18	0.94	0.67	56.64	0.62	-26.40%	0	0.56	0.83	1.57	0.30	1.31
6	5	72.02	1.07	0.93	0.68	56.07	0.59	-26.31%	0	0.21	0.63	1.53	0.30	1.19
7	8	49.57	0.15	0.94	0.72	55.37	0.52	-23.79%	0	-0.22	0.07	0.24	0.30	0.17
8	7	53.80	0.33	0.95	0.72	55.61	0.52	-24.29%	0	-0.07	0.07	0.49	0.30	0.36
9	9	48.85	0.12	0.93	0.73	55.05	0.50	-23.86%	0	-0.41	-0.06	0.27	0.30	0.14
10	6	62.25	0.67	0.96	0.72	55.92	0.53	-25.30%	0	0.12	0.13	1.01	0.30	0.75
11	10	48.37	0.10	0.95	0.74	55.31	0.49	-23.76%	0	-0.25	-0.16	0.23	0.30	0.12
12	15	40.09	-0.23	0.96	0.75	55.24	0.47	-22.77%	0	-0.29	-0.35	-0.28	0.30	-0.26
13	12	46.22	0.02	0.96	0.75	55.44	0.48	-23.49%	0	-0.17	-0.26	0.09	0.30	0.02
14	16	37.71	-0.33	0.96	0.76	54.96	0.46	-22.57%	0	-0.47	-0.42	-0.38	0.30	-0.37
15	18	33.75	-0.49	0.96	0.76	54.83	0.44	-22.13%	0	-0.55	-0.51	-0.61	0.30	-0.55
16	24	0.00	-1.88	0.96	0.79	54.18	0.39	-19.27%	111	-0.95	-0.97	-2.07	-3.48	-2.08
17	23	7.52	-1.57	0.97	0.79	54.47	0.40	-20.02%	101	-0.77	-0.86	-1.68	-3.14	-1.74
18	14	40.58	-0.21	0.97	0.76	55.21	0.45	-22.89%	0	-0.32	-0.44	-0.22	0.30	-0.24
19	13	43.18	-0.11	0.97	0.76	55.27	0.46	-23.20%	0	-0.28	-0.40	-0.06	0.30	-0.12
20	11	46.47	0.03	0.97	0.75	55.35	0.47	-23.59%	0	-0.23	-0.35	0.14	0.30	0.03
21	17	37.62	-0.34	0.97	0.76	55.08	0.44	-22.57%	0	-0.39	-0.51	-0.38	0.30	-0.37
22	19	32.22	-0.56	0.97	0.77	54.98	0.43	-21.93%	0	-0.45	-0.62	-0.71	0.30	-0.62
23	20	30.35	-0.63	0.97	0.77	54.86	0.43	-21.73%	0	-0.53	-0.65	-0.81	0.30	-0.70
24	21	26.46	-0.79	0.97	0.78	54.85	0.42	-21.25%	0	-0.53	-0.73	-1.06	0.30	-0.88
25	22	21.39	-1.00	0.97	0.78	54.62	0.40	-20.69%	0	-0.68	-0.82	-1.34	0.30	-1.11

Figure 20: Quality table for grouping using Multi-Period correlation with adjustment algorithm.

Using Pearson Correlation

Unadjusted grouping

										NORMALIZED VALUES				
										15.00%	15.00%	60.00%	10.00%	100.00%
Number of Groups	Composite Rank	Composite BdScore	Composite Value	Average Correlation Within Group	Average Correlation to Other Groups	Classification Quality (CQ) Index	Diversification Efficiency (DE) Ratio	Grouping Variance Effect	InCorr <= ExCorr	Classification Quality (CQ) Index	Diversification Efficiency (DE) Ratio	Grouping Variance Effect	InCorr <= ExCorr	Normalized Composite Value
2	1	100.00	3.27	0.95	0.46	62.25	1.00	-27.01%	14	4.00	4.00	2.92	3.21	3.38
3	2	74.44	2.14	0.95	0.62	58.40	0.71	-25.47%	122	1.91	1.96	2.31	1.74	2.21
4	3	52.50	1.17	0.95	0.70	56.44	0.56	-22.70%	126	0.85	0.93	1.22	1.69	1.21
5	4	35.73	0.43	0.96	0.74	55.30	0.47	-20.42%	142	0.23	0.32	0.33	1.47	0.44
6	8	26.95	0.04	0.96	0.77	54.74	0.43	-19.53%	210	-0.07	0.05	-0.02	0.55	0.04
7	17	15.13	-0.48	0.96	0.79	54.13	0.39	-17.92%	242	-0.40	-0.28	-0.65	0.12	-0.50
8	22	7.74	-0.81	0.96	0.81	53.76	0.36	-16.94%	269	-0.60	-0.48	-1.03	-0.25	-0.83
9	24	0.00	-1.15	0.96	0.82	53.42	0.33	-15.78%	276	-0.79	-0.68	-1.49	-0.35	-1.19
10	15	18.51	-0.33	0.96	0.79	54.28	0.38	-18.67%	270	-0.32	-0.30	-0.35	-0.26	-0.34
11	16	15.86	-0.45	0.96	0.80	54.05	0.37	-18.17%	245	-0.44	-0.40	-0.55	0.08	-0.46
12	21	9.02	-0.75	0.96	0.81	53.77	0.35	-17.25%	275	-0.60	-0.55	-0.91	-0.33	-0.78
13	23	3.29	-1.00	0.96	0.82	53.56	0.33	-16.54%	312	-0.71	-0.66	-1.19	-0.83	-1.04
14	19	11.27	-0.65	0.97	0.81	53.94	0.35	-17.80%	311	-0.51	-0.50	-0.70	-0.82	-0.67
15	18	13.28	-0.56	0.96	0.81	53.90	0.36	-18.18%	313	-0.53	-0.46	-0.55	-0.85	-0.58
16	20	10.41	-0.69	0.96	0.81	53.73	0.35	-17.68%	301	-0.62	-0.54	-0.74	-0.68	-0.71
17	14	19.41	-0.29	0.97	0.80	54.28	0.37	-19.00%	288	-0.32	-0.37	-0.23	-0.51	-0.30
18	11	22.90	-0.14	0.97	0.79	54.43	0.38	-19.46%	270	-0.24	-0.32	-0.05	-0.26	-0.14
19	9	26.88	0.04	0.97	0.79	54.62	0.39	-20.07%	266	-0.14	-0.24	0.19	-0.21	0.04
20	12	21.71	-0.19	0.97	0.80	54.43	0.38	-19.43%	301	-0.24	-0.34	-0.06	-0.68	-0.20
21	10	25.35	-0.03	0.97	0.79	54.59	0.39	-19.96%	291	-0.15	-0.27	0.15	-0.55	-0.03
22	13	21.51	-0.20	0.97	0.80	54.42	0.38	-19.38%	297	-0.25	-0.35	-0.08	-0.63	-0.20
23	7	28.94	0.13	0.98	0.79	54.75	0.40	-20.54%	293	-0.07	-0.20	0.38	-0.58	0.13
24	6	30.73	0.21	0.98	0.78	54.84	0.40	-20.82%	291	-0.02	-0.17	0.49	-0.55	0.22
25	5	32.41	0.28	0.98	0.78	54.93	0.41	-21.07%	288	0.03	-0.14	0.59	-0.51	0.29

Figure 21: Quality table for grouping using Pearson correlation without adjustment algorithm.

Using Pearson Correlation

Adjusted grouping

										NORMALIZED VALUES				
										15.00%	15.00%	60.00%	10.00%	**100.00%**
Number of Groups	Composite Rank	Composite BdScore	Composite Value	Average Correlation Within Group	Average Correlation to Other Groups	Classification Quality (CQ) Index	Diversification Efficiency (DE) Ratio	Grouping Variance Effect	InCorr <= ExCorr	Classification Quality (CQ) Index	Diversification Efficiency (DE) Ratio	Grouping Variance Effect	InCorr <= ExCorr	Normalized Composite Value
2	1	100.00	2.75	0.95	0.46	62.24	1.00	-27.02%	0	3.93	3.85	2.18	1.02	3.06
3	2	68.39	1.56	0.95	0.62	58.40	0.71	-25.60%	0	1.80	1.77	1.63	1.02	1.73
4	3	56.32	1.10	0.94	0.67	56.75	0.62	-25.02%	0	0.88	1.10	1.40	1.02	1.23
5	5	44.77	0.67	0.94	0.71	55.83	0.54	-23.30%	0	0.37	0.54	0.74	1.02	0.74
6	4	47.30	0.76	0.94	0.71	55.73	0.54	-24.39%	0	0.31	0.56	1.16	1.02	0.85
7	18	8.90	-0.68	0.95	0.78	54.15	0.40	-18.46%	90	-0.56	-0.47	-1.13	-0.57	-0.76
8	24	0.00	-1.02	0.95	0.80	53.78	0.37	-17.45%	120	-0.77	-0.68	-1.52	-1.10	-1.13
9	19	7.64	-0.73	0.94	0.80	53.63	0.38	-18.09%	67	-0.86	-0.63	-1.27	-0.16	-0.81
10	21	5.83	-0.80	0.96	0.78	54.35	0.40	-19.66%	151	-0.46	-0.43	-0.66	-1.64	-0.89
11	12	19.08	-0.30	0.95	0.77	54.48	0.43	-21.14%	85	-0.38	-0.24	-0.09	-0.48	-0.33
12	6	40.52	0.51	0.94	0.74	55.05	0.48	-23.87%	1	-0.06	0.13	0.96	1.00	0.57
13	22	1.56	-0.96	0.96	0.81	53.69	0.35	-17.69%	103	-0.82	-0.79	-1.43	-0.80	-1.07
14	23	0.87	-0.98	0.96	0.80	53.97	0.37	-18.45%	140	-0.66	-0.69	-1.13	-1.45	-1.10
15	20	6.48	-0.77	0.96	0.80	54.03	0.37	-18.88%	106	-0.63	-0.64	-0.97	-0.85	-0.86
16	11	26.00	-0.04	0.95	0.78	54.27	0.41	-20.70%	1	-0.50	-0.40	-0.26	1.00	-0.04
17	16	13.33	-0.52	0.96	0.79	54.36	0.39	-19.73%	82	-0.45	-0.55	-0.64	-0.42	-0.57
18	13	17.12	-0.37	0.97	0.79	54.51	0.40	-20.23%	69	-0.37	-0.49	-0.45	-0.20	-0.42
19	14	16.52	-0.40	0.97	0.78	54.71	0.41	-20.88%	99	-0.26	-0.40	-0.20	-0.72	-0.44
20	9	35.62	0.32	0.96	0.76	55.08	0.44	-22.64%	0	-0.05	-0.16	0.49	1.02	0.36
21	17	10.72	-0.61	0.97	0.78	54.63	0.40	-20.50%	134	-0.30	-0.47	-0.34	-1.34	-0.68
22	10	34.42	0.28	0.97	0.76	55.01	0.43	-22.43%	1	-0.09	-0.21	0.41	1.00	0.31
23	7	37.30	0.39	0.97	0.76	55.26	0.44	-22.97%	0	0.05	-0.14	0.61	1.02	0.43
24	15	13.36	-0.51	0.97	0.78	54.88	0.41	-21.26%	141	-0.16	-0.38	-0.05	-1.47	-0.57
25	8	36.74	0.37	0.97	0.76	55.27	0.44	-22.82%	0	0.06	-0.17	0.55	1.02	0.41

Figure 22: Quality table for grouping using Pearson correlation with adjustment algorithm.

Using Kendall Correlation

Unadjusted grouping

										NORMALIZED VALUES				
										15.00%	15.00%	70.00%	0.00%	**100.00%**
Number of Groups	Composite Rank	Composite BdScore	Composite Value	Average Correlation Within Group	Average Correlation to Other Groups	Classification Quality (CQ) Index	Diversification Efficiency (DE) Ratio	Grouping Variance Effect	InCorr <= ExCorr	Classification Quality (CQ) Index	Diversification Efficiency (DE) Ratio	Grouping Variance Effect	InCorr <= ExCorr	Normalized Composite Value
2	24	0.00	-0.94	0.14	0.06	51.93	1.14	-46.90%	598	-1.13	2.28	-2.16	1.84	-1.56
3	6	79.90	0.59	0.17	0.10	51.76	1.09	-59.73%	328	-1.59	1.92	0.27	3.77	0.98
4	5	79.99	0.59	0.22	0.15	51.69	1.03	-63.81%	851	-1.79	1.53	1.05	0.03	0.98
5	2	97.38	0.92	0.25	0.18	51.89	1.00	-65.89%	784	-1.23	1.30	1.44	0.51	1.53
6	1	100.00	0.97	0.29	0.21	52.06	0.96	-65.81%	738	-0.77	1.00	1.42	0.84	1.62
7	3	96.81	0.91	0.32	0.24	52.05	0.93	-65.50%	724	-0.79	0.78	1.37	0.94	1.52
8	4	84.77	0.68	0.35	0.26	52.16	0.90	-64.79%	957	-0.48	0.58	1.23	-0.73	1.13
9	7	75.98	0.51	0.37	0.29	52.13	0.87	-63.41%	902	-0.57	0.35	0.97	-0.34	0.85
10	8	73.34	0.46	0.39	0.30	52.21	0.85	-62.72%	880	-0.35	0.20	0.84	-0.18	0.77
11	9	63.93	0.28	0.41	0.33	52.19	0.82	-61.24%	844	-0.40	0.00	0.56	0.08	0.47
12	10	63.44	0.27	0.42	0.33	52.21	0.81	-61.30%	871	-0.34	-0.04	0.57	-0.11	0.46
13	11	58.98	0.19	0.43	0.34	52.19	0.80	-60.64%	851	-0.40	-0.14	0.45	0.03	0.31
14	12	51.71	0.05	0.45	0.36	52.28	0.78	-59.55%	895	-0.15	-0.27	0.24	-0.29	0.08
15	13	43.10	-0.11	0.47	0.38	52.33	0.76	-58.19%	907	-0.01	-0.43	-0.02	-0.37	-0.19
16	14	37.85	-0.21	0.49	0.39	52.45	0.74	-57.11%	924	0.34	-0.55	-0.22	-0.49	-0.36
17	15	36.92	-0.23	0.50	0.40	52.56	0.73	-56.61%	919	0.64	-0.62	-0.32	-0.46	-0.39
18	16	30.78	-0.35	0.52	0.41	52.60	0.72	-55.62%	928	0.76	-0.73	-0.50	-0.52	-0.58
19	17	27.40	-0.41	0.52	0.42	52.63	0.71	-55.10%	940	0.84	-0.79	-0.60	-0.61	-0.69
20	18	17.72	-0.60	0.53	0.43	52.53	0.69	-53.99%	938	0.55	-0.90	-0.81	-0.59	-1.00
21	20	11.80	-0.71	0.54	0.44	52.56	0.68	-53.08%	948	0.64	-1.00	-0.99	-0.66	-1.18
22	22	11.18	-0.72	0.56	0.45	52.66	0.67	-52.61%	943	0.94	-1.05	-1.07	-0.63	-1.20
23	23	10.54	-0.74	0.57	0.46	52.83	0.66	-52.02%	949	1.40	-1.11	-1.19	-0.67	-1.22
24	19	13.02	-0.69	0.58	0.46	52.96	0.66	-51.97%	947	1.76	-1.13	-1.20	-0.66	-1.15
25	21	11.37	-0.72	0.59	0.47	53.10	0.65	-51.31%	955	2.15	-1.20	-1.32	-0.71	-1.20

Figure 23: Quality table for grouping using Kendall correlation without adjustment algorithm.

Using Kendall Correlation

Adjusted grouping

										NORMALIZED VALUES				
										15.00%	15.00%	70.00%	0.00%	**100.00%**
Number of Groups	Composite Rank	Composite BdScore	Composite Value	Average Correlation Within Group	Average Correlation to Other Groups	Classification Quality (CQ) Index	Diversification Efficiency (DE) Ratio	Grouping Variance Effect	InCorr <= ExCorr	Classification Quality (CQ) Index	Diversification Efficiency (DE) Ratio	Grouping Variance Effect	InCorr <= ExCorr	Normalized Composite Value
2	19	38.00	-0.32	0.54	0.22	58.19	1.00	-39.15%	999	2.07	1.39	-1.20	-0.70	-0.47
3	4	87.46	0.84	0.17	0.11	51.72	1.15	-59.64%	329	-0.92	2.17	0.94	4.41	1.22
4	3	94.39	1.01	0.24	0.17	51.88	1.07	-62.59%	884	-0.85	1.75	1.25	0.17	1.46
5	2	99.60	1.13	0.27	0.19	52.03	1.03	-64.52%	844	-0.78	1.55	1.45	0.48	1.64
6	1	100.00	1.14	0.30	0.22	52.11	1.00	-64.97%	789	-0.74	1.36	1.49	0.90	1.65
7	21	10.14	-0.98	0.87	0.58	57.23	0.54	-36.26%	993	1.63	-1.13	-1.50	-0.66	-1.42
8	20	12.59	-0.92	0.88	0.58	57.52	0.54	-36.80%	991	1.76	-1.15	-1.45	-0.64	-1.34
9	22	6.13	-1.07	0.89	0.60	57.27	0.51	-35.30%	990	1.65	-1.31	-1.61	-0.64	-1.56
10	23	0.71	-1.20	0.90	0.62	57.06	0.48	-34.02%	989	1.55	-1.45	-1.74	-0.63	-1.74
11	24	0.00	-1.22	0.91	0.63	57.12	0.48	-33.82%	988	1.58	-1.49	-1.76	-0.62	-1.77
12	5	77.96	0.62	0.44	0.35	52.33	0.83	-59.53%	877	-0.64	0.45	0.93	0.23	0.90
13	6	73.28	0.51	0.46	0.37	52.30	0.81	-58.30%	870	-0.66	0.33	0.80	0.28	0.74
14	7	69.40	0.42	0.48	0.38	52.38	0.79	-57.19%	892	-0.61	0.22	0.68	0.11	0.61
15	8	64.40	0.30	0.50	0.40	52.46	0.77	-55.78%	908	-0.58	0.09	0.53	-0.01	0.43
16	9	60.95	0.22	0.52	0.42	52.51	0.75	-54.80%	932	-0.56	0.00	0.43	-0.19	0.32
17	10	58.57	0.16	0.53	0.43	52.59	0.74	-54.09%	920	-0.52	-0.07	0.36	-0.10	0.24
18	11	55.39	0.09	0.54	0.44	52.68	0.72	-53.15%	930	-0.48	-0.15	0.26	-0.18	0.13
19	12	54.60	0.07	0.55	0.44	52.69	0.72	-52.95%	931	-0.47	-0.18	0.24	-0.19	0.10
20	13	50.69	-0.02	0.56	0.45	52.62	0.70	-51.93%	934	-0.50	-0.27	0.13	-0.21	-0.03
21	14	47.91	-0.09	0.57	0.46	52.62	0.69	-51.17%	949	-0.51	-0.33	0.05	-0.32	-0.13
22	15	47.48	-0.10	0.57	0.47	52.69	0.69	-51.00%	945	-0.47	-0.35	0.04	-0.29	-0.14
23	16	44.41	-0.17	0.59	0.48	52.88	0.67	-50.00%	960	-0.38	-0.44	-0.07	-0.41	-0.25
24	17	44.00	-0.18	0.60	0.48	53.03	0.67	-49.78%	952	-0.32	-0.46	-0.09	-0.35	-0.26
25	18	42.37	-0.22	0.61	0.49	53.17	0.66	-49.21%	964	-0.25	-0.51	-0.15	-0.44	-0.32

Figure 24: Quality table for grouping using Kendall correlation with adjustment algorithm.

Using Spearman Correlation

Unadjusted grouping

										NORMALIZED VALUES				
										15.00%	15.00%	70.00%	0.00%	**100.00%**
Number of Groups	Composite Rank	Composite BdScore	Composite Value	Average Correlation Within Group	Average Correlation to Other Groups	Classification Quality (CQ) Index	Diversification Efficiency (DE) Ratio	Grouping Variance Effect	InCorr <= ExCorr	Classification Quality (CQ) Index	Diversification Efficiency (DE) Ratio	Grouping Variance Effect	InCorr <= ExCorr	Normalized Composite Value
2	1	100.00	2.11	0.50	0.27	55.84	1.23	-36.50%	603	3.76	3.15	1.53	4.52	2.19
3	2	97.24	2.02	0.56	0.40	54.00	1.00	-39.68%	823	1.69	2.09	2.08	0.88	2.11
4	3	86.07	1.69	0.63	0.49	53.49	0.86	-38.45%	893	1.11	1.44	1.87	-0.28	1.76
5	4	73.61	1.32	0.67	0.54	53.10	0.77	-36.48%	846	0.68	1.00	1.52	0.50	1.37
6	5	63.19	1.01	0.70	0.58	52.81	0.70	-34.71%	858	0.36	0.68	1.22	0.30	1.05
7	6	55.91	0.79	0.72	0.61	52.70	0.65	-33.34%	906	0.23	0.47	0.98	-0.49	0.82
8	7	46.78	0.52	0.74	0.64	52.49	0.61	-31.66%	913	-0.01	0.26	0.69	-0.61	0.54
9	8	38.98	0.28	0.75	0.66	52.35	0.57	-30.16%	894	-0.16	0.08	0.42	-0.30	0.30
10	9	31.28	0.05	0.77	0.68	52.19	0.54	-28.70%	878	-0.34	-0.08	0.17	-0.03	0.06
11	13	24.52	-0.15	0.78	0.70	52.02	0.51	-27.44%	880	-0.53	-0.22	-0.05	-0.06	-0.15
12	16	20.20	-0.28	0.79	0.71	52.00	0.49	-26.54%	922	-0.56	-0.31	-0.21	-0.76	-0.29
13	18	14.87	-0.44	0.80	0.72	51.93	0.46	-25.47%	907	-0.64	-0.42	-0.39	-0.51	-0.45
14	19	10.77	-0.56	0.81	0.73	51.86	0.45	-24.66%	896	-0.72	-0.50	-0.54	-0.33	-0.58
15	21	7.58	-0.65	0.82	0.74	51.83	0.43	-23.99%	896	-0.74	-0.57	-0.65	-0.33	-0.68
16	22	3.61	-0.77	0.82	0.75	51.75	0.42	-23.23%	896	-0.84	-0.64	-0.78	-0.33	-0.80
17	23	2.34	-0.81	0.82	0.76	51.71	0.41	-23.01%	893	-0.89	-0.67	-0.82	-0.28	-0.84
18	24	0.00	-0.88	0.83	0.76	51.73	0.40	-22.47%	896	-0.87	-0.72	-0.92	-0.33	-0.92
19	20	8.30	-0.74	0.84	0.76	52.09	0.41	-23.08%	895	-0.46	-0.67	-0.81	-0.31	-0.77
20	17	15.22	-0.77	0.84	0.76	52.12	0.40	-22.83%	887	-0.42	-0.70	-0.85	-0.18	-0.80
21	14	23.87	-0.82	0.85	0.76	52.11	0.40	-22.44%	892	-0.44	-0.74	-0.92	-0.26	-0.86
22	11	27.24	-0.72	0.86	0.76	52.38	0.40	-22.86%	891	-0.14	-0.71	-0.85	-0.25	-0.75
23	10	30.45	-0.63	0.86	0.76	52.63	0.41	-23.26%	890	0.15	-0.68	-0.78	-0.23	-0.65
24	15	23.19	-0.84	0.86	0.77	52.28	0.39	-22.08%	887	-0.24	-0.78	-0.98	-0.18	-0.88
25	12	26.37	-0.75	0.87	0.77	52.53	0.39	-22.48%	884	0.03	-0.75	-0.91	-0.13	-0.78

Figure 25: Quality table for grouping using Spearman correlation without adjustment algorithm.

Using Spearman Correlation

Adjusted grouping

										NORMALIZED VALUES				
										15.00%	15.00%	70.00%	0.00%	**100.00%**
Number of Groups	Composite Rank	Composite BdScore	Composite Value	Average Correlation Within Group	Average Correlation to Other Groups	Classification Quality (CQ) Index	Diversification Efficiency (DE) Ratio	Grouping Variance Effect	InCorr <= ExCorr	Classification Quality (CQ) Index	Diversification Efficiency (DE) Ratio	Grouping Variance Effect	InCorr <= ExCorr	Normalized Composite Value
2	1	100.00	2.14	0.52	0.28	56.07	1.23	-36.21%	705	3.78	3.16	1.57	4.27	2.22
3	2	97.46	2.06	0.58	0.41	54.19	1.00	-39.35%	816	1.76	2.10	2.12	1.73	2.14
4	3	85.97	1.72	0.63	0.49	53.51	0.86	-38.23%	907	1.02	1.47	1.92	-0.35	1.78
5	4	72.19	1.30	0.68	0.55	53.19	0.76	-35.86%	884	0.68	0.98	1.51	0.17	1.35
6	5	61.15	0.97	0.71	0.59	52.84	0.69	-34.00%	883	0.31	0.66	1.18	0.20	1.01
7	6	54.80	0.78	0.73	0.62	52.79	0.65	-32.77%	907	0.25	0.46	0.97	-0.35	0.81
8	7	45.56	0.51	0.75	0.64	52.55	0.60	-31.08%	910	-0.01	0.25	0.67	-0.42	0.52
9	8	36.43	0.23	0.77	0.67	52.43	0.56	-29.26%	905	-0.14	0.04	0.35	-0.31	0.24
10	9	29.36	0.02	0.78	0.69	52.27	0.53	-27.93%	895	-0.32	-0.11	0.12	-0.08	0.02
11	10	23.62	-0.15	0.79	0.70	52.12	0.50	-26.85%	884	-0.47	-0.23	-0.07	0.17	-0.16
12	11	19.94	-0.26	0.80	0.72	52.11	0.48	-26.07%	917	-0.48	-0.31	-0.21	-0.58	-0.27
13	12	13.97	-0.44	0.81	0.73	52.02	0.46	-24.88%	901	-0.59	-0.43	-0.41	-0.22	-0.46
14	13	9.09	-0.59	0.82	0.74	51.95	0.44	-23.90%	910	-0.66	-0.53	-0.59	-0.42	-0.61
15	15	6.82	-0.66	0.82	0.75	51.89	0.43	-23.47%	915	-0.72	-0.57	-0.66	-0.54	-0.68
16	20	2.41	-0.79	0.83	0.76	51.82	0.41	-22.59%	916	-0.80	-0.66	-0.82	-0.56	-0.82
17	22	1.69	-0.81	0.83	0.76	51.73	0.41	-22.54%	910	-0.89	-0.67	-0.82	-0.42	-0.84
18	24	0.00	-0.86	0.84	0.77	51.74	0.40	-22.16%	908	-0.88	-0.71	-0.89	-0.38	-0.89
19	17	4.50	-0.73	0.84	0.76	52.12	0.41	-22.72%	914	-0.48	-0.66	-0.79	-0.51	-0.75
20	19	3.81	-0.75	0.85	0.76	52.11	0.40	-22.58%	896	-0.48	-0.68	-0.82	-0.10	-0.78
21	21	1.88	-0.81	0.85	0.77	52.12	0.39	-22.15%	904	-0.48	-0.72	-0.89	-0.29	-0.84
22	16	5.28	-0.70	0.86	0.76	52.39	0.40	-22.58%	900	-0.19	-0.69	-0.82	-0.19	-0.73
23	14	8.65	-0.60	0.87	0.76	52.65	0.41	-23.02%	899	0.10	-0.66	-0.74	-0.17	-0.63
24	23	0.87	-0.84	0.86	0.77	52.29	0.38	-21.73%	904	-0.29	-0.77	-0.97	-0.29	-0.87
25	18	3.94	-0.74	0.87	0.77	52.54	0.39	-22.12%	907	-0.03	-0.74	-0.90	-0.35	-0.77

Figure 26: Quality table for grouping using Spearman correlation with adjustment algorithm.

7.3 EXAMPLE GROUPING OF US EQUITY FUNDS

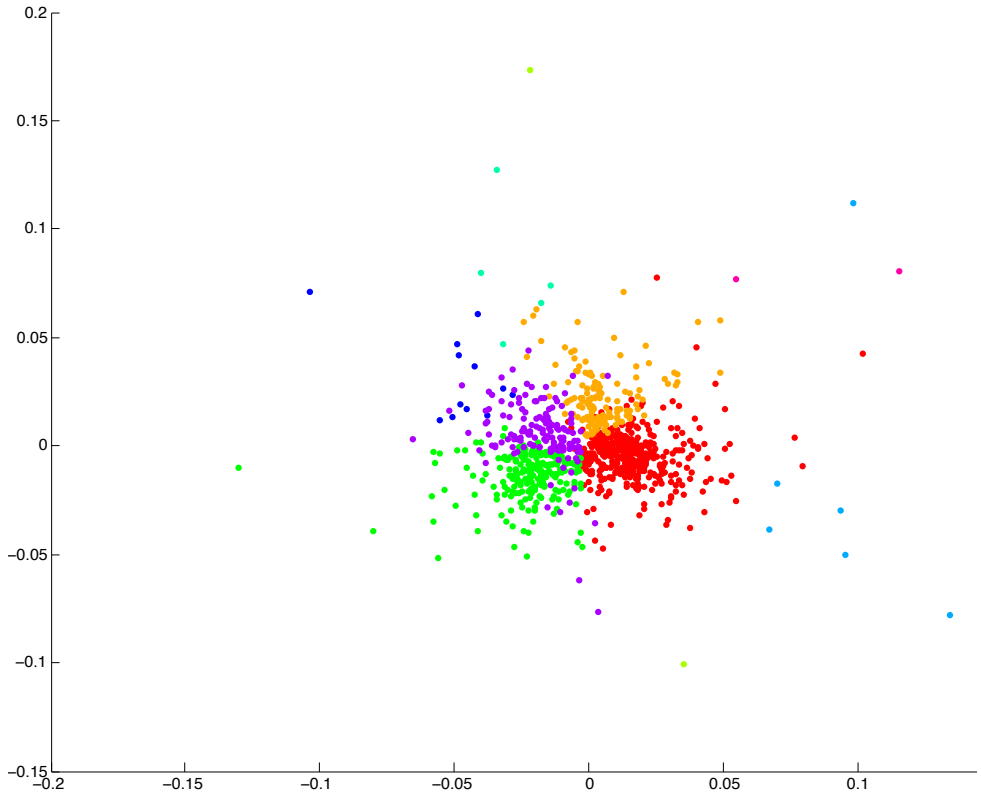


Figure 27: Visualization of sample grouping.

In order to have some concrete examples to point to and to show what the actual result of a clustering on real data looks like, in this section we see the 9-group clustering based on Multi-Period correlation. In other words, this is 1000 US equity funds clustered with Multi-Period correlation as the sole measure of similarity and it is clustered into 9 groups.

Figure 27 is the full picture of the clustering and Figure 28 is slightly zoomed in so that we can see the core of the cluster slightly better, cutting off the most extreme outliers. The graph is calculated using `MATLAB`'s function `mdscale` and is a visualization of the sample data with the different colors representing different groups.

One thing to keep in mind is that `mdscale` tries to perform a least-squares optimization to fit the data into a 2-dimensional space, but the data is in fact not 2-dimensional and the accuracy of `mdscale` is only to about 3 decimals of precision. In the graph we can see that the whole cluster only spans a range of about $[-0.08, 0.08]$ and as such the points at the centre of the cluster might not be placed with adequately high accuracy. The reason for the image is however not to try to distinguish any real results, but to get an idea of the applicability of these results and how one might imagine what the data looks like.

In the results results in Figure 28 it looks like a lot of items from different groups are intermingled, the reason for this is most likely that it is rendered in 2 dimensions so the algorithm is simply not able to clearly separate the values.

If we look at the illustration in 3 dimensions instead, as in Figure 29 we can see that all items actually are separated, and in 3D the cluster looks like a slightly elongated ball (like an

American football) with outliers dotted around it. The 3D illustration is however very hard to convey on a paper without the ability to interactively move around in the structure.

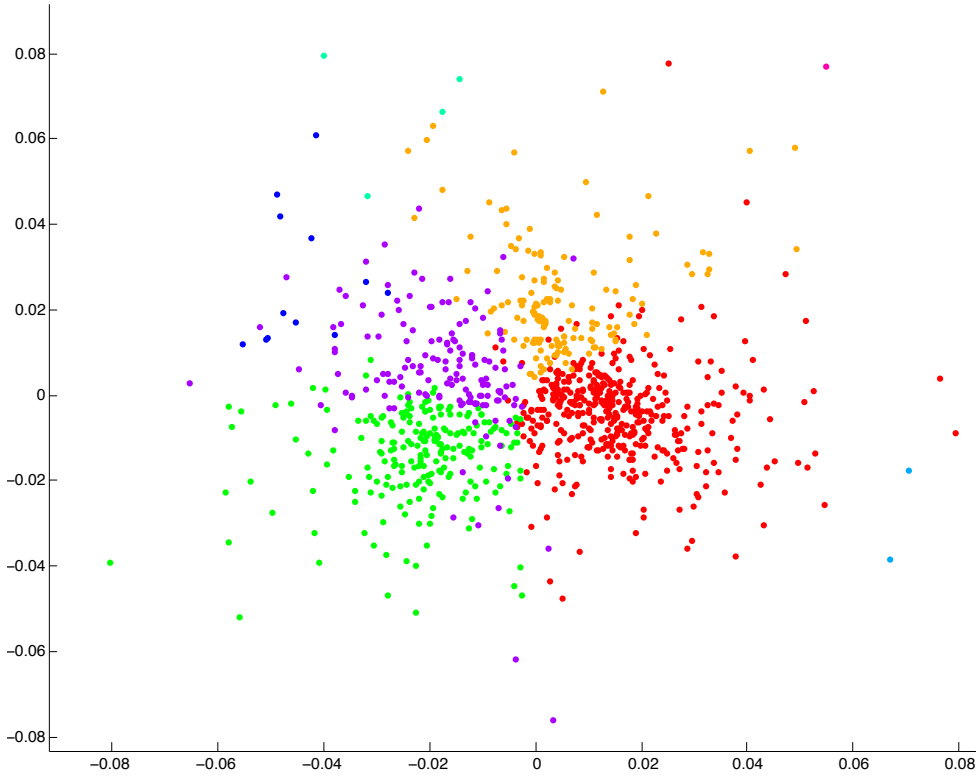


Figure 28: Visualization of sample grouping, zoomed in.

The results of our clustering is simply which security belongs to which cluster. However, based on that information we can gather another interesting result, which is to say how many securities belong to each cluster. In Table 12 the count of securities for each cluster is stated before and after applying the adjustment algorithm.

Table 12: Item count for each cluster

Group #	1	2	3	4	5	6	7	8	9
Pre-adjustment	240	133	149	108	26	39	104	112	89
Post-adjustment	450	146	2	223	5	6	13	153	2

One can note that there are 5 groups essentially rendered meaningless post-adjustment, groups 3, 5, 6, 7 and 9 get so few values in them that they can't be considered real groups. If we color the graph such that these groups are one color and everything else is another, we can very clearly see that the clustering algorithm has identified these groups as severe outliers to the rest of the data, they can simply not efficiently fit into any other group.

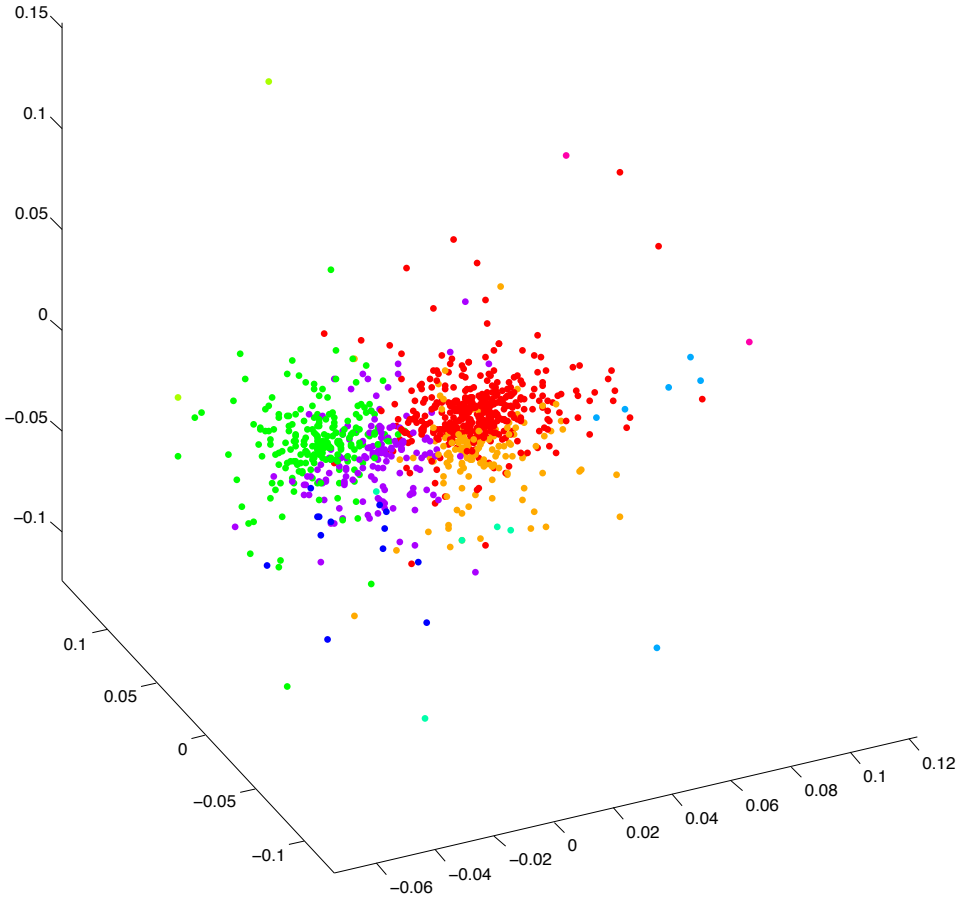


Figure 29: Visualization of sample grouping, in 3 dimensions.

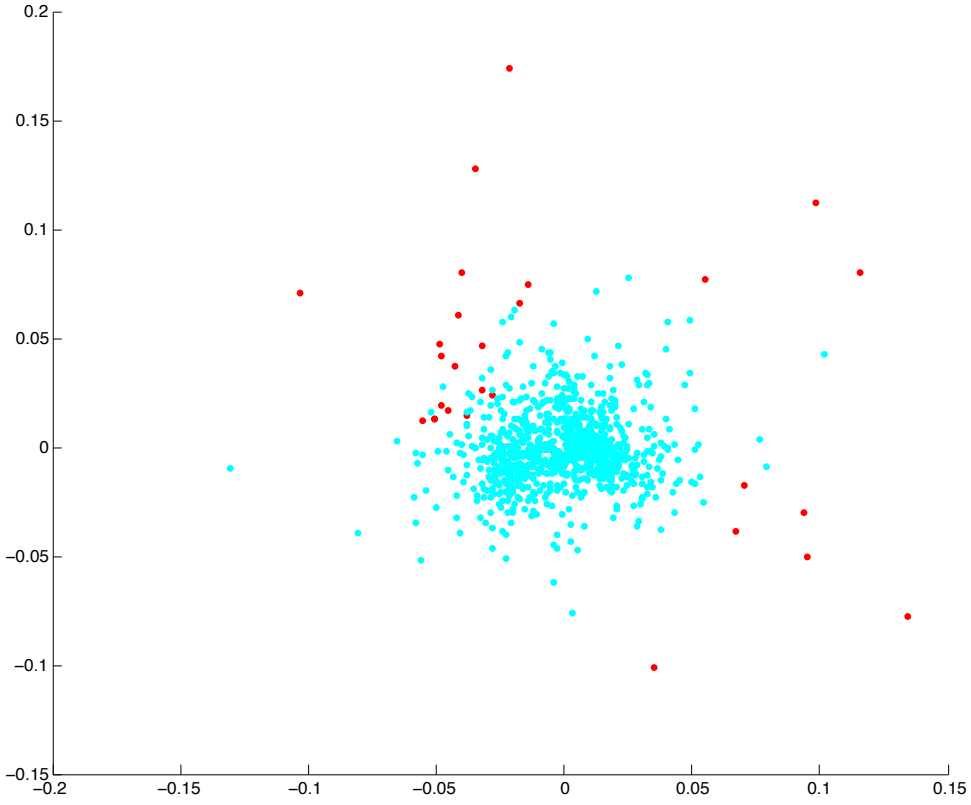


Figure 30: Visualization of outlier groups.

7.3.1 Comparing to Morningstar Style-box categorization

When we're studying these results it is important to remember the starting point of our hypothesis; that it is better to use this kind of similarity based clustering than to "arbitrarily" divide the securities into a fixed set of groups, like the Morningstar Style-box.

The Style-box methodology is fundamentally a holdings-based grouping analysis [Kaplan, 2003]. Morningstar is essentially projecting their holdings-measurements to x and y coordinates (what they call size and value). Then they perform a k-means clustering on the result of that. However, the scale of their x- and y-axis are fixed such that a pre-determined percentage of securities will fall under the different kinds of groups.

The Morningstar methodology is the most commonly used one and it is therefore important to try to observe how using K-medoids clustering with similarity compares to the Style-box. If we take the same visualization as for our Multi-Period correlation-based similarity data and now color the data based on the Morningstar Style-box category, we see the results in Figure 31. Compare this with the pre-adjustment clustering into 9 groups with PAM in Figure 32.

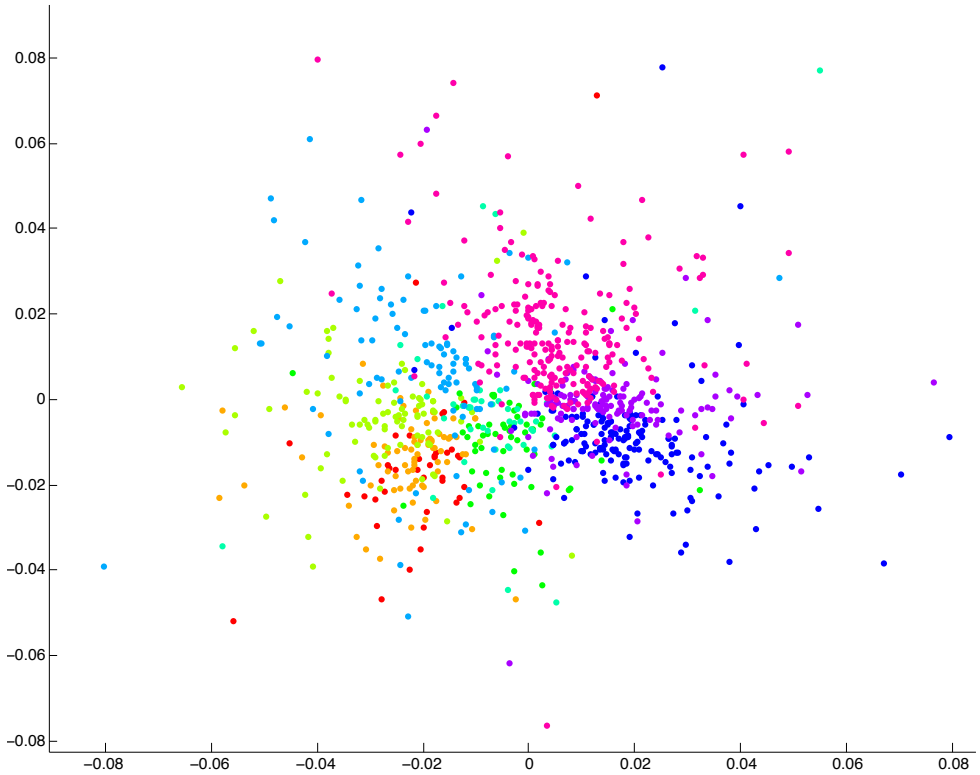


Figure 31: Visualizing the Morningstar Style-box categorization.

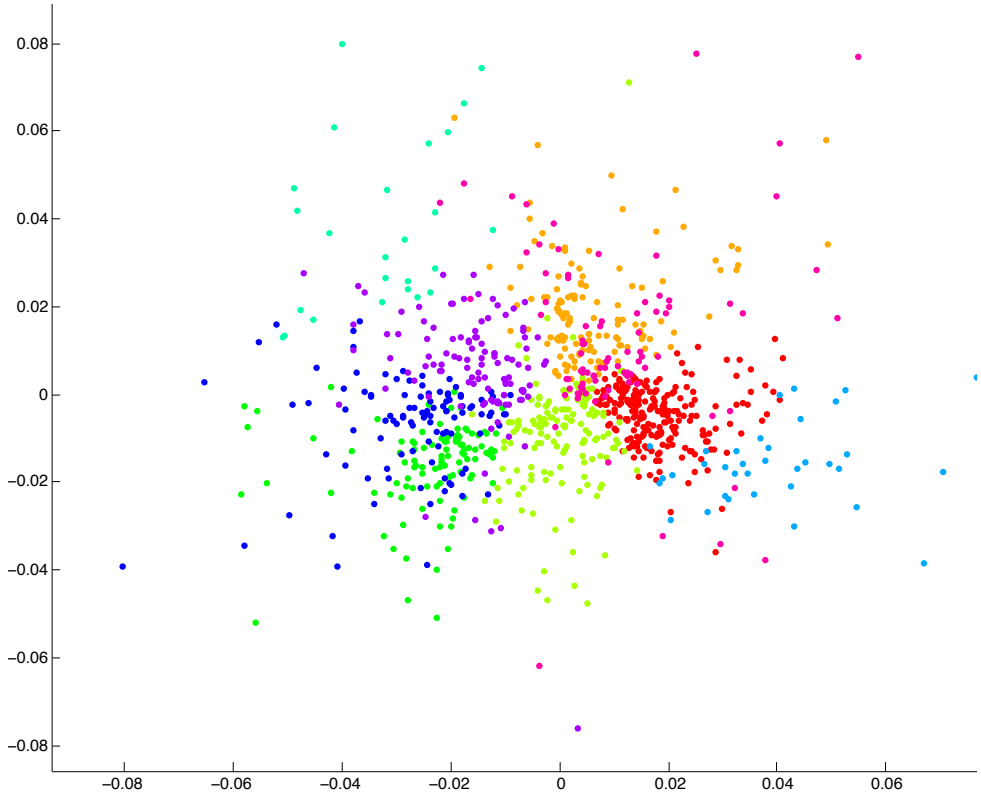


Figure 32: Visualizing the 9-group PAM clustering without adjustment.

It does look surprisingly similar to what we have achieved with Multi-Period and PAM. The visualization doesn't tell a very accurate story however, so let's try to break down the comparison a little bit further. In Table 13 we can see the number of securities in each of the 9 Style-box categories. The three last groups, 7, 8 and 9, are the largest and they are Large Value, Large Blend and Large Growth, respectively. These then make out 58% of the securities (as mentioned earlier, Morningstar aims to have 70% of securities in the Large size categories).

Table 13: Group sizes for the Morningstar Style-box categorization and PAM with 9 groups.

Group #	1	2	3	4	5	6	7	8	9
Morningstar Count	41	81	87	59	49	101	180	165	237
PAM unadjusted	26	108	112	89	39	104	149	133	240
PAM adjusted	2	6	13	5	2	146	153	223	450

We re-iterate the count for the Multi-Period correlation-based clustering into 9 groups, but re-ordering the groups, and we can see that the results are again very similar for Morningstar and the unadjusted PAM results. Note that the ordering of the groups is arbitrary.

We can break down each group in the PAM results by which Style-box category they belong. By doing so we can gain some insight into what categories the groups belong, if there is any consistency with Morningstar or if it's all spread out over the Style-box categories.

Group 1	Value	Blend	Growth
Large	39%	53%	8%
Mid	0%	0%	0%
Small	0%	0%	0%

Group 2	Value	Blend	Growth
Large	1%	6%	90%
Mid	0%	1%	0%
Small	1%	0%	2%

Group 3	Value	Blend	Growth
Large	14%	24%	9%
Mid	26%	17%	0%
Small	3%	1%	6%

Group 4	Value	Blend	Growth
Large	0%	0%	0%
Mid	1%	5%	10%
Small	33%	50%	1%

Group 5	Value	Blend	Growth
Large	0%	0%	31%
Mid	0%	0%	0%
Small	0%	0%	69%

Group 6	Value	Blend	Growth
Large	74%	21%	5%
Mid	0%	0%	0%
Small	0%	0%	0%

Group 7	Value	Blend	Growth
Large	0%	1%	0%
Mid	3%	10%	54%
Small	4%	21%	8%

Group 8	Value	Blend	Growth
Large	0%	1%	14%
Mid	4%	19%	7%
Small	0%	1%	54%

Group 9	Value	Blend	Growth
Large	8%	39%	33%
Mid	4%	4%	2%
Small	0%	0%	9%

Figure 33: Comparing groups of PAM with Style-box category without adjustment.

We can see in Figure 33 that groups 1, 4 and 9 all belong to Large, Small and Large styles respectively. They share about half with the Blend style and half with some other style of the same size. Groups 2, 5 and 6 seem to belong to quite specific style-box categories, where group 2 is the strongest relationship with 90% Large-Growth securities. The other groups seem to be much more spread out, but it is rare that any one group covers more than 3 or 4 categories.

We can see the same thing for the adjusted results, but bear in mind that it is really only groups 1, 2, 4 and 8 that are relevant in this example. Interestingly, group 2 maintains a very high amount (91%) of Large-Growth securities. The other results seem more vague and spread out.

Group 1	Value	Blend	Growth
Large	32%	46%	13%
Mid	5%	3%	0%
Small	0%	0%	1%

Group 2	Value	Blend	Growth
Large	0%	2%	91%
Mid	0%	1%	1%
Small	0%	1%	5%

Group 3	Value	Blend	Growth
Large	0%	0%	0%
Mid	0%	0%	0%
Small	100%	0%	0%

Group 4	Value	Blend	Growth
Large	0%	0%	0%
Mid	9%	12%	22%
Small	19%	34%	4%

Group 5	Value	Blend	Growth
Large	0%	0%	60%
Mid	0%	20%	0%
Small	0%	0%	20%

Group 6	Value	Blend	Growth
Large	83%	0%	17%
Mid	0%	0%	0%
Small	0%	0%	0%

Group 7	Value	Blend	Growth
Large	0%	0%	8%
Mid	0%	0%	15%
Small	0%	0%	77%

Group 8	Value	Blend	Growth
Large	0%	3%	7%
Mid	7%	17%	17%
Small	0%	2%	47%

Group 9	Value	Blend	Growth
Large	50%	0%	0%
Mid	50%	0%	0%
Small	0%	0%	0%

Figure 34: Comparing groups of PAM with Style-box category with adjustment.

Lastly, if we look at the quality measurements for the Morningstar clustering as applied to the Multi-Period correlation based data as well as the Pearson correlation based data together with the 9-group PAM clustering, we get the results seen in Figure 35 and Figure 36.

Measurement	Average Correlation Within Group	Average Correlation to Other Groups	Classification Quality (CQ) Index	Grouping Variance Effect	InCorr <= ExCorr
Morningstar	0.961	0.839	53.061	-14.333%	479
9-group PAM	0.960	0.825	53.389	-15.588%	298
9-group PAM adjusted	0.934	0.732	55.048	-19.265%	0

Figure 35: Comparing quality of Morningstar categorization with PAM using Multi-Period correlation.

Measurement	Average Correlation Within Group	Average Correlation to Other Groups	Classification Quality (CQ) Index	Grouping Variance Effect	InCorr <= ExCorr
Morningstar	0.956	0.830	53.161	-15.150%	475
9-group PAM	0.959	0.823	53.417	-15.777%	276
9-group PAM adjusted	0.942	0.797	53.626	-18.087%	67

Figure 36: Comparing quality of Morningstar categorization with PAM using Pearson correlation.

The quality is strikingly similar between all examples. However, it is clear that the adjustment algorithm makes the PAM clustering outperform both regular PAM and the Morningstar categorization on all quality measurements. As seen in the quality tables earlier, we could move to using 12 groups for the adjusted PAM clustering and increase our overall quality *significantly*.

8 DISCUSSION

Lets remind ourselves of the original motivation for this project.

The current state of security classification and subsequently portfolio selection is heavily dependent on largely arbitrary divisions. In order to get proper diversification one has to put in a lot of work trying to find a diverse selection of securities for ones portfolio. You could try to select securities based on country, industry and company size, but by doing so you could very well fool yourself into thinking that you are more diversified than you actually are.

True risk is not measured by one property, but by an aggregation of tens if not hundreds of measurements. There is simply no way for any human to consider all the data that exists out there. The result is that many turn to short-cuts such as the Morningstar Style-box categorization. There is an inherit problem with such a system, no matter how well they are diversifying between groups, they are ultimately limited to a certain number of groups, a number that might very well be a bad fit for the data considered. At each portfolio creation, people might not want to consider all securities in existence for their categorization. If the Style-box methodology could be adapted to be fitted for a given data-set of securities we actually want to look at, then they could perhaps improve diversification, but are still ultimately limited to the number of groups embedded in their method.

In this thesis we have tried to present a methodology to classifying securities that can

- take any number of similarity measurements into consideration
- for *a given* set of securities
- and cluster the securities into any desired number of groups
- or try all number of groups and then select the number of groups that produced the highest quality results.

The question now becomes, have we succeeded in this endeavour and have we reached our goals? The answer is complex, and while there is no denying that we have herein presented a clustering methodology that can fulfil the above points, we need to look closer at whether or not we would actually want to use this methodology.

8.1 APPLICABILITY OF RESULTS

In this thesis we have presented a lot of information about every aspect needed to implement a piece of software that can take data for a set of securities and cluster them into groups that are as diverse as possible. On top of the basic algorithms we have defined improvements such as FAMES as well as multiple similarity measurements in the form of constants like country or industry as well as different correlation measurements.

It becomes extremely hard to present all of this in a unified way that will not take up another 20 pages. In the results of the previous chapter we have focused a little bit on clustering based only on different types of correlation, as this is the minimum required to see if the method produces any sort of sensible results.

However, it seems clear from these results that they are indeed applicable. Not only do we get groupings that look good on their own in terms of cluster quality as well as when visualizing the results. We can compare them to the existing Morningstar system and see that our basic PAM implementation yields almost identical results. On top of the basic PAM implementation we can add the adjustment algorithm to further increase our quality of clustering, and in the process efficiently identify outliers in the data.

This is all without considering any other similarity factors, which we have every freedom to add at any stage we want.

Without comparing the results to the Morningstar classification, the quality tables yield some interesting results on their own. On the data-set applied, which is US equity funds, the optimal number of groups consistently turns into 2. If we think about this, this aligns with our definition of efficiency. We will always achieve the highest spread between In-Group correlation to Other-Group correlation if we can evenly split the dataset in half. This holds true for any dataset that is homogeneous like the set of US equity funds. To get a different optimal number of groups, we need some fundamental difference in the underlying data, such as more separable or more spread out data.

A trend in the financial advisory industry is to recommend investors to invest in broad-market index funds such as the Wilshire 5000 or S&P 500, along with a broad-index bond fund. Essentially splitting your money into two categories. It stands to argue then, that our results show that if we want to maximize our diversification efficiency, then investing in two broad index-funds might not be a bad idea.

If we look close though, it is not the case that lower number of groups is always better. If we would for instance restrict ourselves to a minimum of 10 investments, it is clear that the optimal number of groups between 10 and up will not be very consistent. Just using the Pearson correlation data as example, the 5th most efficient clustering is using 12 groups, 25 groups is in rank number 8 and in fact if we limited our results to group numbers between 10 and 25 we would find that 12 groups is the optimal and 23 groups is the second-most optimal.

The strengths of this system compared to a static classification system such as Morningstars Style-box labels is clear. We can freely choose a number of groups and yield as-good or better quality than the labels or we can do a quantitive study of the data to see which number of groups would be suitable, possibly with restrictions on minimum and maximum numbers of groups.

8.1.1 Selecting investments

Something that has not been clearly defined is how to select investments from the resulting grouping in order to create an investment portfolio. Part of this is because investment selection is and probably always will be a very subjective matter. With this system we could automatically select investments based on highest Multi-Period Return, based on highest 3-year return, lowest volatility or any other measurement that you could create for a security. In other words, the user would simply pick with some pre-defined strategy of her own from each group.

There is a possible method of improving the diversification in the selection strategy that one could think of. The worst-case scenario when selecting investments from the different clusters is that we choose investments that are right on the border of the cluster towards each other, this problem is inherent in any classification system. We can theoretically pick securities that are right next to each other, and thus very similar, despite them being in different groups.

It would therefore be wise to try to create some sort of weighting function in the selection process that puts higher weight on securities towards the centre of a cluster if the cluster is surrounded, or higher towards an empty edge of the cluster if the cluster has no neighbours at that end. If one visualizes the cluster in 2D as we have done in this thesis, the weighting function would add a third dimension where the function would look like a blanked whose height over a security depends on the proximity of that security to a security in a different group.

Trying to implement such a function is unfortunately beyond the scope of this thesis. And in the end we will leave it up to the investment analyst to choose the correct investment for their portfolio, now that they have all the data in hand in an accessible format.

8.2 DIFFICULTIES

While the theory and implementation presents of us with a myriad of opportunities to cluster security data, there are also complications involved in this process. There are things that are hard to draw conclusions from as well as technical limitations to what you would want to present an end-customer with. There are difficulties in the theory such as finding the correct weighting function in a Multi-Period return for instance. There are also difficulties in the Multi-Period correlation in that we are calculating an average correlation of correlations that are with overlapping periods, so every correlation in the set from which we calculate the average are not independent and identically distributed. However, while the first problem is not something we can "solve", it is also a strength of the system that the user is allowed to put in their own subjective assessments. The latter problem with Multi-Period correlation can only be solved by not using it. Luckily, there is nothing in the clustering methodology that limits us to using Multi-Period correlation.

There are two main theoretical problems that needs to be discussed, whether or not we want to use the adjustment algorithm and which similarity measurement or groups of measurements are the best ones to use. There needs to be further research on these topics to get exact answers, but we can still try to learn from what we have seen in the results.

8.2.1 Clustering with or without adjustment

The purpose of the adjustment algorithm is to bring the clustering from an objective function that tries to minimize "the total dissimilarity of all objects to the medoid closest to the object", to an objective function that "tries to make the average similarity between an object and it's own group higher than the similarity to any other group".

What we really want to achieve when clustering securities is in fact the latter. We want items in the group to be as similar to each other as possible, not as similar to a medoid as possible. However, the exact effects of the latter (adjusted) situation is not as well-studied as the effects of the former when talking about clustering. The adjustment algorithm designed in this thesis has a rather high computational cost, and perhaps this could be further improved.

In the results, we see that invariably we will get higher quality clusters with the adjustment algorithm as opposed to without it. Considering that using PAM without adjustment produces near-identical results with Morningstar classification in terms of quality, it would be interesting to further study exactly what Morningstar is doing and if they are really optimizing their classification for the correct things.

An unexpected outcome of using the adjustment algorithm is that it will automatically detect outliers that are "too far" from the rest of the data and place those outliers in their own groups. The effect of this is that while we say that we want 10 groups to invest in, we might in fact only be receiving 5 real groups, where the other 5 are singleton groups or groups with very few members because they are so far away from the rest of the data. It is still important to consider these investments in any portfolio to be selected, but without large enough groups to provide choice in the investment, perhaps investors will not feel comfortable with such a result. On the other hand, this effect of being able to clearly distinguish outliers might be a highly desirable feature. For example, if the data set is looking at hedge funds and mutual funds, the big outliers will be the hedge funds that truly are delivering returns that have very low correlation to mutual funds - which is one of the main justifications that most hedge funds claim for their high fees. It is therefore always important to briefly study the results of the clustering when the adjustment algorithm is used. If we really want 10 groups and there are 5 outlier groups, we might have to increase the number of groups to 15 in order to achieve our goal, or the outlier groups are a valuable part of the result.

As a very interesting aside, something the author is unable to answer is why the adjustment algorithm works so well for Multi-Period correlation. It works decently for Pearson correlation but not nearly as well as for Multi-Period correlation. Furthermore, it is actually making the situation worse in a lot of cases for Kendall and Spearman correlation. Freely speculating, the problem with Kendall and Spearman could be that the resulting correlations are all so similar, thus the resulting clusters so tight, that just changing the group of one security will make hundreds of other securities positions worse.

During the calculation of the quality tables in the Results chapter, the adjustment algorithm iterations was capped at 10,000. Whenever the adjustment algorithm successfully finished and brought the InCorr <= ExCorr measurement down to 0, it did so in about 400 to 600 iterations. In every other case it either stopped because of cycle-detection or it hit the maximum amount of iterations. The cycle-detection system built-in is a 2-step loop un-roll, meaning that it is a fixed-point iteration where it runs a given function on a given input once, then once more on the output of that. If any of those two values are identical to the other or to the initial input, the algorithm has to stop or it would be cyclical and never end.

The reason the algorithm hit the maximum amount of iterations at any point is that there are much more complicated cyclical patterns involved than can be seen looking just 2 steps back. During the process of creating the quality tables one such situation was investigated, and it was the case that it moved around at least 7 securities before it got back to it's original state and started that cycle over again. Implementing a cycle-detection system of sufficient depth would be detrimental to the performance of the algorithm.

In the end, it seems clear that the adjustment algorithm does produce better clustering results, but there are still some issues to further investigate before the author would put this system into production.

8.2.2 Which similarity measurement to use

The question of which similarity measurement(s) to use is a very interesting but difficult question. Upon explaining Multi-Period correlation for this thesis examiner, he immediately identified the problem that the measurements are not i.i.d. But perhaps we could achieve the desired properties with some other measurement. One of the purposes of Multi-Period correlation is to smooth out the data such that there is less focus on the outliers. Two other measurements are proposed in the thesis, Kendall and Spearman rank correlations. However, the problem with these are that they seem to consistently understate the correlation when applied to real data, which makes the similarities smaller and the clusters tighter. Ideally we want to have little focus on outliers but still relatively spread-out clusters. A different measurement to use not covered in the report would be to say that we take the square root of all the values from which the correlation is calculated. In other words, using Pearson correlation on square-rooted values. Since the values are returns and returns can be negative, we would have to define $\sqrt{x} = \text{sign}(x) \cdot \sqrt{|x|}$. This would have the desired property of making the effect of significant outliers smaller, but would maintain a normal correlation measurement as opposed to a rank correlation measurement.

However, it should be up to the user of the system to choose which correlation measurement they want and to test for themselves which measurements they think yield the best results. One should keep in mind that the intention is that correlation is only one of the similarity measurements used, and that a whole host of other ones will be introduced such that the importance of the correlation is slightly downplayed.

8.2.3 Technical difficulties

Aside from implementation difficulties and optimizing the system, there are technical difficulties that are inherent of the algorithms. For instance the adjustment algorithm's complexity is linear in the number of securities, but the work required to calculate whether or not to place the security in another group is not insignificant.

Likewise, calculating the dissimilarity matrix will always be slow due to the fact that you need to calculate $N(N-1)/2$ dissimilarities where N is the number of securities. Currently the system calculates the dissimilarity matrix for Multi-Period correlation for 500 securities in about 47 seconds on the authors computer. Though it takes 182 seconds to do it for 1000 securities. Clearly, if this system is supposed to be fully automatic and dynamic and not use any pre-calculated values, we would need to either severely optimize (something that might or might not be possible) the code or we need to bite the bullet and store pre-computed values

if it is intended that the system be used regularly on more than 1000 securities. As seen in the clustering chapter earlier, the actual clustering can be performed quite quickly, and even for very large datasets if we use FAMES we can do the clustering in a matter of seconds. However, if the number of iterations on the adjustment algorithm is not severely restricted, that process will take quite a lot of time.

One proposed way to deal with this difficulty would be to pre-compute the correlations for all securities that we can get data on, cutting our computation time for the end-user to a reasonable number of seconds to a minute of waiting time to get their results. Considering the analytical power of the results it would provide to the user, waiting a minute or so for the results would not be a major issue.

8.3 POSSIBLE FUTURE EXPANSION

This thesis has been very much a practical exploration of the feasibility of using clustering as a methodology to classify securities. While the results are positive and hint at real possibilities of using this system reliably. There is much room for improvement, particularly in the research stage. To name only a few things one could continue studies on:

- Instead of using K-medoids, can we find a method that constructs groups such that we directly optimize our desired objective function, circumventing the need for the adjustment algorithm?
- Are there more or better quality measurements for a given clustering?
- What are the exact mathematical properties of Multi-Period returns and Multi-Period correlations? Are they mathematically sound to use or does the i.i.d problem render them useless and/or misleading?
- What dissimilarity measurements are best to use, and how do you weigh them? Is there an objective way or a way to apply a weighting so as to try to maximize diversity by some measurement?

On top of these possible future theoretical expansions, it would be very interesting to study the application of these algorithms to a set of more spread out data, for instance investigating global stocks or funds or include things such as commodities and currency exchanges as well.

Would for instance clustering equities, bonds and commodities produce three separable groups, or would it still be a relatively homogenous data-set?

8.4 CONCLUSION

We set out on this journey to find a security classification system that allowed far greater flexibility than the current classification systems in use today. We have presented a complete system to do this classification and shown that it has acceptable results compared to Morningstars Style-box categorization.

While there are complicated issues still unanswered, the author believes that the overall approach of calculating dissimilarities between securities and using K-medoids to cluster the

securities hold enormous potential. The resulting system is a powerful tool in the hand of an investment analyst or an investment advisor.

Ultimately, any classification system is just a tool for analysts to help them along the way. As such, it is the goal of the tool to empower the analyst and make the data available work for the analyst. The clustering methodology presented not only gives us well diversified groups with flexible parameters, it gives the analyst a way to incorporate any data they wish. It puts the use of the data at the fingertips of the analyst.

With traditional classification systems, you either work within the framework of the system or you should discard it entirely. With the proposed methodology of this thesis, the system works for the analyst, not the other way around.

Any conclusion at this stage would have to be subjective, but it is the authors conclusion that the methodology presented in this thesis, properly developed, would have the power to both replace or augment any other major classification system in use today.

BIBLIOGRAPHY

Hastie, Trevor, Robert Tibshirani, and Jerome Friedman

2011 *The Elements of Statistical Learning*, Springer Series in Statistics, Springer Science+Business Media, LLC.

Higham, Nicholas

2001 *Computing the Nearest Correlation Matrix*, http://eprints.ma.man.ac.uk/232/01/covered/MIMS_ep2006_70.pdf.

Kaplan, Paul D.

2003 *Holdings-Based and Returns-Based Style Models*, http://corporate.morningstar.com/cf/documents/MethodologyDocuments/ResearchPapers/Holdings-basedAndReturns-basedStyleModels_PK.pdf.

Kaufman, Leonard and Peter J. Rousseeuw

1990 *Finding Groups in Data - An Introduction to Cluster Analysis*, Wiley Series in Probability And Statistics, John Wiley & Sons, Inc.

Levy, Moshe and Ya'acov Ritov

2011 "Mean-variance efficient portfolios with many assets: 50% short," *Quantitative Finance*, Vol. 11, No. 10, pp. 1461–1471.

Markowitz, Harry

1952 "Portfolio Selection," *The Journal of Finance*, Vol. 7, pp. 77–91.

Marrison, Chris

2002 *The Fundamentals of Risk Measurement*, McGraw-Hill.

Morningstar

2011 *Morningstar Global Equity Classification Structure*, http://corporate.morningstar.com/us/documents/methodologydocuments/methodologypapers/equityclassmethodology.pdf.

2012 *Morningstar Data FAQs*, http://www.morningstar.com/Help/Data.html.

Paterlini, Adriano Arantes, Mario A. Nascimento, and Caetano Traina Junior

2011 "Using Pivots to Speed-Up k-Medoids Clustering," *Journal of Information and Data Management*, Vol. 2, pp. 221–236.

Rebonato, Riccardo and Peter Jackel

1999 *The most general methodology to create a valid correlation matrix for risk management and option pricing purposes*, http://www.quarchome.org/correlationmatrix.pdf.