Fall 2017

Modeling Volatility of Financial Time Series Using Arc Length

Benjamin H. Hoerlein

MODELING VOLATILITY OF FINANCIAL TIME SERIES USING ARC LENGTH

by

BENJAMIN HANS HOERLEIN

(Under the Direction of Tharanga Wickramarachchi)

ABSTRACT

This thesis explores how arc length can be modeled and used to measure the risk involved with a financial time series. Having arc length as a measure of volatility can help an investor in sorting which stocks are safer/riskier to invest in. A Gamma autoregressive model of order one(GAR(1)) is proposed to model arc length series. Kernel regression based bias correction is studied when model parameters are estimated using method of moment procedure. As an application, a model-based clustering involving thirty different stocks is presented using k-means++ and hierarchical clustering techniques.

INDEX WORDS: Time Series, Arc length, Gamma distribution, Clustering analysis

2009 Mathematics Subject Classification: Statistics

MODELING VOLATILITY OF FINANCIAL TIME SERIES USING ARC LENGTH

by

BENJAMIN HANS HOERLEIN

B.S., Florida Institute of Technology, 2016

A Thesis Submitted to the Graduate Faculty of Georgia Southern University in Partial

Fulfillment of the Requirements for the Degree

MASTER OF SCIENCE

STATESBORO, GEORGIA

MODELING VOLATILITY OF FINANCIAL TIME SERIES USING ARC LENGTH

by

BENJAMIN HANS HOERLEIN

Major Professor:	Tharanga Wickramarachchi
Committee:	Stephen Carden
	Arpita Chatterjee

Electronic Version Approved:
December 2017

DEDICATION

This thesis is dedicated to my family who has always supported me in my endeavors, as well as my fiance. Without the support of my family and loved ones, I would not be in the position I am in today. So, for that, I am very thankful and I dedicate this thesis to them.

ACKNOWLEDGMENTS

I would like to thank, Dr. Wickramarachchi, for all of the work and guidance he has put into my thesis as well as my development in understanding upper level statistics. Also, I would like to thank Dr. Carden and Dr. Chatterjee for kindly accepting to be apart of my Thesis Committee. Finally, I would like the thank the Mathematics/Statistics Department at Georgia Southern University for allowing me to grow my understanding of statistics by completing my Master's Degree here.

TABLE OF CONTENTS

LIST OF FIGURES

Figure Page

CHAPTER 1

INTRODUCTION

1.1 An Overview of Financial Time Series Analysis

In financial time series, a variety of models have been proposed to model time dependent data. The autoregressive moving average model(ARMA), posed by Peter Wittle[18] is one of the famous time series models. One main assumption in ARMA models is the constant variance. In financial time series, this assumption becomes unrealistic. The first big stride in measuring financial volatility would come from Fisher Black, Robert Merton, and Myron Scholes [1] in 1973 when they created the *Black-Scholes Model*. This model is used for modeling price variation over time of different financial instruments. The Black-Scholes model has a non-constant variance due to the geometric Brownian motion used in the model. The second stride for nonlinear modeling for time series came from Engle[6]. Engle won a Nobel prize in economics for his work developing the *autoregressive conditional heteroskedastic*(ARCH) time series model. The biggest stride in capturing non-constant variance came from Bollersley[3] who generalized the ARCH model and proposed the *generalized autoregressive conditional heteroskedastic* (GARCH) model. While the ARCH model assumes that the conditional variance depends only on past variance, the GARCH model assumes that conditional variance depends on both past variance and past observations.

As described by Wickramarachchi et al. [20]: Arc length is a natural measure of the fluctuations in a data series and can be used to quantify volatility. They proved the asymptotic distribution of sample arc length and used it to detect change points in terms of volatility. Wickramarachchi and Tunno [19], discusses how sample arc length can be used to classify financial time series into highly volatile and less volatile series. This method is a feature-based clustering technique. In this thesis, we try to model the arc length series of a financial time series and use a model-based technique to cluster them according to volatility(risk).

1.2 Prices vs. Log-Returns

It is a common practice that log returns of a financial time series is taken into consideration rather than raw prices. Let P_t be the raw prices of a discrete financial time series and is observed at equally spaced out times t. Mikosch and Stărică [16] have shown the usefulness of a log-transformation in the financial time series process. This log-transformation, called *Log-Returns* is mathematically defined as

$$x_t = \log P_t - \log P_{t-1} \tag{1.1}$$

where $t = 2, 3, \cdots$. Figure 1.1 shows time series plot of the Coca-Cola Company(KO) from December 2005 to March 2007, just before the recession hit. The plot on the left is the raw prices for KO and the plot on the right is the log-returns for KO. The log-return plot exposes that the Coca-Cola Company was most volatile around October 2006 and shows the non-constant volatility over time.

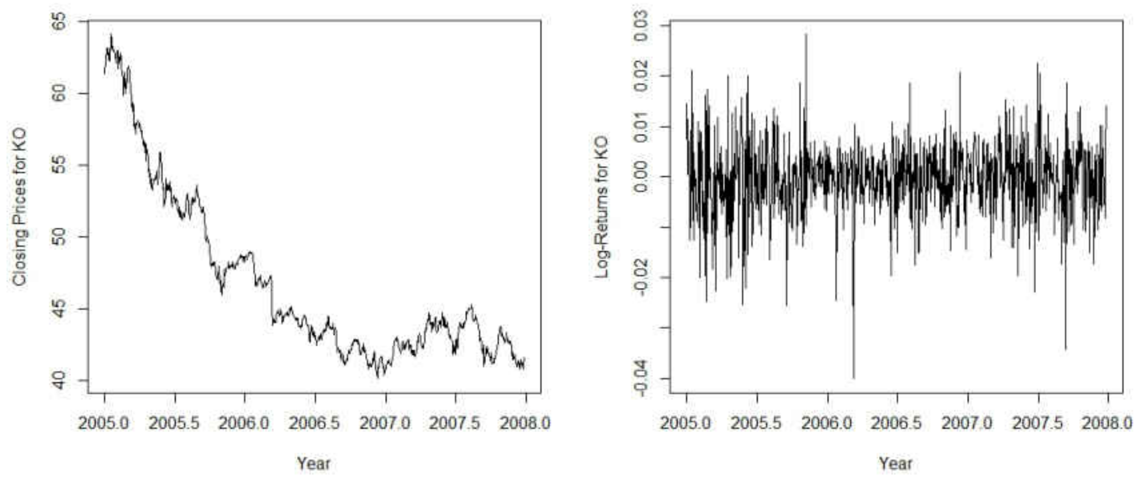

Figure 1.1: Time series plot for raw prices for KO(left) and time series plot of log-returns for KO(right)

Mikosch and Stărică [16] state the following three properties of log-returns of financial time series data that can not be observed in raw prices:

1. The distribution of log-returns is not normal, but rather with a high peak at zero and heavy-tailed. This comes from most days having little change, i.e. a “flat” day.

2. There is dependence in the tails of the distribution. These tails represent major changes in returns, which do not happen often, but tend to trend together.

3. The sample auto-covariance function(ACF) in the log return process is negligible at all lags, expect possibly the first lag.

To help understand some of the properties visually, lets consider the first property. Figure 1.2 is the empirical density of the log-return daily closing prices for the Coca-Cola Company. As one can see, this density has a very high peak with heavy tails. This follows what was stated in property one.

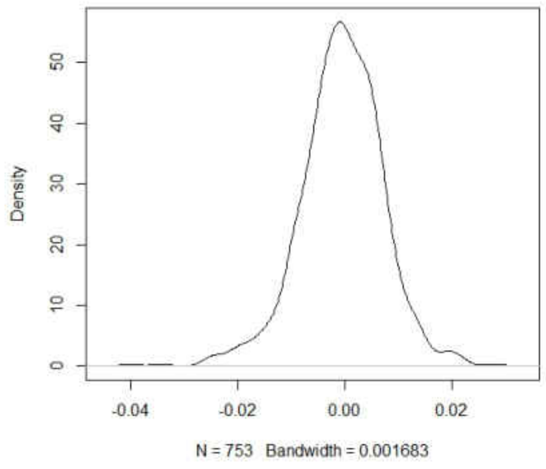

Figure 1.2: Kernel density plot of log-return daily closing prices of KO

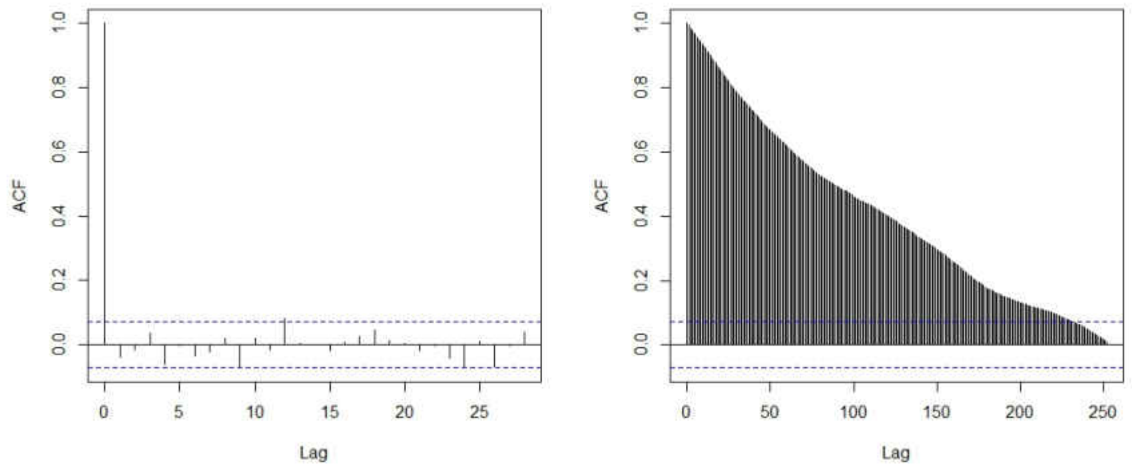

Figure 1.3: ACF graph for log returns prices for KO(left) and ACF graph for raw prices for KO(right)

The sample ACF of log-return prices for KO in figure 1.3 shows that all lags are negligible expect for the first lag. This supports the third claim of log-returns of financial time series data stated by Mikosch and Stărică [16].

1.3 Arc length

As described by Wickramarchchi et al. [20]: arc length is a natural measure of the fluctuations in a data series and can be used to quantify volatility. Also, Wickramarchchi et al, [20] proved that the difference between two sample arc lengths has an asymptotic normal distribution, see Corollary 1 from [20] for the detailed proof. The sample *arc length* is defined as follows: Let P_t be the raw prices observed at t = 2,3,$\cdots$,n. Then the sample arc length at lag 1 is defined as:

$$x_t = \sum_{t=2}^{n} \sqrt{1 + (\log P_t - \log P_{t-1})^2} \tag{1.2}$$

In this study, we focus on the series of arc length pieces of the log transformed series, $\sqrt{1 + (\log P_t - \log P_{t-1})^2}$. Figure 1.4 shows the calculated arc length returns for the Coca-Cola Company and for Google. As one can see by looking at the y-scale, the arc length values for KO(left) have a low variation through 2005 to 2007 compared to Google(right).

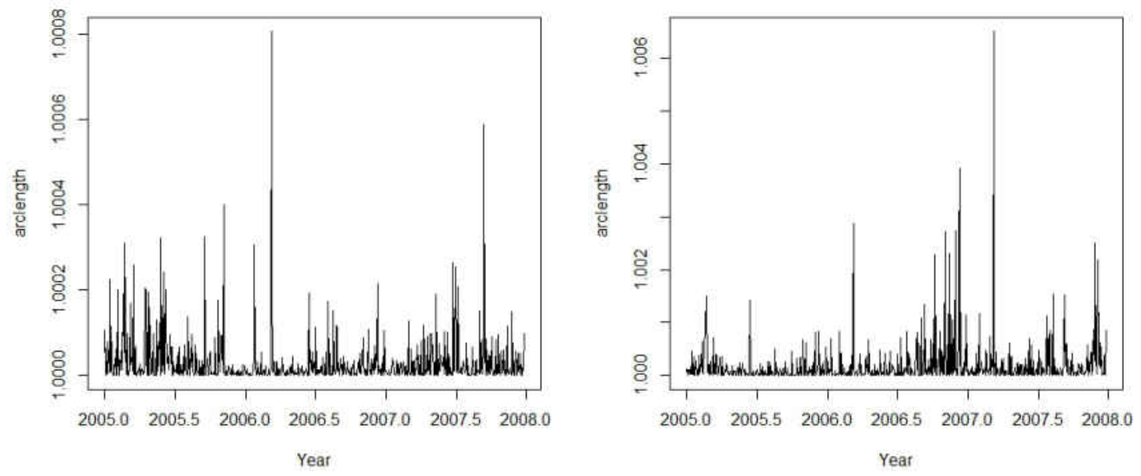

Figure 1.4: Time series plot of arc length for KO(left) and time series plot of arc length for GOOGL(right)

1.4 Motivation

In investment decision making, clustering stocks based on risk they carry can be very useful for reducing losses one may incur. The idea of clustering stocks based on arc length(as a measure of risk) was first proposed by Wickramarachchi et al[19]. This paper grouped financial time series into different clusters using sample arc length. Since this has been done observing one attribute of a series, it is called feature-based clustering.

In this thesis, we propose to fit an appropriate model to series of arc length pieces and apply a model based clustering technique to group time series based on the risk they carry. The rationale behind the idea is that a model for arc length series represents how its volatility changes over time. We expect it to represent a different aspect of time series in terms of volatility than the sample arc length.

This is how the thesis will proceed. In chapter 2, the properties of autoregressive processes are presented. In addition, method of moments estimation, kernel regression, and clustering techniques will be presented. Chapter 3 will contain results based on the methods and theory in chapter 2. Chapter 4 will contain the conclusion of this thesis as well as potential future studies.

CHAPTER 2

THEORY AND METHODS

2.1 Introduction

In this chapter, we will discuss the autoregression model for modeling arc length data. Furthermore, we present the shifted gamma distribution and discuss method of moments estimators of an AR(1) process that follows a shifted gamma distribution, proposed by Fernandez and Salas [7]. This section also discusses concepts behind kernel regression that is proposed as a bias correction technique for kurtosis estimator. Finally, a discussion on different types of clustering techniques is presented.

2.2 AR Models

First, we define an AR process of order p, since the AR(1) model has been used to describe the autocorrelation structure in arc length series. Note that white noise is defined as (WN).

Definition 2.1. *(Brockwell and Davis[4])*

The time series $\{X_t\}$ is an AR(p) process if it is stationary and satisfies (for every t)

$$X_t = \phi_1 X_{t-1} + \cdots + \phi_p X_{t-p} + Z_t \tag{2.1}$$

where $Z_t \sim WN(0, \sigma^2)$.

In this thesis, we will be focusing on the AR(1) process:

Definition 2.2. *(Brockwell and Davis[4])*

The time series $\{X_t\}$ is an AR(1) process if it is stationary and satisfies (for every t)

$$X_t = \phi X_{t-1} + Z_t \tag{2.2}$$

where $Z_t \sim WN(0,\sigma^2)$ *and* $|\phi| < 1$.

Note that $|\phi| < 1$ guarantees that AR(1) process has an unique stationary solution.

2.3 The Shifted Gamma Distribution

It is apparent that arc length sequences of a financial time series follows a heavily right skewed distribution. It is also shifted to the right since one step ahead arc length is bounded by 1 from the left. The empirical density plot for General Electric's(GE) arc length, shown below, confirms these attributes discussed above.

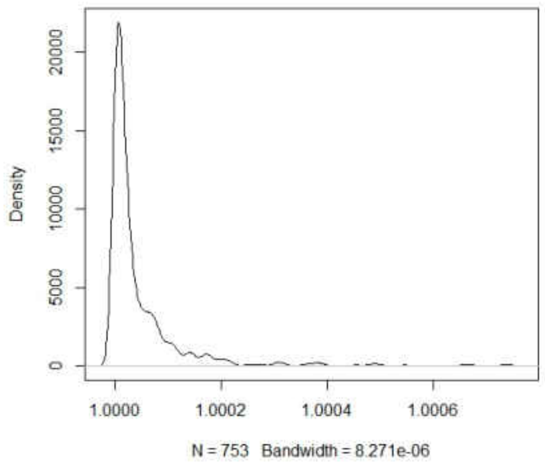

Figure 2.1: Arc length density plot for general electric company

As one can see, it is justifiable to suggest that the marginal distribution of arc length sequences follow a shifted gamma distribution. A randomly generated shifted gamma distribution with parameters ($\alpha = 1$, $\beta = 1$, $\lambda = 1$) was created in R using the rgamma3 function [15] and the empirical density is shown below being compared to the Arc length Density plot for GE.

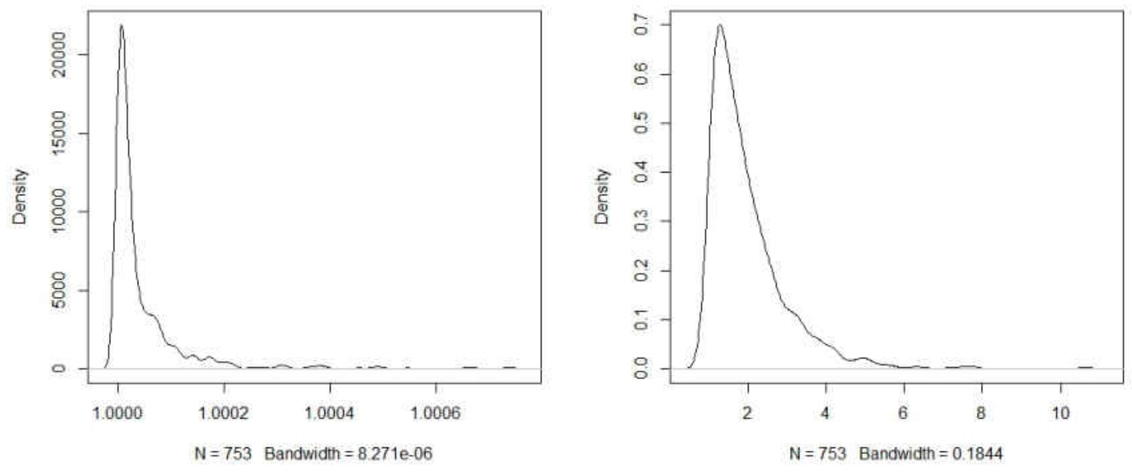

Figure 2.2: Arc length empirical density plot for general electric company(left) and a randomly generated density plot for a shifted gamma distribution(right)

The narrow peak in empirical density of GE above on the left in Figure 2.2 suggests an extremely large scale parameter.

Definition 2.3. *(Fernandez and Salas [7])*

Let X be a RV that follows a shifted gamma distribution, then the probability density function of X is given by:

$$f(x) = \frac{\alpha^{\beta}(x-\lambda)^{\beta-1}e^{-\alpha(x-\lambda)}}{\Gamma(\beta)} \tag{2.3}$$

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

2.4 Method of Moments Estimation

Method of moments and maximum likelihood estimation are two highly popular procedures discussed in inferential statistics. Method of moments estimation has been used in many studies [5] in the past due to its simplicity. In fact, Yule Walker estimation in time series setting is a method of moment estimation. The following is the definition used in Casella and Berger Statistical Inference textbook on page 312 for *method of moments* estimation [5]:

Let $X_1, \cdots, X_n$ be a sample from a population with pdf or pmf $f(x|\theta_1, \cdots, \theta_k)$. Method of moments estimators are found by equating the first k sample moments to the corresponding k population moments, and solving the resulting system of simultaneous equations. More precisely, define

$$
\begin{aligned}
m_1 &= \frac{1}{n}\sum_{i=1}^{n} X_i^1, & \acute{\mu}_1 &= EX^1 \\
m_2 &= \frac{1}{n}\sum_{i=1}^{n} X_i^2, & \acute{\mu}_2 &= EX^2 \\
& & &\vdots \\
m_k &= \frac{1}{n}\sum_{i=1}^{n} X_i^k, & \acute{\mu}_k &= EX^k
\end{aligned}
$$

The population moment $\acute{\mu}_j$ will typically be a function of $(\theta_1, \cdots, \theta_k)$, say $\acute{\mu}_j(\theta_1, \cdots, \theta_k)$. The method of moments estimator $(\bar{\theta}_1, \cdots, \bar{\theta}_k)$ of $(\theta_1, \cdots, \theta_k)$ is obtained by solving the following system of equations for $(\theta_1, \cdots, \theta_k)$ in terms of $(m_1, \cdots, m_k)$:

$$
\begin{aligned}
m_1 &= \acute{\mu}_1(\theta_1, \cdots, \theta_k) \\
m_2 &= \acute{\mu}_2(\theta_1, \cdots, \theta_k) \\
&\vdots \\
m_k &= \acute{\mu}_k(\theta_1, \cdots, \theta_k)
\end{aligned}
$$

As one can see, method of moments estimation can be a very quick method for estimating parameter of a given distribution. These estimators are often improved in order to reduce the bias in them. In this thesis, method of moments estimation will be implemented using techniques given by Fernandez and Salas [7].

Fernandez and Salas [7] published a paper in 1990 that analyzed streamflow data over time using an AR(1) process with gamma marginals. In this paper, a first-order gamma-autoregressive(GAR(1)) model, as given below, was used:

$$X_t = \phi X_{t-1} + \varepsilon_t. \tag{2.4}$$

where X_t has a shifted gamma distribution, ϕ is the auto-regression coefficient, and ε_t is an independent variable.

The stationary linear GAR(1) process from Equation 2.4, has four parameters that need to be estimated: α, β, λ, and ϕ. The following equations of mean(μ), variance(σ^2), skewness coefficient(γ), and lag one autocorrelation(ϕ) are used to derive method of moment estimators [7] of desired parameters.

$$\mu = \lambda + \frac{\beta}{\alpha}, \tag{2.5}$$

$$\sigma^2 = \frac{\beta}{\alpha^2}, \tag{2.6}$$

$$\gamma = \frac{2}{\sqrt{\beta}}, \tag{2.7}$$

$$\phi = \rho_1. \tag{2.8}$$

Equations 2.5 through 2.8 can be estimated based on a sample $X_1, X_2, \cdots, X_N$, using the following well known relationships:

$$m = \frac{1}{N}\sum_{i=1}^{N} X_i, \tag{2.9}$$

$$s^2 = \frac{1}{N-1}\sum_{i=1}^{N}(X_i - m)^2, \tag{2.10}$$

$$g_1 = \frac{N}{(N-1)(N-2)s^3}\sum_{i=1}^{N}(X_i - m)^3, \tag{2.11}$$

$$r_1 = \frac{1}{(N-1)s^2}\sum_{i=1}^{N}(X_i - m)(X_{i+1} - m). \tag{2.12}$$

where m is the estimator of μ, s^2 is the estimator for σ^2, g_1 is the estimator for γ, r_1 is the estimator of ρ_1, and N is the sample size. Unfortunately, the estimator of σ^2, γ, and ϕ are biased. But Fernandez and Salas [7] have suggested a bias correction for ϕ and σ^2. These bias corrections are presented in Equations 2.13 through 2.15. A bias correction must be done before using m, s^2, g_1, and r_1 into the Equations 2.5 through 2.8 to estimate the GAR(1) model parameters. The following equations shows the bias corrected estimator $\hat{\rho}_1$ and $\hat{\sigma}^2$:

$$\hat{\rho}_1 = \frac{r_1 N + 1}{N-4}, \tag{2.13}$$

where r_1 was calculated from equation (2.12)

$$\hat{\sigma}^2 = \frac{N-1}{N-K}s^2, \tag{2.14}$$

where

$$K = \frac{[N(1-\hat{\rho}_1) - 2\hat{\rho}_1(1-\hat{\rho}_1^N)]}{[N(1-\hat{\rho}_1)^2]}. \tag{2.15}$$

Fernandez and Salas [7] has suggested a bias correction for $\hat{\gamma}$ given in Equation 2.11. The method Fernandez and Salas proposed does not work in our particular problem, and will be discussed in the next section.

2.5 Fernandez and Salas Suggested bias correction for $\hat{\gamma}$

It has been noticed that the estimation of γ, given in Equation 2.11, yields a negative bias [7]. In 1974, Wallis et al [17] used a simulation study and suggested a bias correction for *iid* random variables of gamma, lognormal, and weibull distributions. Then, Bobee and Robiataille [2] proposed a correction factor to adjust the estimator of γ. The corresponding bias corrected sample skewness coefficient(g) is defined as:

$$g = \frac{\frac{1}{N}\sum_{i=1}^{N}(X_i - m)^3}{(\frac{1}{N}\sum_{i=1}^{N}(X_i - m)^2)^{3/2}}. \tag{2.16}$$

As described in Fernandez and Salas [7], Kirby [10], showed that Equation 2.16 is bounded by $\pm L$ given as:

$$L = \frac{(N-2)}{\sqrt{N-1}}. \tag{2.17}$$

Similarly, $\pm\sqrt{N}$ equals the bounds of g_1 given in Equation 2.11. Bobee and Robitatille(1975) [2] gave a bias corrected estimator of γ for *iid* gamma variables with $0.25 \leq \gamma \leq 5$ and $20 \leq N \leq 90$, as follows:

$$\hat{\gamma}_0 = \frac{Lg_1[A + B(\frac{L^2}{N})g_1^2]}{\sqrt{N}}, \tag{2.18}$$

where A and B are defined as:

$$A = 1 + 6.51N^{-1} + 20.2N^{-2}, \tag{2.19}$$

$$B = 1.48N^{-1} + 6.77N^{-2}. \tag{2.20}$$

None of the stated Equations in 2.17 through 2.20 can be used in time series setting because of the independence assumption. Fernandez and Salas conducted a Monte Carlo simulation study, discussed later in the chapter, for GAR(1) model in equation (2.4) with the goal of introducing a bias correction for the skewness coefficient. The following are the summarized results from the study:

1. The bias of the estimator for γ decreases as N increases.
2. As L increases, ρ_1 increases as well.
3. If $\rho_1 = 0$, then the estimated skewness values are very similar to Equation 2.18.
4. When the values are not linearly correlated, then equation 2.18 does not hold.
5. Results from the Monto Carlo Simulation indicate that the only factors that are affecting the skewness are the auto-correlation coefficient and the sample size

From these notes above. Fernandez and Salas [7] proposed the following unbiased estimator of γ:

$$\hat{\gamma} = \frac{\gamma_0}{f} \tag{2.21}$$

where f is an additional factor to be used when the values are auto-correlated when $0.25 \leq \gamma \leq 2$ and $0 \leq \rho_1 \leq 0.8$. Using a non-linear least-squares equation scheme the coefficients of the following regression equation were obtained:

$$f = (1 - 3.12p_1^{3.7}N^{-0.49}) \tag{2.22}$$

A few big takeways from this Monto Carlo simulation study and bias correction for γ are the followings:

1. Bobee and Robiataille [2] proposed bias correction can not be applied for correlated data.
2. Fernandez and Salas [7] bias correction is not appropriate in our application due to higher kurtosis and large sample size.

As a result, a different method needs to be proposed in order to estimate the skewness with a large sample size. Before we discuss a different estimation technique, we will present how the innovation sequence of GAR(1) can be generated. This idea has been used in subsequent simulation studies.

Fernandez and Salas [7] gave two alternative approaches to calculate ε. The first approach was found by Gaver and Lewis [8] and will be used for integer values of β of equation 2.4 is given by:

$$\varepsilon = \frac{\lambda(1-\phi)}{\beta} + \sum_{j=1}^{\beta} \eta_j. \tag{2.23}$$

where $\eta_j = 0$, with probability ϕ. $\eta_j = exp(\alpha)$, with probability $(1-\phi)$ and $exp(\alpha)$ is an exponentially distributed random variable with the mean equal to $\frac{1}{\alpha}$. The second approach, proposed by Lawrance [11], found a solution for non-integer β values. In this case, ε can be obtained by the following:

$$\varepsilon = \lambda(1-\phi) + \eta. \tag{2.24}$$

where $\eta = 0$ if $M = 0$, or $\eta = \sum_{j=1}^{M} Y_j(\phi)^{U_j}$ if $M > 0$,
where M is an integer random variable that follows a Poisson distribution with mean value equal to $-\beta ln(\phi)$. The set U_j is an *iid* random variable with an uniform distribution in (0,1). Also, Y_j is an *iid* random variable with an exponential distribution with mean $(\frac{1}{\alpha})$.

The next section will introduce the non-parametric regression technique of kernel regression.

2.6 Kernel Regression

Kernel regression is a non-parametric regression technique. This method is different from linear and polynomial regression since it does not assume a particular form for the conditional expectation.

Let $E(Y|X = x_i) = g(x_i)$ and

$$y_i = g(x_i) + \varepsilon_i$$

where ε is the random error that has mean zero. Then the predicted y is given by $\hat{y} = \hat{g}(x)$. In simple linear regression the function g takes the form of $g(X) = \beta_0 + \beta_1 X$. But in kernel regression, as stated above, we do not assume any particular form.

A reasonable approximation to the regression curve $g(x_0)$ will be the weighted mean of response variable near a point x_0. This local averaging procedure can be defined as

$$\hat{g}(x_0) = \sum_{i=1}^{N} w_{N,h,i}(x_0) y_i \tag{2.25}$$

The average will smooth the data. The weights depend on the value of x_0 and on h, where h is called the bandwidth or the smoothing parameter. As h gets smaller, (that is, the size of the neighborhood gets smaller) $\hat{g}(x_0)$ becomes less biased but also has a greater variance.

In kernel regression, weights are constructed from a kernel function. A kernel function, $K(z)$ satisfies following conditions.

1. $K(z) > 0$
2. $\int K(z)dz = 1$
3. $K(z) = K(-z)$
4. $\int z^2 K(z)dz < \infty$

Then weights are defines as

$$w_{N,h,i}(x_0) = \frac{K\left(\frac{x_i - x_0}{h}\right)}{\sum_{i=i}^{N} K\left(\frac{x_i - x_0}{h}\right)}.$$

The bandwidth h determines the degree of smoothing. As mentioned above, a large h increases the width of the bins, increasing the smoothness of $\hat{g}(x_0)$. The famous $Nadaraya-$

Watson kernel weighted average estimator is given by,

$$\hat{g}(x_0) = \frac{\sum_{i=i}^{N} K\left(\frac{x_i - x_0}{h}\right) y_i}{\sum_{i=i}^{N} K\left(\frac{x_i - x_0}{h}\right)}.$$

There are number of commonly used kernel functions.

1. Box kernel: $K(z) = I_{|z|<1/2}$
2. Triangle kernel: $K(z) = \left(1 - \frac{|z|}{c}\right) I_{|z|<\frac{1}{c}}$
3. Gaussian kernel: $K(z) = \frac{\exp\left(\frac{-z^2}{2}\right)}{\sqrt{2\pi}}$
4. Epanechnikov kernel $K(z) = \left(\frac{3}{4}\right)\left(1 - z^2\right) I_{|z|<1}$

It is essential to choose an optimal value for the bandwidth. We typically use cross validation to pick the optimal h^*. That is, we find the value of h such that

$$\underset{h}{\text{minimize}}\ CV(h) = \frac{1}{n}\sum_{i=1}^{n}\left(y_i - \hat{g}_{-i,h}(x_i)\right)^2$$

where $\hat{g}_{-i,h}(x_i)$ is the estimated y_i leaving out the point (x_i, y_i).

We use the *Nadaraya – Watson* estimator with the Gaussian kernel in our study.

2.7 Clustering Algorithms

There are many different techniques to cluster data in general. A few of these clustering algorithms are explained in this section. To find more details about clustering algorithms, refer to Liao [12].

2.7.1 K-means and K-means++

K-means clustering algorithm is suitable when you have unlabeled data and you find groups for your data. The number of groups will be represented by the variable K. Each centroid of a cluster is based on different features which will define the resulting group. Examining the feature weight of each centriod can show someone what kind of group each cluster represents. The pseudo-code for the K-Means algorithm was first created by MacQueen [13] and goes as follow:

1. Choose the number of clusters k for your set S.
2. Randomly partition S into k clusters and determine their centers (averages) or directly generate k random points as cluster centers.
3. Assign each member from S to the nearest cluster, using some pre-chosen norm.
4. Recompute the new cluster centers.
5. Repeat steps 3 and 4 until clusters stabilize.

The K-means++ clustering algorithm was proposed by Ostrovsky et al [14]. The K-means++ algorithm is an improvement from the k-means algorithm since the K-means++ clustering algorithm is more carefully selecting the initial centers. This will create a more optimal method of clustering. The following is the algorithm of K-means++ clustering:

1. Choose one center uniformly at random from the data points.
2. For each data point x, compute the distance D(x) between x and the nearest center that has already been chosen.
3. Add one new data point at random as a new center, using a weighted probability distribution where point x is chosen with probability proportional to $(D(x))^2$.
4. Repeat Steps 2 and 3 until k distinct centers have been chosen.

2.7.2 Hierarchical Clustering Algorithms

Hierarchical Clustering Algorithms work by grouping data on a one-by-one basis of the nearest distance measure between each data points. For this thesis, the Euclidean distance is used as the measure of distance, the definition follows:

Definition 2.4. *For two points $\vec{x} \in \mathbb{R}^n$ and $\vec{y} \in \mathbb{R}^n$ the euclidean distance (d) is:*

$$d = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2} \tag{2.26}$$

Given a set of N items to be clustered, and an $N * N$ distance matrix, the basic process of hierarchical clustering defined by Johnson [9] is as follows:

1. Start by assigning each item to a cluster, so that if you have N items, you now have N clusters, each containing just one item. Let the distances between the clusters be the same as the distances between the items they contain.

2. Find the closest pair of clusters and merge them into a single cluster, so that now you one cluster less.

3. Compute distances between the new cluster and each of the old clusters.

4. Repeat steps 2 and 3 until all items are clustered into a single cluster of size N.

Step 3 of this algorithm can be completed in different ways, which would be the *single-linkage*, *complete-linkage*, and *average-linkage*.

2.7.3 Clustering Linkage Functions

We will briefly introduce each clustering linkage functions. First, the single-linkage algorithm will be presented. For *single-linkage* clustering, which can also be referred as the *minimum method*, considers the shortest distances between any members of two clusters while calculating the distance. *Complete-linkage* clustering, which is also called the *maximum method*, considers the maximum distances between any members of two clusters while calculating the distance. *Average-linkage* clustering considers the average distances between any members of two clusters while calculating the distance. The following is the *single-linkage* clustering algorithm that was introduced by Johnson [9]:

Let $D = [d(i,j)]$ be an $N * N$ proximity matrix. The clusterings are assigned sequence numbers $0, 1, \cdots, (n-1)$ and L(k) is the level of the kth clustering. A cluster with sequence number m is denoted (m) and the proximity between clusters (r) and (s) is denoted $d[(r),(s)]$. The single-linkage clustering algorithm follows these steps:

1. Begin with the disjoint clustering having level $L(0) = 0$ and sequence number $m = 0$.

2. Find the least dissimilar pair of clusters in the current clustering, say pair (r), (s), according to $d[(r),(s)] = \min(d[(i),(j)])$ where the minimum is over all pairs of clusters in the current clustering.

3. Increment the sequence number : $m = m + 1$. Merge clusters (r) and (s) into a single cluster to form the next clustering m. Set the level of this clustering to $L(m) = d[(r),(s)]$.

4. Update the proximity matrix, D, by deleting the rows and columns corresponding to clusters (r) and (s) and adding a row and column corresponding to the newly formed cluster. The proximity between the new cluster, denoted (r,s) and old cluster (k) is defined: $d[(k),(r,s)] = \min(d[(k),(r)], d[(k),(s)])$.

5. If all objects are in one cluster, stop. Else, go to step 2.

Average-linkage and *complete-linkage* have similar algorithms to the single-linkage, expect one would consider the maximum or average instead of the minimum for all the pairs in each cluster.

CHAPTER 3
RESULTS

Chapter 3 of the thesis presents the results of the research question. We first present the kernel regression based bias correction technique for the skewness parameter and its merits compared to techniques proposed in literature. Proceeding sub-sections discuss model based clustering proposed on arc length series of stocks found in the Dow Jones Index (DWJ).

3.1 Bias Correction of Kurtosis Estimator

For a fixed sample size, simulation can be used to bias correct using kernel regression. For each value of λ, α, β, and ϕ in the range of values given below, we simulate 100 samples of size 753 each from different shifted gamma distribution that has a lag-one autocorrelation structure. The fixed sample size has been chosen as 753 since it represents three years worth of stock returns. We choose values of each parameter from a limited range so that it works for our application. We were unable to cover the complete range, since the running time of the simulation part was too long. These bounds are chosen based on a preliminary study carried out on those thirty stocks without considering any bias correction. Following this argument, we fix the value of λ at 1.000164, which is the average of sample means of thirty arc length series from DWJ index. The value of λ can be easily picked from a range of values in the future when the method is expanded. Other parameters are chosen as follows:

$10 \leq \alpha \leq 10^8$ with increments of 10^n(where n has the values of $1 \leq n \leq 8$ with increments of 1), $0.004 \leq \beta \leq 1$ with increments of 0.002, (this guarantees that the kurtosis parameter falls between 2 and 31.63), and $0 \leq \phi \leq 0.21$ with increments of 0.01.

Based on these chosen values we generate 100 samples of size 753 from each of $21 \times 499 \times 8 = 83,832$ shifted gamma distributions with AR(1) autocorrelation structure. For each generated sample its sample kurtosis using Equation 2.11 and bias corrected sample autocorrelation at lag-one are computed. Then, for each combination of parameter values, 100 values of $\bar{g_1}$ and $\hat{\phi}$ are averaged. At the same time, the true kurtosis has been computed using Equation 2.7. This simulation creates a database of $\gamma, \hat{g_1}$, and $\bar{\hat{\phi}}$. These data are used to fit a kernel regression model for true γ against $\hat{\bar{g_1}}$ and $\bar{\hat{\phi}}$ using Gaussian kernel. In this process we use the Nadaraya-Watson estimator discussed in section 2.6. The optimal bandwidth of the corresponding model is 0.000218 for $\bar{\hat{\phi}}$ and 0.0788 for$\hat{\bar{g_1}}$. Note that we considered a burn period of the first five observations in the random generation. The pseudocode corresponding to the data generation is provided in Algorithm 1 and Algorithm 2.

```
input  : $n$, $\phi$, $\lambda$, $\alpha$, $\beta$
output: $\bar{\hat{\phi}}$,$\bar{\hat{g}}_1$,$\gamma$
M = randomly generated poisson distribution with parameters of N and
  $\lambda = -\beta log(\phi)$;
for k=1 to 100 do
    x[0] = randomly generated shifted gamma distribution with parameters of
      $n = 1, \alpha, \beta$, and $\lambda$;
    $\varepsilon = 0$;
    for i=1 to N do
        Simulate Data
        if  M[i] != 0 then
            $Y_j$ = randomly generated exponential distribution with parameters M[i]
              and rate = $\alpha$;
            $U_j$ = randomly generated uniform distribution with parameters M[i], 0,
              and 1;
            $\eta = \sum_{i=1}^{N} Y_j^{U_j}$;
        end
        $else(\eta = 0)$;
        $\varepsilon[i] = \lambda(1-\phi)+\eta$;
    end
    for i=2 to N do
        $x[i] = \phi x[i-1]+\varepsilon[i]$;
    end
    $x = x[6:N]$;
    $\hat{\phi}[k] = acf(x)$ with lag 1;
    $\hat{\phi}[k] = \frac{(\hat{\phi}[k])(n+1)}{(n-4)}$;
    $\hat{g_1} = \frac{n}{(n-1)(n-2)(var(x)^{3/2}) \sum_{k=1}^{753}(x-\bar{x})^3}$;
end
```

Algorithm 1: Psuedocode for Simulation Study

```
Parameter Estimation
$\bar{\hat{\phi}} = mean(\hat{\phi})$;
$\bar{\hat{g_1}} = mean(\hat{g_1})$;
$\gamma = \frac{2}{\sqrt{\beta}}$;
```

Algorithm 2: Psuedocode for Parameter Estimation

The fitted kernel regression model is presented in figure 3.1. Raw data are not shown on the same plot since it becomes too messy. The fitted value of γ is taken as the new bias corrected kurtosis estimate denoted by $\hat{\gamma}_{bc}$ of each combination of $\bar{\hat{g_1}}$ and $\bar{\hat{\phi}}$.

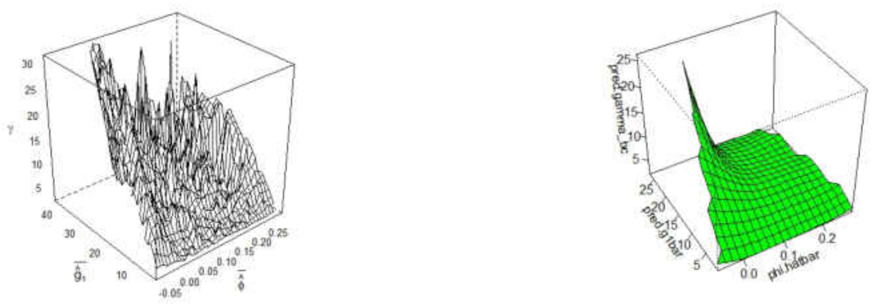

Figure 3.1: Simulation study estimated values(left) of $\bar{\hat{\phi}}, \bar{\hat{g}}_1$, and γ, kernel regression of the estimated values(right) of $\bar{\hat{\phi}}, \bar{\hat{g}}_1$, and $\hat{\gamma}_{bc}$)

We carry out a second kernel regression analysis to see how well this idea works. In this study, we partition our 83,832 data values into a training data set, consisting of 99% of the data, and a validation data set, consisting of 1% of the data.

The following is the algorithm that was implemented in R to validate the data:

1. Create a data frame that contains $\bar{\hat{\phi}}, \bar{\hat{g}}_1$, and γ as vectors.

2. Create the training data set where 99% of the data frame created in step 1 was chosen.

3. Create two vectors containing the training set and the validation set(1% of data).

4. Run the kernel regression function and then observe the results.

Figure 3.2 provides the lattice plot and the kernel regression of the validation data set. Note that the optimal bandwidth from the training data set is used for predictions.

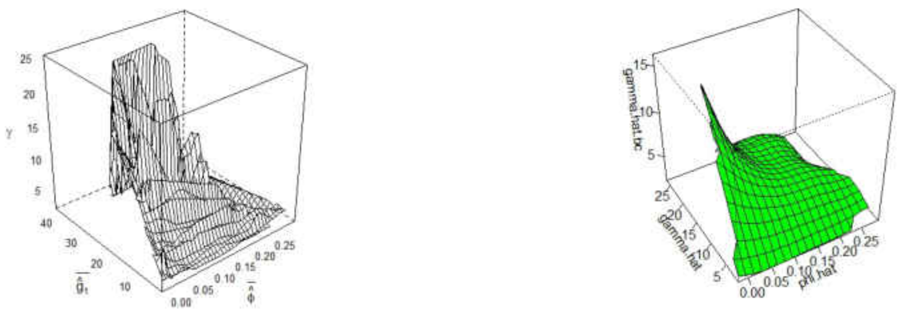

Figure 3.2: Lattice plot of validation data(left) and bias corrected for the validation data set(right)

By looking at two plots, we can see that having a pool of data and using a kernel regression model for prediction can be used in practice.

We use the residual plots of the validation data set under the proposed method and the method Fernandez and Salas [7] proposed to graphically study how well each method performs. Note that, we assume that one blindly use the latter method, without considering its limitations. Figure 3.3 presents residual plots under both methods. It is evident that the bias correction proposed in literature tends to over predict the kurtosis parameter compared to the method we propose in the study. But it is important to point out that none of the methods are perfect.

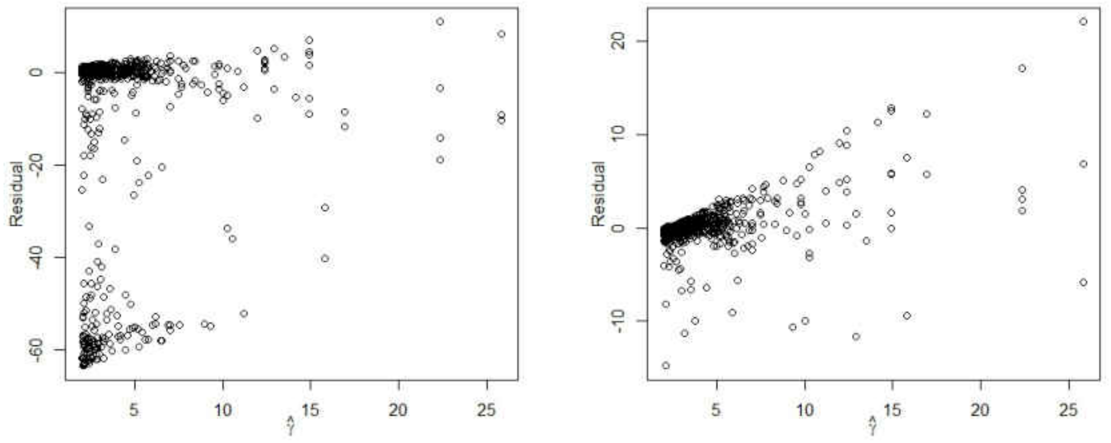

Figure 3.3: Residual plot for bias correction from literature(left) and residual plot for bias correction using kernel regression(right)

Both plots have a fanning pattern from the smallest residuals to the maximum residuals. The following are a few key observations from Figure 3.3:

1. The range of the residuals clearly favors our proposes bias correction.
2. The bias correction generated by kernel regression has a more constant funnel compared to the Fernenadez and Salas [7] proposed bias correction.
3. We tried different transformations since the plot shows an increasing variance. But none of them worked well. There is some room for improvements.

From the notes above, it is clear that the kernel regression method works better compared to the bias correction offered by the Fernandez and Salas [7].

Furthermore, we computed the mean sum of squares of errors (MSSE) under method proposed by Fernandez and Salas [7] and kernel regression method and found out those to be 512.54 and 5.09, respectively. As a results, we can clearly see that the MSSE under our method is significantly smaller compared to the method we found in literature.

3.2 An Application

As mentioned before, we fit GAR(1) models for arc length series of thirty stocks in DWJ index. We used method of moments estimation with kernel regression based bias correction for kurtosis parameter when parameters are estimated. Simulated pool of data and the optimal bandwidth obtained using kernel regression are used to bias correct kurtosis estimates. Bias corrected parameter estimates of the thirty stocks are given in Figure 3.4:

Stock	$\hat{\alpha}$	$\hat{\beta}$	$\hat{\phi}$	$\hat{\lambda}$	$\hat{\gamma}_{bc}$
KO	$2.32E^7$	0.0826	0.1050	1.00003	6.961
JNJ	$2.65E^7$	0.1268	0.0537	1.00003	5.617
PG	$6.74E^6$	0.0557	0.1669	1.00004	8.472
GE	$1.86E^7$	0.1339	0.1141	1.00005	5.466
WMT	$4.43E^6$	0.0747	0.0551	1.00006	7.318
UTX	$8.75E^1$	0.0053	−0.0003	1.00029	27.43
IBM	$2.39E^7$	0.7828	0.0908	1.00006	2.261
T	$5.04E^6$	0.0811	0.1152	1.00007	7.021
DD	$2.24E^6$	0.0543	0.0253	1.00007	8.584
DIS	$4.58E^6$	0.1051	0.0739	1.00007	6.168
MSFT	$5.69E^4$	0.0060	0.0111	1.00008	25.820
BP	$5.96E^6$	0.0944	0.0170	1.00008	6.511
JPM	$4.44E^6$	0.1553	0.1099	1.00008	5.075
C	$2.32E^6$	0.1392	0.1	1.00008	5.361
MCD	$6.75E^6$	0.1719	0.1656	1.00008	4.824
HON	$9.83E^6$	0.1983	0.0364	1.00008	4.492
BA	$2.13E^7$	0.6769	0.0968	1.00009	2.431
AXP	$1.00E^5$	0.0120	0.0301	1.00010	18.257
HD	$6.25E^6$	0.1945	0.1140	1.00010	4.535
IP	$1.18E^7$	0.2664	0.0458	1.00010	3.875
CVX	$1.88E^7$	0.3512	0.0411	1.00010	3.375
XOM	$1.09E^7$	0.1339	0.1141	1.00010	3.776
MRK	$8.07E^4$	0.0140	0.0438	1.00010	16.901
INTC	$1.64E^5$	0.0186	−0.0038	1.00012	14.683
HPQ	$1.21E^5$	0.0180	0.0015	1.00012	14.907
CAT	$9.65E^1$	0.0054	−0.0009	1.00035	27.255
AA	$1.98E^6$	0.1725	0.0414	1.00135	4.816
EKDAQ	$5.60E^4$	0.0160	0.0345	1.00017	15.811
GOOGL	$3.52E^5$	0.0656	0.1070	1.00018	7.806
AAPL	$9.01E^1$	0.0054	−0.0020	1.00051	27.223

Figure 3.4: Unbiased parameter estimates for thirty stocks

Note that there are four stocks (UTX, INTC, CAT, AAPL) that indicate a negative first order auto-correlation. No literature is available about bias correction of kurtosis parameter when negative autocorrelation is present. Furthermore, no research has been carried out about generating gamma marginal AR(1) processes with negative autocorrelation. It sill remains as an open question. As a result, we were unable to carry out a bias correction for kurtosis in that case. Thus the $\hat{\gamma}_{bc}$ listed in Figure 3.4 for those corresponding stocks UTX, INTC, CAT, and AAPL are the kurtosis parameter estimates without a bias correction. The matter will be discussed more in the conclusion.

3.3 Cluster Analysis

In this thesis, Model based clustering is applied to identify stocks with similar volatility. The Euclidean distance between models, i.e, the distance between parameter estimates, is computed to measure the similarity between two stocks. The Figure 3.5 shows the dendogram created based on the complete linkage function(left) and a cluster plot based on k-means clustering(right) when $k = 5$. The dendogram appears to be the same under all other linkage methods.

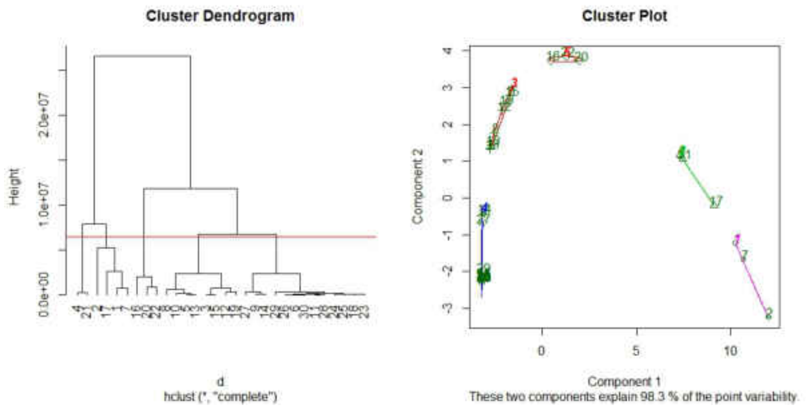

Figure 3.5: The hierarchical dendogram for the complete linkage(left) and a cluster plot based on k-means clustering(right)

The horizontal line drew above creates five cluster of stocks for the dendogram. On the right plot, we can see where each stock is clustered when we chose our $k = 5$ for k-means clustering. Figure 3.6 shows where each stock were grouped according to hierarchical clustering and k-means clustering. Depending on the position one places the partition line, the number of clusters you get might change. But having a higher number of clusters may not be practically important. As one can see from Figure 3.6, hierarchical clustering and k-means clustering grouped each stock in the same cluster, with the exception of stock 17(BA).

Here are a few notes to take away from Figure 3.6:

1. Notice that the stocks(UTX, CAT, AAPL, and INTC) who had negative correlation coefficient are grouped in cluster 5.

2. The majority of big tech companies such as Apple, Google, Untied Technologies, Microsoft,..etc are grouped in cluster 5.

Cluster Number	**Stocks**
Cluster 1	GE CVX
Cluster 2	KO JNJ BA
Cluster 3	IP XOM HON
Cluster 4	PG WMT T DIS BP JPM MCD HD
Cluster 5	UTX DD MSFT C AXP MRK INTC HPQ CAT AA EKDAQ GOOGL AAPL

Cluster Number	**Stocks**
Cluster 1	GE CVX BA
Cluster 2	KO JNJ
Cluster 3	IP XOM HON
Cluster 4	PG WMT T DIS BP JPM MCD HD
Cluster 5	UTX DD MSFT C AXP MRK INTC HPQ CAT AA EKDAQ GOOGL AAPL

Figure 3.6: Cluster placement for each stock for hierarchical clustering(top) and k-means clustering(bottom).

3.4 Feature-Based Clustering vs Model-Based Clustering

The paper published by Wickramarachchi et al [19] in 2015 used the exact same thirty stocks as an application. The paper [19] used the sample arc lengths as a feature of volatility and carried out a feature-based clustering of stocks in terms of volatility. K-mean++ algorithm with three cluster has been used in the application. Feature-based clustering reduces large amount of data into one feature that carries an attribute of a series and takes it into account when clustering is performed. One draw back of this method is that it fails to take any correlation structure or the behavior of data when they are grouped. This method fails to take any correlation structure or the behavior of data over time, when they are grouped.

Figure 3.8 compares results of the current study to the results of Wickramarchchi et al [19]. The figure shows how each stock has been grouped under each respective method.

Stock	**Company Name**	**K − means + + Model − Based**	**K − means + + Feature − Based**
KO	Coca − Cola	1	1
JNJ	Johnson & Johnson	1	1
PG	The Procter and Gamble	2	1
GE	General Electric	1	1
WMT	Wal − mart	2	1
UTX	United Technologies	3	1
IBM	IBM	1	2
T	AT& T	2	1
DD	E.I du Pont de Nemours	3	1
DIS	WaltDisney	1	1
MSFT	Microsoft	3	1
BP	BP	2	2
JPM	JPMorgan and Chase	2	1
C	Citigroup	3	1
MCD	McDonald′s	2	1
HON	Honeywell	2	2
BA	Boeing	1	2
AXP	*American Express*	3	1
HD	Home Depot	2	2
IP	International Paper Company	2	2
CVX	Chervon	1	2
XOM	Exxon Mobil	2	2
MRK	Merck	3	1
INTC	Intel	3	2
HPQ	HP	3	2
CAT	Caterpillar	3	2
AA	Alcoa	3	2
EKDAQ	EKDAQ	3	3
GOOGL	Google	3	3
AAPL	Apple	3	3

Figure 3.7: Cluster placement for K-means++ based on feature/model-based clustering

Following are the key points we observed.

1. As one can see feature-based clustering method, has sixteen stocks in cluster 1, eleven in cluster 2, and three in cluster 3.

2. Under Model-based clustering, there are six stocks in cluster 1, eleven stocks in cluster 2, and thirteen stocks in cluster 3.

3. As stated above, our model-based clustering method is more focused on the distribution and the correlation structure of arc length series, while the feature-based technique is more focused on the overall behavior.

Figure 3.9 presents log-returns of a few selected stocks found in clusters 1, 2, and 3 under model based technique. We choose CVX, KO, JNJ from cluster 1, T, WMT, MCD from cluster 2, and AXP, C, and MRK from cluster 3.

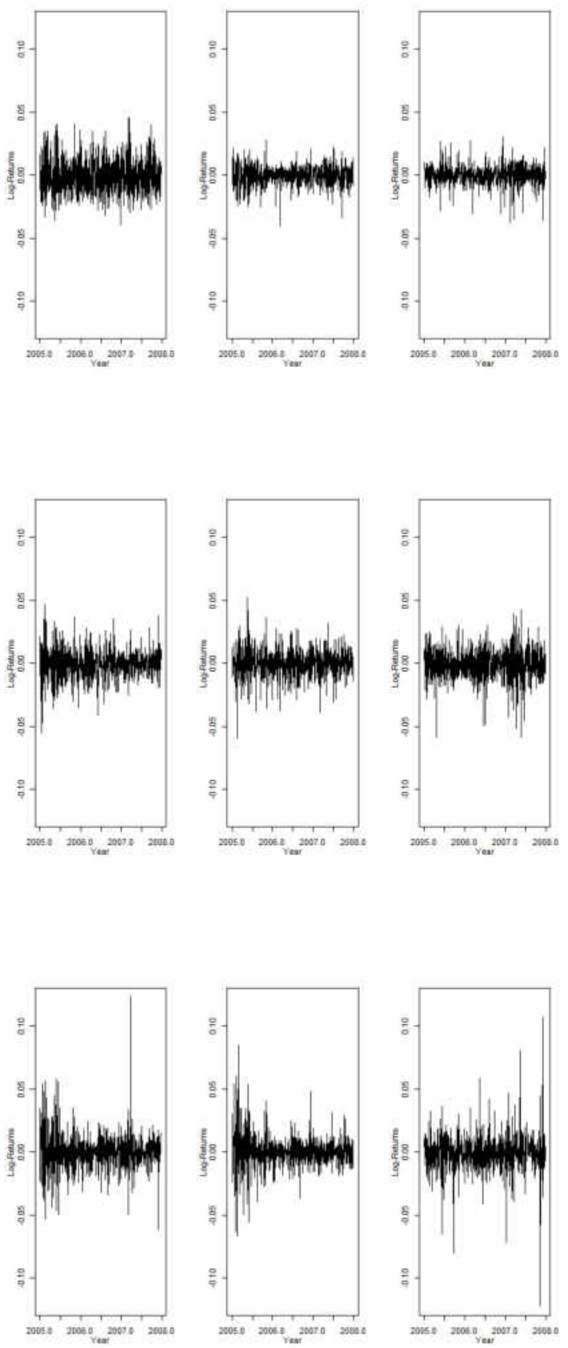

Figure 3.8: Log-returns for cluster 1(row 1), cluster 2(row 2), and cluster 3(row 3)

Below are some key take away from Figure 3.9:

1. Those three stocks in high volatile cluster 3 under model-based clustering appears in low volatile cluster 1 under feature-based clustering.

2. Stock CVX is grouped in less volatile cluster 1 under model-based clustering while it has been placed in mid-volatile cluster 2 by feature-based clustering.

3. Those three mid-volatile stocks under model based clustering has been categorized as less volatile stocks by feature-based clustering.

Figure 3.10 presents log-returns of CVX, BP, and CAT. An interesting feature we observed in figure 3.10 is that all three stocks, CVX, BP, and CAT, have been grouped in the mid-volatile cluster 2 by feature-based clustering. But the visual inspection of three plots of log returns does not support this result. The model-based technique we propose, place them in less, mid, and high volatile groups respectively.

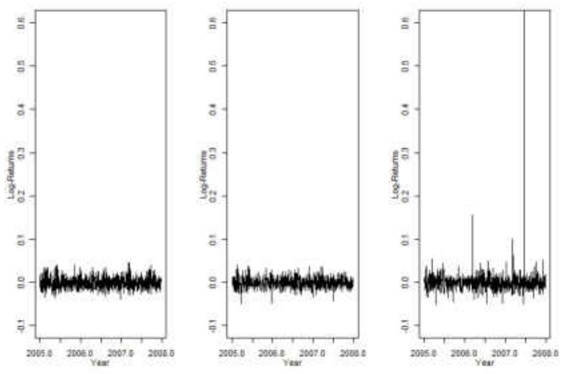

Figure 3.9: Log-returns for CVX(left), BP(middle), and CAT(right)

In addition, the Figure 3.11 shows log-returns of UTX and CAT. Visually both stocks have to be in same cluster. As expected the method we propose places them in high volatile cluster 3. But the feature-based clustering technique failed to place them in the same cluster.

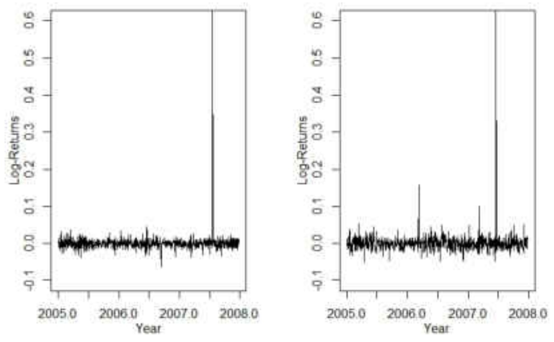

Figure 3.10: Log-returns for UTX(left) and CAT(right)

We can suggest that model-based clustering provides sensible results compared to feature-based clustering presented in Wickramarachchi et al [19].

It is important to note that the estimated scale parameter($\hat{\alpha}$) has a significant role in determining where each stock is placed. When looking at all estimated scale parameters, most of the values are over one million expect for a few. Those stocks that have a very low $\hat{\alpha}$ value are all grouped in the same cluster, which would be the third. It makes perfect sense since α represents the scale parameter. Therefore those stocks with a low $\hat{\alpha}$ value are considered as highly volatile. The converse can be confirmed as well since the stocks with the highest $\hat{\alpha}$ value, are all grouped in the first cluster.

CHAPTER 4
CONCLUSION

In this study, we explored the possibility of fitting a time series model for the arc length series of a financial time series. We proposed a GAR(1) type model. One of the main outcomes of the study is that the kernel regression based bias correction is significantly better compared to the bias correction proposed by Fernandez and Salas [7] for the kurtosis estimator. We successfully carried out a model-based clustering on thirty stocks in DWJ as an application. We were able to identify low volatile and less volatile stocks better compared to feature-based clustering proposed by Wickramarchchi et al [19].

There is a lot of room to improve our study. First, we have to generalize our simulation study as it covers a full range of parameter values. We can also consider ways of improving the fit of the model. We can explore the suitability of other types of kernel functions and some additional predictors for the model.

One other unexplored area in the bias correction of kurtosis estimator is the case where negative lag-one autocorrelation is present. Generating gamma marginals with negative lag-one autocorrelation structure also remains as an open problem.

Bibliography

[1] Black, F., Merton, R., and Scholes, M. (1973). The pricing of options and corporate liabilities. *Journal of Political Economy*, 81(3):637–654.

[2] Bobee, B. and Robitatille, R. (1975). Correction of bias in the estimation of the coefficient of skewness. *Water Resour*, 11:851–854.

[3] Bollerslev, T. (1986). Generalized autoregressive conditional heteroskedasticity. *Journal of Econometrics*, 31(3):307 – 327.

[4] Brockwell, P. J. and Davis, R. A. (2002). *Introduction to time series and forecasting*. Springer texts in statistics. Springer, New York, Berlin, Heidelberg.

[5] Casella, G. and Berger, R, L. (2002). *Statisical Inference*. Duxbury Press.

[6] Engle, R. (1982). Autoregressive conditional heteroscedasticity with estimates of the variance of united kingdom inflation. *Econometrica*, 50(4):987–1007.

[7] Fernandez, B. and J.D, S. (1990). Gamma autoregressive models for stream-flow simulation. *Journal of Hydraulic Engineering*, 116:1403–1414.

[8] Gaver, D. and Lewis.P.A.W (1980). First order autoregressive gamma sequences and point process. *Adv. Applied Probability*, 12:727–745.

[9] Johnson, S. (1967). Hierarchical clustering schemes. *Psychometrika*, 2:241–254.

[10] Kirby, W. (1974). Algebraic boundness of sample statistics. *Water Resour*, 10:220–222.

[11] Lawrance, A. J. (1982). The innovation distribution of a gamma distributed autoregressive process. *Scandinavian J. Statistics*, 9:234–236.

[12] Liao, T. (2005). Clustering of time series data-a survey. *Pattern Recognition*, 38:1857–1874.

[13] MacQueen, J. (1967). Some methods for classification and analysis of multivariate observations. *Proceedings of the 5th Berkeley Symposium on Mathematical Statistics and Probability*, 1:281–297.

[14] Ostrovsky, R., Rabani, T., Schilman, L., and Swamy, C. (2012). The effectiveness of lloyd-type methods for the k-means problem. *Journal of ACM*, 59:article 28.

[15] R Core Team (2014). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. `http://www.R-project.org/`.

[16] Thomas Mikosch, C. S. (2000). Limit theory for the sample autocorrelations and extremes of a garch (1, 1) process. *The Annals of Statistics*, 28(5):1427–1451.

[17] Wallis, J. R., Matalas, N. C., and Slack, J. R. (1974). Just a moment! *Water Resour*, 10:211–219.

[18] Whittle, P. and Sargent, T. J. (1983). *Prediction and Regulation by Linear Least-Square Methods*. University of Minnesota Press, ned - new edition edition.

[19] Wickramarachchi, T. and Tunno, F. (2015). Using arc length to cluster financial time series according to risk. *Communications in Statistics: Case Studies, Data Analysis and Applications*, 1(4):217–225.

[20] Wickramarachchi, T. D., Gallagher, C., and Lund, R. (2015). Arc length asymptotics for multivariate time series. *Applied Stochastic Models in Business and Industry*, 31(2):264–281.

Appendix A

R CODE

Appendix A contains R code for the simulation study, as well as code for the graphics generated in Chapters 1, 2, and 3.

```
## Calculating Arc Length ##

install.packages("readxl")
library(readxl) ## To read Excel Files ##

data<-read_excel("data.xlsx")
x<-data[,30]
x
n=length(x)

yt1 = 0
yt =0

## Used to plot the Arc length of a specific stock out of the group of thirty c
for(k in 10)
{
  for(i in 1:753)
  {
  yt[i] = sqrt(1+((log(data[,k][i+1]))-log(data[,k][i]))^2) #arclength
  }
  arclength = ts(yt,start=c(2005, 1), frequency=252)
  plot(arclength,ylim = c(1,1.0025),xlab="Year", main = "Arclength for CAT")
```

```
}

## Calculating and plot Log-Returns ##

for(j in 1:753)
{
  yt1[j] = log(x[j+1]/x[j])
}

plot(ts((yt1)),main = "Log Return Prices for Coca-Cola Company",xlab = "Data", 

## Plotting the Empirical Density for a certain stock ##

plot(density(yt1),main = "Kernel Density Plot of Log-Returns for Coca-Cola Coma


#####Simulation Study######

###Create Function to Simulate Data###
##N - Number of Repetitions Desired##
##n - Size of Simulated Data Sets##
##lambda,alpha,beta - Gamma Parameters##
##phi - GAR(1) Parameters##

install.packages("FAdist")
library(FAdist)## used to create a shifted gamma distribution
```

```
#Initialization#


phi = seq(-0.05,0.21,0.01)

phi = 0.05

power = c(1:8)

alpha = 10^power


beta = seq(0.004,1,0.002)

beta = 0.1

n = 754

N=n+5


#Create Function for Simulation study as suggested by Fernandez and Salas#


simu = function(n, phi, lamada, alpha, beta){

  M = rpois(N,lambda= -beta*log(phi))


  g1=0

  phihat=0


  for(k in 1:100){

    x0 = rgamma3(n=1, shape=alpha, scale=beta, thres=lamada) ##X0 term for the

    epsil = 0



    for(i in 1:N){

      if(M[i]!=0){

        Yj = rexp(M[i],rate=alpha)
        Uj = runif(M[i], 0, 1)
        eta = sum(Yj^Uj)
      } else{eta=0}
      epsil[i] = lamada*(1-phi)+eta
    }
    x=0
    x[1] = phi*x0+epsil[1] #first observation

    for(i in 2:N){
      x[i] = phi*x[i-1]+epsil[i]
    }

    x= x[6:N]

    phihat[k] = acf(x)$acf[2]
    phihat[k] = (phihat[k]*n+1)/(n-4)

    g1[k] = (n/((n-1)*(n-2)*var(x)^(3/2)))*sum((x-mean(x))^3)
  }

  phihatbar = mean(phihat)
  g1bar = mean(g1)

  gamma = 2/sqrt(beta)
  true_phi=phi
  newlist = list("phihatbar"= phihatbar, "g1bar"=g1bar, "gamma"=gamma, "true_ph
  return(newlist)
}
```

```
#Creating the Training data set for the simulation study#

v1 = sapply(beta, function(beta) sapply(phi, function(phi) sapply(alpha, simu, 

## Used to get estimated values for our simulation ##

k=seq(1,672,4)
s=seq(2,672,4)
t=seq(3,672,4)
j=1:499

phihatbarout = v1[k,j]
g1barout = v1[s,j]
gammaout = v1[t,j]

phihatvec=as.numeric(phihatbarout)
g1barvec=as.numeric(g1barout)
gammavec=as.numeric(gammaout)

d3 = data.frame(phihatvec,g1barvec,gammavec)
train1 = sample(nrow(d3),0.99*nrow(d3))
d4 <- d3[train1,] ## Training data set
d4.validate<-d3[-train1,] ## Validation data set

## Graphing the Training data set vs Full data set ##
```

```
install.packages("sm")
library(sm)

k1=sm.regression(cbind(d4[,1], d4[,2]), d4[,3],xlab="Phihatbar",ylab="G1bar",zl

k=sm.regression(cbind(phihatvec, g1barvec), gammavec) ## Full Data Set

## Creating the bandwidth for the Predicted $\gamma$ values ##

install.packages("np")
library(np)

bw <- npregbw(xdat=data.frame(d4[,1:2]), ydat=d4[,3], regtype="lc", bwmethod="c

predict.out<-fitted(npreg(exdat=data.frame(d4[,1:2]), bws=bw1)) ## Predicted Ga

## Calculating MSE, Bias, and Plot Residual Plots for Predicted $\gamma$ ##

gamma_obs = d4.validate[,3]
gamma_estimate = predict.out
variance = var(gamma_estimate)
bias = gamma_estimate - d4.validate[,3]
MSE = variance + bias^2
res = d4.validate[,3] - gamma_estimate
plot(gamma_obs,res, xlab = "gamma observed", ylab ="Residual", main = "Residual

## Calculating the MSE, Bias, and Plot Residual Plots for $\gamma$ from suggest
## Method Suggested by Fernandez and Salas ##
n = 754
L = (n-2)/sqrt(n-1)

A = 1+6.51*n^(-1)+20.2*n^(-2)
B = 1.48*n^(-1)+6.77*n^(-2)

gamma0 = (L*g1.validate*(A+B*((L^2)/n)*g1.validate^2))/sqrt(n)

f = (1 - 3.12*(phi.validate^(3.7))*n^(-.49))

gamma = gamma0/f

gamma

## Calculating the MSE and ploting the Residual Plot ##
gamma_obs = d4.validate[,3]
gamma_estiamate = gamma
variance = var(gamma)
bias = gamma - d4.validate[,3]
MSE = variance + bias^2


res3 = d4.validate[,3]-gamma
plot(gamma_obs,res3,xlab = "gamma observed",ylab="Residual",main = "Residual Pl

## Method of Moments Calculation using the unbias $\gamma$ estimation ##
```

```
### Same method but with the yt
n = length(yt)

m=mean(yt)

s2 = var(yt)



g1 = (n/((n-1)*(n-2)*s2^(3/2)))*sum((yt-m)^3)



r1 = (1/((n-1)*s2))*sum((yt[1:n-1]-m)*(yt[2:n]-m));



## Correction for r1

p1 = (r1*n+1)/(n-4)

## correction for s2

K = ((n*(1-p1^2))-2*p1*(1-p1^n))/(n*(1-p1)^2)


sigma2 = s2*(n-1)/(n-K)



## Fitting with Unbias Gamma ##
```

```
d3 = data.frame(abs(phi),g1)



Unbias_Gamma<-fitted(npreg(exdat=data.frame(d3), bws=bw))




## Estimating Shifted Gamma parameters with unbias Gamma ##


Unbias_beta = (2/Unbias_Gamma)^2

Unbias_beta


phi = p1

p1


Unbias_alpha = Unbias_beta/sigma2

Unbias_alpha


Unbias_lamada = m - Unbias_beta/Unbias_alpha

Unbias_lamada


Unbias_alpha

Unbias_beta

p1

Unbias_lamada

Unbias_Gamma
```

```
## Model-based clustering for unbias esitimates ##

alpha = c(23159914,26522442,6739005,18617797,4433520,87.45,23927783,5043800,224
beta = c(0.08255588,0.1267607,0.05573179,0.1338863,0.07468809,0.005316261,0.782
phi = c(0.10497,0.053684,0.166852,0.114144,0.055074,-0.000301,0.090807,0.115215
lamada = c(1.00003,1.00003,1.00004,1.000045,1.000061,1.000285,1.000062,1.000065
gamma = c(6.960749,5.617431,8.471855,5.465903,7.318201,27.43007,2.260544,7.0208

m = matrix(c(alpha,beta,phi,lamada,gamma),ncol =5, nrow =30)

## Creating a function for the euclidian distance calculation ##
distfunc <- function(x) {
  d <- dist(m, method = "euclidian")
  #d <- as.dist((1-cor(t(x))))

  return (d)
}

d <- distfunc(m)

#1. Hierarchical
#run hierarchical clustering using single, average, and complete methods#
groups <- hclust(d,method="single")
plot(groups, hang=-1)



#2. K-means, k = 3 for three clusters#
```

```
### k-means (this uses euclidean distance)
clusters <- kmeans(as.matrix(d), 3, nstart = 30)
plot(as.matrix(d), col = clusters$cluster)
points(clusters$centers, col = 1:10, pch = 8)

install.packages('cluster')
#better way to visualize clusters
library(cluster)
clusplot(as.matrix(d), clusters$cluster, color=T, shade=T, labels=2, lines=0)



## K-means++ clustering ##

alpha = c(23159914,26522442,6739005,18617797,4433520,87.45,23927783,5043800,224
beta = c(0.08255588,0.1267607,0.05573179,0.1338863,0.07468809,0.005316261,0.782
phi = c(0.10497,0.053684,0.166852,0.114144,0.055074,-0.000301,0.090807,0.115215
lamada = c(1.00003,1.00003,1.00004,1.000045,1.000061,1.000285,1.000062,1.000065
gamma = c(6.960749,5.617431,8.471855,5.465903,7.318201,27.43007,2.260544,7.0208

m = matrix(c(alpha,beta,phi,lamada),ncol =4, nrow =30)

k=3



s=vector()
s[1]=sample(1:30,1)
```

```
c=matrix(,ncol=4)
c[1,] = m[s[1],]

d1=1:30

for(i in 1:30){
  d1[i]=sum((c[1,]-m[i,])^2)  ##computeD(x)^2
}
prob=d1/sum(d1)
s[2]=sample(1:30,1,prob=prob)

c = rbind(c,m[s[2],])

for(i in 1:30){
  d1[i] = min(sum((c[1,]-m[i,])^2), sum((c[2,]-m[i,])^2)) ## Compute min(D1^2,
}
prob=d1/sum(d1)
s[3]=sample(1:30, 1, prob=prob)
c = rbind(c,m[s[3],])

kmeans(m, c)
```