Data Nuggets: A Method for Reducing Big Data While Preserving Data Structure

Traymon E. Beavers[1,2], Javier Cabrera [*] [2], Mariusz Lubomirski[1], Dhammika Amaratunga[1], and Jeffrey E. Teigler[1]

[1]Janssen R&D, Spring House, PA
[2]Department of Statistics, Rutgers University, New Brunswick NJ

October 22, 2022

Abstract

Big data has created new challenges for data analysis, particularly in the realm of creating meaningful clusters of data. Clustering techniques, such as K-means or hierarchical clustering are popular methods for performing exploratory analysis on large datasets. Unfortunately, these methods are not always possible to apply to big data due to memory or time constraints generated by calculations of order $P * \frac{N(N-1)}{2}$. To circumvent this problem, typically the clustering technique is applied to a random sample drawn from the dataset; however, a weakness is that the structure of the dataset, particularly at the edges, is not necessarily maintained. We propose a new solution through the concept of "data nuggets", which reduce a large dataset to a small collection of nuggets of data, each containing a center, weight, and scale parameter. The data nuggets are then input into algorithms that compute methods such as principal components analysis and clustering in a more computationally efficient manner. We show the consistency of the data nuggets based covariance estimator and apply the methodology of data nuggets to perform exploratory analysis of a flow cytometry dataset containing over one million observations using weighted PCA and weighted K-means clustering. Supplementary materials for this article are available online.

Keywords: big data, clustering, random sampling, weighted K-means

[*]Javier Cabrera is funded by a grant, in part, funded by the National Heart, Lung, and Blood Institute (NHLBI), National Institutes of Health (R01-HL150065).

1 Introduction

Extremely large datasets, sometimes known as “Big Data”, are common in most areas of research and business including the pharmaceutical industry (?). Such data is computationally difficult to analyze so it is standard to use a smaller random sample for the analysis, although the sample to sample variability may produce analyses with substantially different results. This motivates the use of representative samples that may preserve the structure of the original dataset.

Flury considered the idea of a representative sample of a distribution F or a set of “principal points” which are a set of k points that minimize the Euclidean distance from a random variable from F to the closest point in the set (?). Tibshirani extended this idea to a set of “principal lines” that approximate a dataset (?).

Ghosh, Cabrera, et al. analyzed big data in the form of comorbidity binary variables for millions of patients (?). They grouped the data to reduce the number of observations and applied a weighted form of K-means clustering.

Independently, Mak and Joseph proposed the concept of “support points” to represent big data with a small subset of observations in ?. These support points are the same as principal points but instead they minimize the energy distance (?).

Immunological research has been transformed by the continued generation and improvement of flow cytometry methods (?). These methods use fluorochrome-conjugated antibodies to differentiate cell populations based on surface and internal expression of delineating proteins. Datasets generated by a standard flow cytometry experiments routinely include the interrogation of millions of cells with greater than 12 different parameters, often more. Classical flow cytometry analysis required bivariate graphing for visualization, but as more parameters are measured this becomes inefficient and can lead to overlooking of important cellular phenomena. Methods for dimensionality reduction and cluster identification therefore become critical for managing these large datasets.

The most typical method would be to apply a clustering technique to the dataset, such as K-means clustering or hierarchical clustering; however, a dataset as large as those found in flow cytometry experiments would require far too many resources, such as computational memory and time. As it will be shown later, earlier proposals such as random samples or support points may not be able to capture the structure around the edges of the dataset.

We propose a different method which instead reduces the millions of data points into a smaller collection of "data nuggets". All the individual data points coalesce into many data nuggets, while still retaining the structure of the data. A weighted form of K-means clustering can then be used to configure the data nuggets into various clusters

Section 2 introduces notation and provides a brief overview of the issues that can arise when attempting to use common clustering methods to cluster large datasets. Section 3 describes the algorithm for creating data nuggets and the algorithm for creating clusters using weighted K-means clustering. This section also provides simulation results comparing the accuracy of K-means clustering to weighted K-means clustering for clustering data nuggets generated from a dataset with binary variables, compares data nuggets to the support points given by Mak and Joseph, and provides a demonstration of how well data nuggets perform on datasets when P is large using a simulated dataset. Section 4 applies the algorithm to a flow cytometry dataset containing over one million B cells. Section 5 describes an R package created to use the method and future work that can be done concerning this method.

2 Limitations for Large Datasets

We now introduce notation by generalizing our motivating example of a flow cytometry dataset. Suppose an experiment with N observations (B cells), where N is in the millions, is conducted to measure the level of expression of P different proteins. Let $\mathbf{X}$ be the matrix containing the information pertaining to the levels of expression of each protein for each B cell so that:

$$\mathbf{X} = \begin{bmatrix} \underset{\sim}{\mathbf{x}}_1 \\ \underset{\sim}{\mathbf{x}}_2 \\ \vdots \\ \underset{\sim}{\mathbf{x}}_{N-1} \\ \underset{\sim}{\mathbf{x}}_N \end{bmatrix} = \begin{bmatrix} x_{11} & x_{22} & \cdots & x_{1(p-1)} & x_{1p} \\ x_{21} & x_{22} & \cdots & x_{2(p-1)} & x_{2p} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ x_{(N-1)1} & x_{(N-1)2} & \cdots & x_{(N-1)(P-1)} & x_{(N-1)P} \\ x_{N1} & x_{N2} & \cdots & x_{N(P-1)} & x_{NP} \end{bmatrix}$$

Where $\underset{\sim}{\mathbf{x}}_n$ is the row vector containing the protein expression levels for the n^{th} B cell and x_{np} is the level of expression of protein p for the n^{th} B cell, for $n = 1, 2, ..., N$ and $p = 1, 2, ..., P$.

The goal of the experiment is to find out if there are any meaningful groups of cells based on the level of expression of the P different proteins. We can search for these groups of cells by placing cells into different clusters and then determining if any of the proteins have a particularly weak or strong level of expression in any of the clusters. The memory usage and computation needed is of the order of $P * \frac{N(N-1)}{2}$ for typical clustering techniques. In general, clustering is performed in one of two ways: through K-means clustering or hierarchical clustering (**?**).

Both methods of clustering have limitations for very large datasets. For K-means clustering, the final cluster assignments heavily depend on the initial choice of cluster centers (**?**). A clear remedy for this is to choose multiple initial cluster centers, conduct K-means clustering, and choose the set of clusters which minimizes the total within cluster sum of squares. For datasets with a large number of observations many initial centers may need to be attempted. For the LLoyd, Forgy, and MacQueen algorithms given in **???**, the time cost is high in R, which may lead the user to sacrifice the number of initial cluster centers they choose to evaluate.

On the other hand, the Hartigan & Wong algorithm given in **?** may fail to finish running for large datasets because the memory cost necessary to store the closest cluster assignment and the second closest cluster assignment for each observation is too high. This is also the case for hierarchical clustering methods which may not even have a chance to begin because the distance matrix cannot be formed for datasets that are too large.

A common solution to this problem has been to retrieve a random sample of the data and use a clustering algorithm on this reduced dataset. The intuition is that if the sample is sufficiently large, the data structure of the sampled data should match the data structure of the entire data. Unfortunately, this intuition does not always hold. Further, since a distance matrix is needed for hierarchical clustering, the random sample may need to be reduced to a very small amount of observations when compared to the entire dataset.

Another possible solution is to reduce the large dataset to a set of only a few data points which are chosen to represent the dataset as a whole in a way that is different than the methods given in the form of principal points, principal lines, or support points. To avoid the memory and time constraints of using full datasets or large random samples, the pitfalls associated with massive data reduction, and the lack of focus on the edges of the data structure which may occur when using support points, we propose using data nuggets.

3 Data Nuggets

The process of creating data nuggets is inspired by the idea of using a P-dimensional grid that encompasses the entire dataset to partition the observations into M equally sized cells. Each cell would represent a data nugget, where the centers of these cells would form the data nugget centers, the amount of observations from the dataset which exist in these cells would form the data nugget weights, and the trace of the covariance matrices for the observations within each cell divided by P would form the data nugget scales.

When both P and M are low this is a relatively simple feat, but when either P or M is large the amount of computational resources required becomes unrealistic. A more feasible option would be to use observations already within the dataset as the initial centers of data nuggets. This can be done by choosing observations in the dataset which are as equally spaced apart as possible. Then all the remaining observations are assigned to the data nugget with the nearest center according to a distance metric. This method will ensure that each observation will only be assigned to a single data nugget and each data nugget will contain at least one observation. Each data nugget is then re-centered by either finding the mean of the observations it contains or choosing a random observation to be the center.

We detail the algorithm to create data nuggets below.

Algorithm 1. *Create M data nuggets given: $\mathbf{X}$, an $N \times P$ data matrix; a choice for how a data nugget's center will be chosen (mean or random); R, the number of observations to randomly sample from $\mathbf{X}$ when finding portions of the initial number of data nugget centers; C, the proportion of observations to remove from the data matrix at each iteration when finding data nugget centers; M_{init}, the initial number of data nugget centers to create; M, the final number of data nuggets to create; and D, a distance metric.*

1. Randomly split $\mathbf{X}$ into the set of submatrices $\{\mathbf{X}_g | g = 1, 2, ..., G\}$, where $G = \lfloor \frac{N}{R} \rfloor + \mathbb{1}(R \pmod R > 0)$ and $\mathbb{1}(\mathscr{A}) = 1$ when $\mathscr{A}$ is true and 0 otherwise.

2. For $g = 1, 2, ..., G - 1$:

   2.1 Create a distance matrix for the observations contained in $\mathbf{X}_g$ using the given distance metric D.

   2.2 Let N_g be the current number of observations contained in $\mathbf{X}_g$. Find the $\lfloor CN_g \rfloor$ smallest non-diagonal distances in the matrix. Let $\mathcal{A}$ and $\mathcal{B}$ be the set of $\lfloor CN_g \rfloor$ observations from the rows and columns of the distance matrix, respectively, that have these distances between them.

   2.3 Remove $\mathcal{A} \cup \mathcal{B}$ from $\mathbf{X}_g$.

   2.4 Repeat steps 2.1 through 2.3 until $N_g = \lceil \frac{M_{init}}{G} \rceil$ and let $\mathbf{X}_g^* = \mathbf{X}_g$.

3. If the number of observations contained $\mathbf{X}_G$ is greater than $\lceil \frac{M_{init}}{G} \rceil$, conduct steps 2.1 through 2.4 (replacing $\mathbf{X}_g$ and $\mathbf{X}_g^*$ with $\mathbf{X}_G$ and $\mathbf{X}_G^*$, respectively) to produce data matrix $\mathbf{X}_G^*$. Otherwise, let $\mathbf{X}_G^* = \mathbf{X}_G$

4. Let $(\mathbf{X}^*)' = [(\mathbf{X}_1^*)', (\mathbf{X}_2^*)', ..., (\mathbf{X}_G^*)']$ so that $\mathbf{X}^*$ is the data matrix containing the (roughly) M_{init} centers of the initial set of data nuggets.

5. Conduct steps 2.1 through 2.4 (replacing $\mathbf{X}_g$, $\lceil \frac{M_{init}}{G} \rceil$, and $\mathbf{X}_g^*$ with $\mathbf{X}^*$, M, and $\mathbf{X}^{**}$, respectively) to produce data matrix $\mathbf{X}^{**}$. The M observations that remain in $\mathbf{X}^{**}$ are the initial centers of the final set of data nuggets. Let data nugget j have center $\underset{\sim}{\mathbf{c}}_j$ for $j = 1, 2, ..., M$.

6. For each observation $\underset{\sim}{\mathbf{x}}_i$ in data matrix $\mathbf{X}$, assign $\underset{\sim}{\mathbf{x}}_i$ to data nugget j so that $D\left(\underset{\sim}{\mathbf{x}}_i, \underset{\sim}{\mathbf{c}}_j\right)$ is minimized over j. Let N_j be the number of observations assigned to data nugget j, and let $w_j = N_j$ be the weight of data nugget j for $j = 1, 2, ..., M$.

7. Re-center all of the data nuggets by choosing $\underset{\sim}{\mathbf{c}}_j$ to be either the mean of all the observations assigned to data nugget j or a random observation assigned to data nugget j, depending on the user's choice.

8. Finally, let $s_j = \frac{tr(Cov(\mathbf{X}_j))}{P}$ be the scale of data nugget j when $N_j > 1$, where $\mathbf{X}_j$ is the submatrix of observations from $\mathbf{X}$ which belong to data nugget j. When $N_j = 1, s_j = 0$.

The computational cost of this algorithm in terms of arithmetic complexity can be defined by $O\Big(P * \Big[G(\frac{R(R-1)}{2})(\psi_1) + \frac{(M_{init})(M_{init}-1)}{2}(\psi_2) + (N+1)M\Big]\Big)$ where ψ_1 and ψ_2 are the number of iterations it takes to reduce R to $\lceil \frac{M_{init}}{G} \rceil$ and M_{init} to M, respectively. As such, $2 \leq \psi_1 \leq R - \frac{M_{init}}{G} + 1$ and $2 \leq \psi_2 \leq M_{init} - M + 1$. Where ψ_1 and ψ_2 fall within their respective ranges depends on the choice of C. Specifically:

$$\lim_{C\to 0} \psi_1 = R - \frac{M_{init}}{G} + 1, \quad \lim_{C\to 0} \psi_2 = M_{init} - M + 1, \quad \lim_{C\to 1} \psi_1 = \lim_{C\to 1} \psi_2 = 2$$

Figure 1 uses density plots to compare the amount of data structure maintained after reducing a bivariate dataset of 15,601 observations to a simple random sample of 2,000 observations versus reducing that same bivariate dataset to 2,000 data nuggets using **Algorithm 1**. Figure 1 and all figures that follow were created using R (**?**).

The data nuggets were created by reducing the entire dataset to 12,304 initial data nugget centers ($R = 5,000$, $C = 0.10$, and $M_{init} =$ 15,601) which were then reduced to 2,000 data nuggets ($M =$ 2,000) with data nugget centers chosen to be the mean and Euclidean distance as the chosen distance metric. The dataset is a mixture of data derived by sampling 15,000 observations from two independent standard normal distributions and combining these observations with 601 observations which create a "smile" that is hidden inside the random noise.

The upper left plot shows a scatter plot of the entire dataset of 15,601 observations, the upper right plot shows the density plot of the entire dataset of 15,601 observations, the

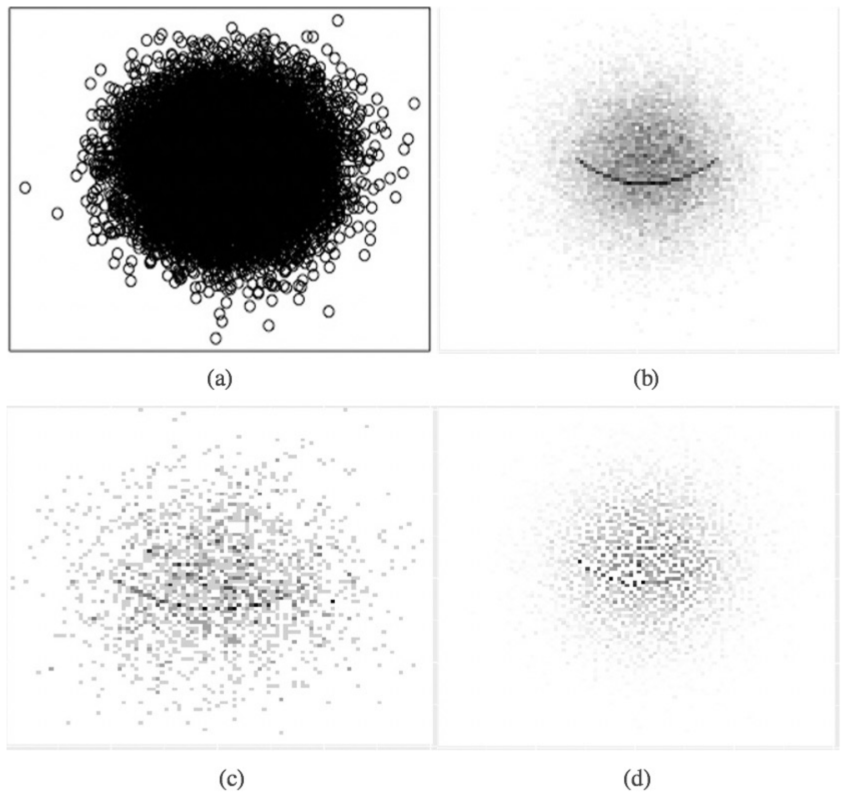


Figure 1: Comparing Density Plots for Random Sample and Data Nuggets

lower left plot shows the density plot of the random sample of 2,000 observations, and the lower right plot shows the density plot for the 2,000 data nuggets.

The density plots for the entire dataset and the random sample are created by dividing the area of the original scatterplot into a 100×100 grid and counting the number of observations in the dataset which fall inside each cell of the grid. The cells are then colored on a gradient according to how many observations are in the cell. Cells with a low number of observations produce cool colors like blue or light green while cells with a higher number of observations produce hot colors such as yellow or orange.

The density plot for the data nuggets is produced in a similar manner but with a slight modification. Once again, the area of the original scatterplot is divided into a 100×100 grid. However, instead of using the number of data nuggets that fall within each cell, the sum of the weights of the data nuggets that fall within the cell is used. Then the cells are colored accordingly.

As shown in Figure 1, the density plot for the entire dataset clearly shows a thin smile inside the ball of random noise. The density plot for the random sample faintly produces the smile, but most of the smile is colored green (unlike the smile in the density plot for the entire dataset which is colored yellow) and there is a large amount of random noise surrounding the smile.

The density plot for the data nuggets shows a much more distinct smile. Most of this smile is colored yellow or orange and the amount of random noise is much more concentrated around the smile, matching what is seen in the density plot for the entire dataset.

Data nuggets can also be refined by splitting data nuggets with scale parameters too large and/or "shapes" too nonspherical. The purpose of this method is to provide each data nugget with a more common level of within-data-nugget variability. The largest eigenvalue of a data nugget (i.e. the largest eigenvalue yielded from the covariance matrix of the dataset composed of the observations assigned to that data nugget) is used as a proxy for the within-data-nugget variability. Specifically, if the largest eigenvalue of a data nugget is too large, then we believe that the within-data-nugget variability is too large.

Data nuggets are refined as detailed in the algorithm below.

Algorithm 2. *Refine M data nuggets given: $\mathbf{X}$, an $N \times P$ data matrix; a set of M data nuggets formed from $\mathbf{X}$ using **Algorithm 1**; ν, a percentile for splitting data nuggets according to their largest eigenvalue; and N_{min}, the minimum number of observations that a data nugget must contain as a result of this algorithm.*

1. For every data nugget:

   1.1 Let $\mathbf{X}_j$ be the submatrix of observations from $\mathbf{X}$ which belong to data nugget j.

   1.2 When $P \geq 2$ let ζ_j be the largest eigenvalue of $Cov\left(\mathbf{X}_j\right)$. When $P = 1$ let ζ_j be the scale of data nugget j.

2. Obtain η, the quantile of the non-zero ζ_j's corresponding to the ν^{th} percentile.

3. Create $\mathcal{A}$, a list of all data nuggets with scales larger than η.

4. For every data nugget $j \in \mathcal{A}$:

4.1 If data nugget j contains at least greater than $2N_{min}$ observations, split data nugget j into two new data nuggets using K-means clustering.

4.2 If either of the two new data nuggets created in step 4.1 contain less than N_{min} observations, delete these two data nuggets and retain data nugget j. Otherwise, delete data nugget j and remove data nugget j from $\mathcal{A}$.

5. Repeat steps 2 and 3 until $\mathcal{A}$ is empty or step 4 is completed without any data nuggets being removed from $\mathcal{A}$.

Note that it is possible that data nuggets may be split an undesirable amount of times if the algorithm is left unchecked. As such, a limit can be placed on the number of times steps 2, 3 and/or 4 is executed before the algorithm ends.

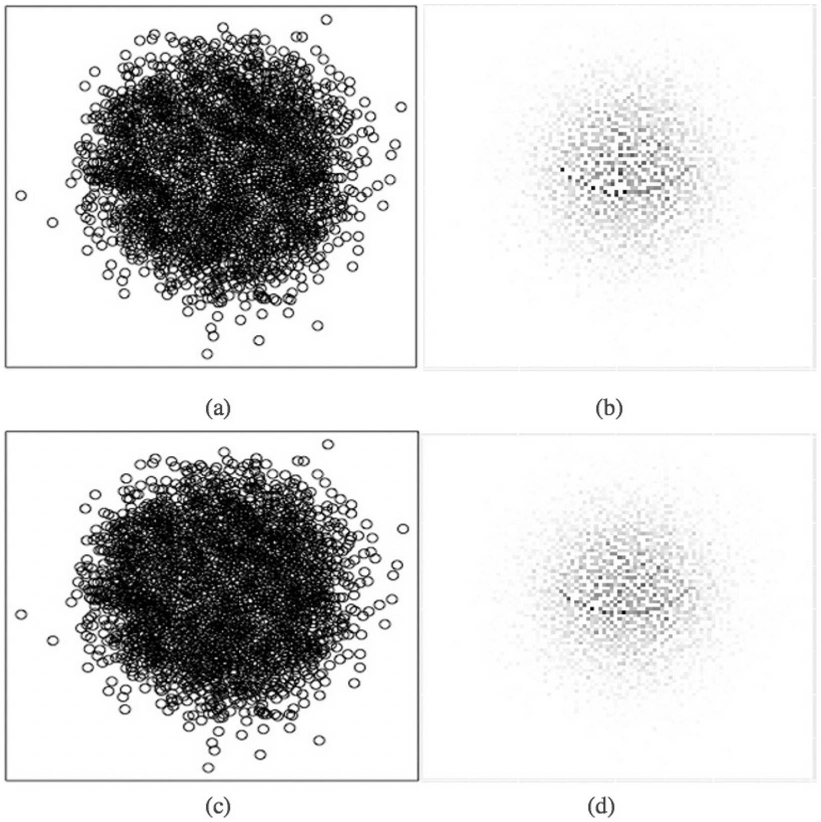


Figure 2: Comparing Density Plots for Original and Refined Data Nuggets

Using density plots, Figure 2 compares the original 2,000 data nuggets refined to 2,504 data nuggets using **Algorithm 2** with $\nu = .5$ and $N_{min} = 2$.

The first row of plots is the scatter plot of the original 2,000 data nuggets (upper left plot) beside its corresponding density plot (upper right plot). The second row of plots is the scatter plot of the refined 2,504 data nuggets (lower left plot) beside its corresponding density plot (lower right plot). The density plot for the 2,504 data nuggets has a much more consistent smile with fewer gaps and the ball of random noise is slightly more concentrated around the smile.

After the data nuggets are created, modified versions of common statistical techniques can be applied to them. In this paper we explore weighted K-means clustering and weighted principal component analysis (PCA). In both of these methods we ignore the scale parameter of the data nuggets and instead focus on using the weight parameter. This is done because the within-data-nugget variability of the data nuggets is minuscule for a large enough number of data nuggets, provided the mean is chosen to be the data nuggets' centers. This notion is formalized in **Lemma 1** and **Theorem 1**.

Let $\mathbf{X}$ be an $N \times P$ data matrix whose rows are random vectors from some population with mean $\underset{\sim}{0}$ and covariance matrix $\boldsymbol{\Sigma}$. Without loss of generality, let $\mathbf{X}$ be centered at $\underset{\sim}{0}$ and let the sample covariance matrix of $\mathbf{X}$ be $\mathbf{S} = (N-1)^{-1}\mathbf{X}\mathbf{X}'$. Let $\{\underset{\sim}{\mathbf{c}}_1, ..., \underset{\sim}{\mathbf{c}}_M\}$, $\{w_1, ..., w_M\}$, and $\{s_1, ..., s_M\}$ be the centers, weights, and scales, respectively, of M data nuggets created with **Algorithm 1** with data nugget centers chosen to be the mean and using Euclidean distance as the choice of distance metric and refined with **Algorithm 2** to form a representative dataset of $\mathbf{X}$.

Also, let these data nuggets be ordered so that $s_1 \leq s_2 \leq \cdots \leq s_M$. Further, let $\mathbf{X}_j$ be the submatrix of w_j observations from $\mathbf{X}$ contained in data nugget j for $j = 1, ..., M$. Finally, let $\mathbf{S}_{DN} = (N-1)^{-1}\sum_{j=1}^{M} w_j \mathbf{c}_j \mathbf{c}_j'$, $\boldsymbol{\Delta}_j = Cov(\mathbf{X}_j)$ for $j = 1, ..., M$, and $\boldsymbol{\Delta} = (N-1)^{-1}\sum_{j=1}^{M}(w_j - 1)\boldsymbol{\Delta}_j$. Note that $\mathbf{S}_{DN}$ and $\boldsymbol{\Delta}$ are estimates of the variability between the data nugget centers and the sum of the within-data-nugget variability of the M data nuggets, respectively. Then the following can be shown:

Lemma 1. *If all M data nuggets locally have approximately a continuous uniform distribution inside a region that is approximately a P-dimensional sphere and $s_1 \approx s_2 \approx \cdots \approx s_M \approx s$ for some $s > 0$, then there is a method for splitting these M data nuggets so that* *when N increases to $2^P N$, M increases to $2^P M$ and $\lim_{N\to\infty} s_{M^*} = 0$ where M^* is the number of data nuggets created and refined to form a representative dataset for N observations.*

Proof. Since all M data nuggets locally have approximately a continuous uniform distribution inside a P-dimensional sphere, let the radius of this sphere for data nugget j be r_j for $j = 1, 2, ..., M$. Further, $\boldsymbol{\Delta}_j \approx s_j^2 \mathbb{I}_P$ for $j = 1, 2, ..., M$, where $\mathbb{I}_P$ is the $P \times P$ identity matrix. Also, since $s_j^2 \mathbb{I}_P$ is the covariance matrix for a continuous uniform distribution inside a P-dimensional sphere with radius r_j, $s_j = br_j$ for some $0 < b < 1$ for $j = 1, 2, ..., M$; therefore, since $s_1 \approx s_2 \approx \cdots \approx s_M \approx s$, $r_1 \approx r_2 \approx \cdots \approx r_M \approx r'$.

Let $\mathbf{X}^{(i)}$, $N^{(i)}$, $M^{(i)}$, $s^{(i)}$ be the new data matrix, number of observations, number of data nuggets, and common scale of the data nuggets, respectively, the i^{th} time the data nuggets are split and let $\mathbf{X}^{(0)}$, $N^{(0)}$, $M^{(0)}$, $s^{(0)}$ be $\mathbf{X}$, N, M, and s respectively.

Suppose we begin with N observations and M data nuggets. Suppose $(2^P - 1)N$ more observations are collected so that $\mathbf{X}$ becomes $\mathbf{X}^{(1)}$, an $N^{(1)} \times P$ data matrix, where $N^{(1)} = 2^P N$. To accomodate these new observations, suppose we split the M data nuggets into $M^{(1)} = 2^P M$ data nuggets so that each new data nugget has approximately a continuous uniform distribution inside a P-dimensional sphere with radius $r_j \approx \frac{r'}{2}$ for $j = 1, 2, ..., M^{(1)}$. As such, $s_j \approx s^{(1)} < 2^{-1} r'$ for $j = 1, 2, ..., M^{(1)}$.

If $M^{(i-1)}$ data nuggets are split into $M^{(i)}$ data nuggets in the manner described above every time $(2^P - 1)N^{(i-1)}$ observations are added to data matrix $\mathbf{X}^{(i-1)}$ for $i = 2, 3, 4, ...$, then the i^{th} time the data nuggets are split, $s_{M^{(i)}} \approx s^{(i)} < 2^{-i} r'$. Finally, consider that $\lim_{i\to\infty} 2^{-i} r' = 0 \implies \lim_{i\to\infty} s^{(i)} = 0 \implies \lim_{i\to\infty} s_{M^{(i)}} = 0$ and $i \to \infty \iff N \to \infty$; therefore, $\lim_{N\to\infty} s_{M^*} = 0$ where $M^* = M^{(i)}$.

$\square$

Theorem 1. *If all M data nuggets locally have approximately a continuous uniform distribution inside a region that is approximately a P-dimensional sphere and $s_1 \approx s_2 \approx \cdots \approx s_M \approx s$ for some $s \geq 0$, then $\boldsymbol{\Sigma} = \lim_{N\to\infty} \mathbf{S} \approx \lim_{N\to\infty} \mathbf{S}_{DN}$.*

Proof. Let:

$$\mathbf{X} = \begin{bmatrix} \underset{\sim}{\mathbf{x}}_1 \\ \underset{\sim}{\mathbf{x}}_2 \\ \vdots \\ \underset{\sim}{\mathbf{x}}_N \end{bmatrix}$$

where $\underset{\sim}{\mathbf{x}}_i$ is the $1 \times P$ vector of responses for observation i. Further, let $\underset{\sim}{\delta}_{jk} = \underset{\sim}{\mathbf{c}}_j - \underset{\sim}{\mathbf{x}}_{jk}$ for $j = 1, 2, ..., M; k = 1, 2, ..., w_j$. First note that:

$$(N-1)\mathbf{S} = \mathbf{X}\mathbf{X}' = \sum_{i=1}^{N} \underset{\sim}{\mathbf{x}}_i \underset{\sim}{\mathbf{x}}_i'$$

Now observe

$$\sum_{i=1}^{N} \underset{\sim}{\mathbf{x}}_i \underset{\sim}{\mathbf{x}}_i' = \sum_{k=1}^{w_1} \underset{\sim}{\mathbf{x}}_{1k} \underset{\sim}{\mathbf{x}}_{1k}' + \sum_{k=1}^{w_2} \underset{\sim}{\mathbf{x}}_{2k} \underset{\sim}{\mathbf{x}}_{2k}' + \cdots + \sum_{k=1}^{w_M} \underset{\sim}{\mathbf{x}}_{Mk} \underset{\sim}{\mathbf{x}}_{Mk}'$$

$$\sum_{j=1}^{M} \sum_{k=1}^{w_j} \underset{\sim}{\mathbf{x}}_{jk} \underset{\sim}{\mathbf{x}}_{jk}' = \sum_{j=1}^{M} \sum_{k=1}^{w_j} (\underset{\sim}{\mathbf{c}}_j - \underset{\sim}{\delta}_{jk})(\underset{\sim}{\mathbf{c}}_j - \underset{\sim}{\delta}_{jk})'$$

$$\sum_{j=1}^{M} \sum_{k=1}^{w_j} \left(\underset{\sim}{\mathbf{c}}_j \underset{\sim}{\mathbf{c}}_j' - \underset{\sim}{\mathbf{c}}_j \underset{\sim}{\delta}_{jk}' - \underset{\sim}{\delta}_{jk} \underset{\sim}{\mathbf{c}}_j' + \underset{\sim}{\delta}_{jk} \underset{\sim}{\delta}_{jk}' \right)$$

Since the data nuggets are re-centered after all observations are assigned, $\sum_{k=1}^{w_j} \underset{\sim}{\mathbf{c}}_j \underset{\sim}{\delta}_{jk}' = \sum_{k=1}^{w_j} \underset{\sim}{\delta}_{jk} \underset{\sim}{\mathbf{c}}_j' = \mathbf{0}_P$ where $\mathbf{0}_P$ is the $P \times P$ matrix of zeroes. So:

$$\sum_{j=1}^{M} \sum_{k=1}^{w_j} \left(\underset{\sim}{\mathbf{c}}_j \underset{\sim}{\mathbf{c}}_j' - \underset{\sim}{\mathbf{c}}_j \underset{\sim}{\delta}_{jk}' - \underset{\sim}{\delta}_{jk} \underset{\sim}{\mathbf{c}}_j' + \underset{\sim}{\delta}_{jk} \underset{\sim}{\delta}_{jk}' \right) = \sum_{j=1}^{M} \sum_{k=1}^{w_j} \underset{\sim}{\mathbf{c}}_j \underset{\sim}{\mathbf{c}}_j' + \sum_{j=1}^{M} \sum_{k=1}^{w_j} \underset{\sim}{\delta}_{jk} \underset{\sim}{\delta}_{jk}'$$

Observe that $\mathbf{\Delta}_j = \sum_{k=1}^{w_j} \underset{\sim}{\delta}_{jk} \underset{\sim}{\delta}_{jk}'$ for $j = 1, 2, ..., M$. So:

$$\sum_{j=1}^{M} \sum_{k=1}^{w_j} \underset{\sim}{\mathbf{c}}_j \underset{\sim}{\mathbf{c}}_j' + \sum_{j=1}^{M} \sum_{k=1}^{w_j} \underset{\sim}{\delta}_{jk} \underset{\sim}{\delta}_{jk}' = \sum_{j=1}^{M} w_j \underset{\sim}{\mathbf{c}}_j \underset{\sim}{\mathbf{c}}_j' + \sum_{j=1}^{M} \mathbf{\Delta}_j$$

and

$$\mathbf{S} = (N-1)^{-1} \sum_{j=1}^{M} w_j \underset{\sim}{\mathbf{c}}_j \underset{\sim}{\mathbf{c}}_j' + (N-1)^{-1} \sum_{j=1}^{M} \mathbf{\Delta}_j = \mathbf{S}_{DN} + (N-1)^{-1} \sum_{j=1}^{M} \mathbf{\Delta}_j$$

Since all M data nuggets have approximately identical scales equivalent to s for some $s > 0$, $\sum_{j=1}^{M} \mathbf{\Delta}_j \approx M s^2 \mathbb{I}_P$ and $\mathbf{\Delta} \approx (N-1)^{-1}(N-M)s^2\mathbb{I}_P$, so:

$$\mathbf{S} \approx \mathbf{S}_{DN} + M(N-M)^{-1}\mathbf{\Delta}$$

By **Lemma 1** and using the method provided in the proof of **Lemma 1**:

$$\lim_{N\to\infty} M(N-M)^{-1}\mathbf{\Delta} \approx \lim_{N\to\infty} M(N-1)^{-1}s^2\mathbb{I}_P \approx \lim_{N\to\infty} M(N-1)^{-1}s_{M^*}^2\mathbb{I}_P = \mathbf{0}_P$$

Therefore:

$$\mathbf{\Sigma} = \lim_{N\to\infty} \mathbf{S} \approx \lim_{N\to\infty} \mathbf{S}_{DN} + \lim_{N\to\infty} M(N-M)^{-1}\mathbf{\Delta} = \lim_{N\to\infty} \mathbf{S}_{DN}$$

□

By **Theorem 1**, given a large enough sample size and a large enough number of corresponding data nuggets, the amount of variability within the dataset is preserved when the dataset is reduced without accounting for the within-data-nugget variability of each data nugget. As such, this within-data-nugget variability can be ignored when applying statistical techniques to the data nuggets.

3.1 Weighted K-means Clustering for Data Nuggets

We now introduce a weighted K-means clustering algorithm that can be used to form clusters of these data nuggets. It is worth noting that other weighted K-means clustering methods have been developed. An example is an algorithm that can be used for analyzing social networks (**?**). This algorithm is designed for the purpose of finding clusters of nodes in a social network where weights are assigned to the edges that connect the nodes. The weights of the edges are described as the "intimacy" level between the two nodes that the edge connects. In our algorithm, the weights of each data nugget are a measure of how many observations from the original dataset are contained in the data nugget.

We describe a method of weighted K-means clustering to form clusters of data nuggets with the algorithm below.

Algorithm 3. *Conduct weighted K-means clustering to form clusters of data nuggets given: M data nuggets; K, the number of clusters to be created; $\underset{\sim}{\mathbf{w}}$, the $M \times 1$ vector containing the weight of each data nugget; and $p_{reassign}$, the proportion of random data nuggets to attempt to reassign in step 4.*

Let the total weighted within cluster sum of squares, Ω, be defined by

$$\Omega \equiv \sum_{k=1}^{K} \sum_{\underset{\sim}{\mathbf{x}} \in L_k} \underset{\sim}{\mathbf{w}}'(\underset{\sim}{\mathbf{x}} - \underset{\sim}{\mu}_k)'(\underset{\sim}{\mathbf{x}} - \underset{\sim}{\mu}_k)$$

where $\{L_k | j = 1, 2, ..., K\}$ is the set of K clusters and $\{\underset{\sim}{\mu}_k | k = 1, 2, ..., K\}$ the set of respective cluster centers.

1. Choose K data nugget centers to be the initial cluster centers, $\underset{\sim}{\mu}_{01}, \underset{\sim}{\mu}_{02}, ..., \underset{\sim}{\mu}_{0K}$, of K clusters, $L_1, L_2, ..., L_K$, respectively.

2. For $j = 1, 2, ..., M$, assign data nugget j to the cluster L_k for which $D(\underset{\sim}{\mathbf{c}}_j, \underset{\sim}{\mu}_{0k})$ is minimized over $k = 1, 2, ...K$ where $\underset{\sim}{\mathbf{c}}_j$ is the center of data nugget j and D is the Euclidean distance metric.

3. Recalculate the cluster centers $\underset{\sim}{\mu}_1, \underset{\sim}{\mu}_2, ..., \underset{\sim}{\mu}_K$ as the mean of the centers of all the data nuggets within clusters $L_1, L_2, ..., L_K$, respectively.

4. For $i = 1, 2, ..., \lceil M * p_{\text{reassign}} \rceil$ randomly selected data nuggets:

    4.1 Retrieve the cluster assignment for data nugget j, $L_{(k')}$, calculate Ω, and let $\Omega_{(k')} = \Omega$.

    4.2 Reassign data nugget j to every cluster in $\{L_1, L_2, ..., L_K\} \setminus L_{(k')}$ and calculate Ω for each of the $K - 1$ possible reassignments, $\{\Omega_1, \Omega_2, ..., \Omega_K\} \setminus \Omega_{(k')}$, where Ω_k is the total weighted within cluster sum of squares when data nugget j is assigned to cluster L_k for $k = 1, 2, ..., K$ and $k \neq (k')$.

    4.3 If $\Omega_k = min\left(\{\Omega_1, \Omega_2, ..., \Omega_K\}\right) < \Omega_{(k')}$, reassign data nugget j to cluster L_k and recalculate the cluster centers $\underset{\sim}{\mu}_1, \underset{\sim}{\mu}_2, ..., \underset{\sim}{\mu}_K$ as the mean of the centers of all the data nuggets within clusters $L_1, L_2, ..., L_K$, respectively.

5. Repeat step 4 until step 4 is completed without executing step 4.3.

The outcome of **Algorithm 3** can be improved by repeating the algorithm with multiple choices for the initial centers chosen in step 1. The clustering assignments which minimize the total weighted within cluster sum of squares would then be chosen as the clustering configuration. Further, it may take an extremely long time for the algorithm to converge. As such, a limit can be placed on the number of times step 4 is executed before the algorithm ends.

To illustrate the usefulness of **Algorithm 3** we conducted a simulation using binary data. This simulated dataset is meant to mimic a list of 300,000 patients and whether they suffer from a list of ten conditions. We separate the data into three clusters based on the set of conditions these patients suffer from. Let L_1, L_2, and L_3 represent the three clusters. Let p be the probability of having any condition. Let,

$$\underset{\sim}{\mathbf{x}} = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_{10} \end{pmatrix} \qquad \underset{\sim}{\mathbf{y}} = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_{10} \end{pmatrix} \qquad \underset{\sim}{\mathbf{z}} = \begin{pmatrix} z_1 \\ z_2 \\ \vdots \\ z_{10} \end{pmatrix}$$

where $x_i \sim Bin\,(1, 1-p)$ for $i = 1, 2, 3, 4, 5$, $x_i \sim Bin\,(1, p)$ for $i = 6, 7, 8, 9, 10$, $y_i \sim Bin\,(1, p)$ for $i = 1, 2, 3, 4, 5$, $y_i \sim Bin\,(1, 1-p)$ for $i = 6, 7, 8, 9, 10$, and $z_i \sim Bin\,(1, p)$ for $i = 1, 2, ..., 10$. L_1 is formed by sampling 100,000 observations of form $\underset{\sim}{\mathbf{x}}$, L_2 is formed by sampling 100,000 observations of form $\underset{\sim}{\mathbf{y}}$, and L_3 is formed by sampling 100,000 observations of form $\underset{\sim}{\mathbf{z}}$. These 300,000 observations are then placed together in a dataset.

Since these observations are binary, there are at most $2^{10} = 1024$ possible unique observations. Each of these observations will represent a data nugget, and the weight of the data nugget is the number of times this data nugget center appears in the dataset.

Using the Hartigan & Wong algorithm for K-means clustering and the weighted K-means clustering method given by **Algorithm 3** with $p_{\text{reassign}} = .5$, we assign each data nugget to one of three clusters. Next, for every observation in the original dataset, we append the cluster assigned to its corresponding data nugget for each method. Finally, we find the proportion of correct cluster assignments for every possible permutation of cluster assignments and choose the cluster configuration which produces the best result for each method.

We repeat this process for 100 iterations and find the mean proportion of data nuggets correctly reassigned to their proper cluster for each method. We compare the two methods for various choices of p. Three random sets of centers are used to initialize the algorithms for each iteration and the cluster configurations with the least within cluster sum of squares and weighted within cluster sum of squares for the K-means clustering method and the weighted K-means clustering method, respectively, are chosen. The simulation results are given in Table 1.

p	K-means (H-W)	Weighted K-means
0.80	0.5572	0.8274
0.82	0.5607	0.8519
0.84	0.5903	0.8738
0.86	0.6140	0.8767
0.88	0.6355	0.8735
0.90	0.6754	0.9100

Table 1: Correct Cluster Classification Simulation Results

It is clear to see that the weighted K-means algorithm outperforms the Hartigan-Wong algorithm in terms of the mean proportion of correct reclassification. This provides proper motivation to believe that using weighted K-means clustering to form clusters of data nuggets provides better results than ignoring the weights and simply using K-means clustering. As for choosing the best number of clusters, popular methods such as constructing silhouettes or calculating the gap statistic are used to detect the optimal number of clusters (**??**). These methods, and others, can be updated to include information concerning the data nugget weights.

3.2 Data Nuggets vs. Support Points

In Section 1 we mentioned another method of producing representative datasets called support points given by Mak and Joseph. The goal of data nuggets and support points are the same on the surface: to create a small dataset that represents the large dataset it comes from. That being said, the resulting representative datasets may differ greatly in terms of producing the correct quantiles corresponding to the highest percentiles of the probability distributions they are meant to represent.

This is by design in the case of support points, since they are defined as a set of M points in the dataset which has the best goodness of fit to the underlying distribution governing the dataset in terms of energy distance as defined in **?**. This definition forces support points to be chosen as observations that exist near the center of the data, ultimately forsaking observations that exist at the extreme edges of the data.

Data nuggets on the other hand are designed to avoid this problem. Since the algorithm that creates data nuggets chooses observations to delete based on how close they are to other observations, observations on the edges of the data are safer from elimination and are usually guaranteed to remain as an initial data nugget center. The fact that this observation is at the edge of the data is not lost since it will be reflected in a low weight parameter for the data nugget.

We now produce the results of a simulation which examines this difference in a 1-dimensional setting. The simulation was conducted by randomly sampling 100,000 observations from a standard normal distribution. Let this random sample of observations be $\underset{\sim}{\hat{\mathbf{z}}}$. 100 support points and 100 data nuggets are then created from $\underset{\sim}{\hat{\mathbf{z}}}$. The support points were generated using the R package "support" created by Mak (**?**). The data nuggets were generated with **Algorithm 1** by choosing $\mathbf{X} = \underset{\sim}{\hat{\mathbf{z}}}$, data nugget centers chosen to be the mean, $R = 5000$, $C = .1$, $M_{init} = 1,000$, $M = 100$, and D to be the Euclidean distance metric. The data nuggets are then ordered by their centers, least to greatest.

We then compute the quantiles corresponding to the 95^{th}, 96^{th}, 97^{th}, 98^{th} and 99^{th} percentiles for each representative dataset. The quantiles for the support points are calculated in the typical fashion; however, calculating the quantiles for the data nuggets requires a

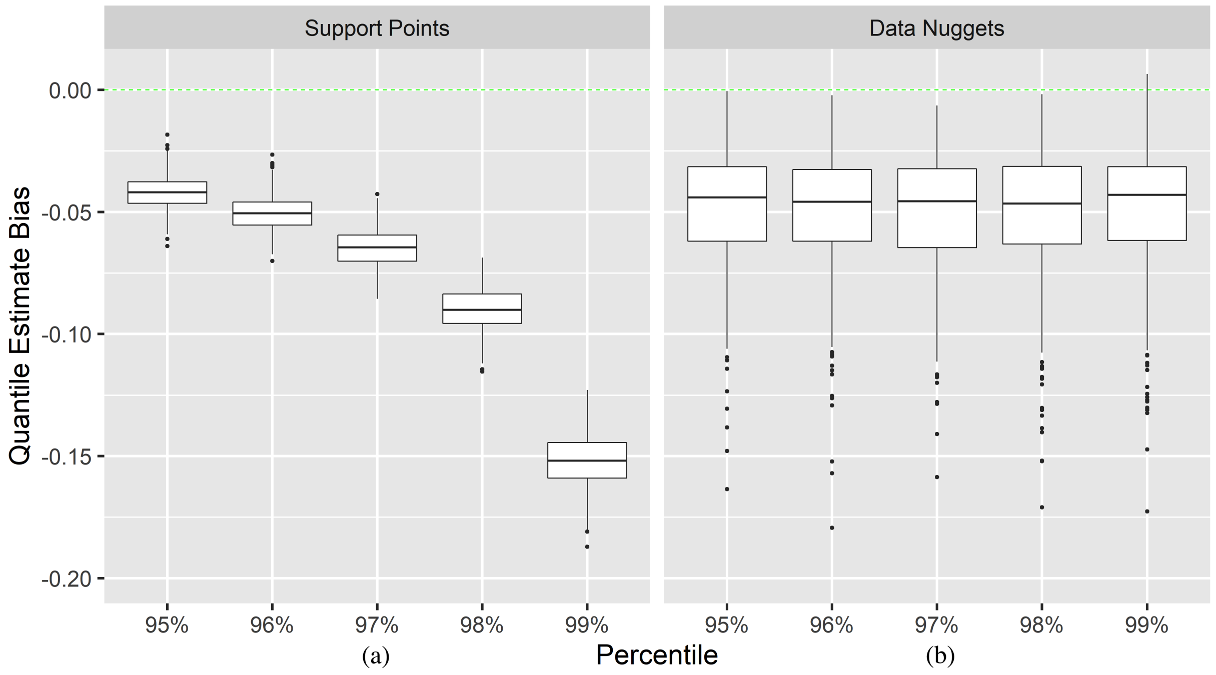


Figure 3: Quantile Bias Simulation Results

more thoughtful process.

First, a linear regression model is fit with the cumulative sums of the data nugget weights divided by 100,000 as the predictor variable and the data nugget centers as the response variable. Then, .95, .96, .97, .98, and .99 are plugged into the resulting regression equation to produce the quantiles corresponding to those percentiles for the data nuggets.

Finally, the true quantiles for a standard normal distribution are subtracted from the quantiles calculated for each method to calculate the bias of each quantiles for each method. This simulation was repeated for 500 sets of $\hat{\underset{\sim}{\mathbf{z}}}$ and Figure 3 shows the results. Figure 3 and all figures that follow were produced using the R package "ggplot2" (**?**). The box plots in Figure 3 represent the distribution of quantile estimate bias for each corresponding percentile for each method.

It is clear to see that support points perform poorly in terms of bias compared to data nuggets for calculating the quantiles at the upper tail of the normal distribution. Also note that since the bias for the quantiles given by the data nuggets is consistent across the percentiles, there may be a simple correction constant that can be applied to each quantile to eliminate the bias entirely.

3.3 Effectiveness When P is Large

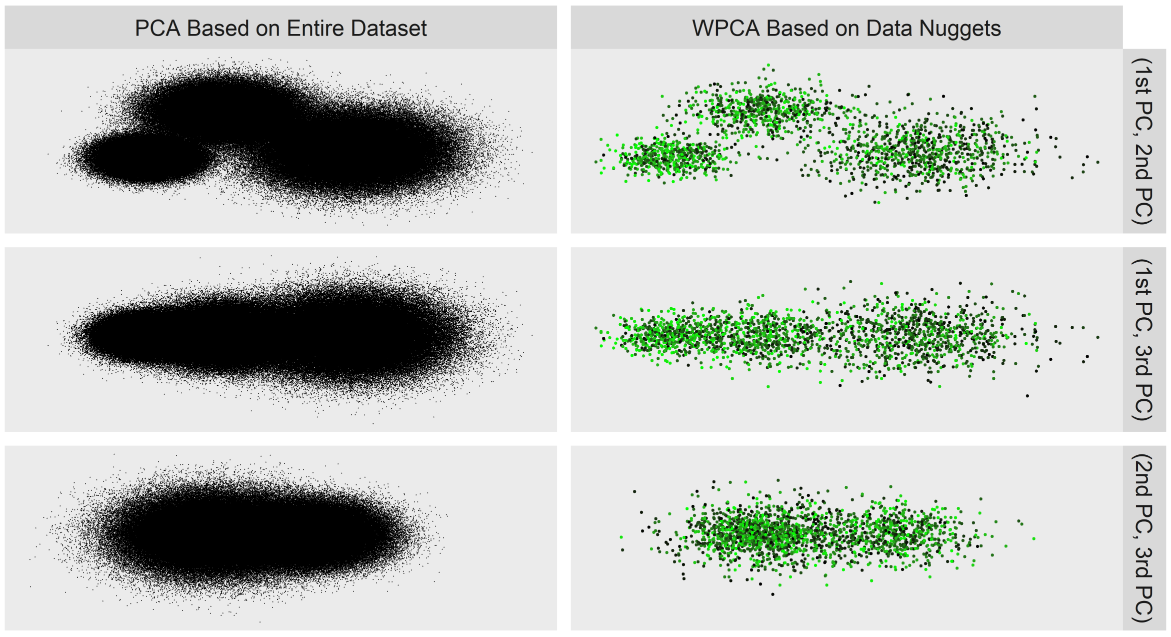
When the number of variables is large, questions can reasonably be raised about whether or not data nuggets remain effective at capturing the structure of the entire dataset. A simulation was created to examine this question.

First, the $600,000 \times 3$ data matrix $\mathbf{X_1}$ was formed by combining the observations yielded from randomly sampling 200,000 vectors from $N_3((0,0,10)', \mathbf{\Sigma_X})$, 200,000 vectors from $N_3\Big(\Big(0, \frac{6}{\sqrt{2}}, \frac{6}{\sqrt{2}}\Big)', \mathbf{\Sigma_X}\Big)$, and 200,000 vectors from $N_3\Big(\Big(\frac{10}{\sqrt{3}}, \frac{10}{\sqrt{3}}, \frac{10}{\sqrt{3}}\Big)', \mathbf{\Sigma_X}\Big)$, where:

$$\mathbf{\Sigma_X} = \begin{bmatrix} 4 & 0 & 0 \\ 0 & 2.25 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Second, the $600,000 \times 3$ data matrix $\mathbf{X}_2$ was formed by randomly sampling 600,000 vectors from $N_{197}(\mathbf{0}', \mathbb{I}_{197})$, where $\mathbf{0}$ is the vector containing only 0's. Third, the data matrix $\mathbf{X}_3$ was formed by horizontally concatenating $\mathbf{X}_1$ and $\mathbf{X}_2$ so that $\mathbf{X}_3 = [\mathbf{X}_1\mathbf{X}_2]$. Fourth, a 200×200 random rotation $\mathbf{T}$ was created by randomly sampling 200 vectors from $N_{200}(\mathbf{0}', \mathbb{I}_{200})$. Finally, the data matrix $\mathbf{X}$ was formed by applying the random rotation $\mathbf{T}$ to $\mathbf{X}_3$ so that $\mathbf{X_T} = \mathbf{T}\mathbf{X}_3$.

2,000 data nuggets were then generated with **Algorithm 1** by choosing $\mathbf{X} = \mathbf{X_T}$, data nugget centers chosen to be random, $R = 5,000$, $C = .1$, $M_{init} = 10,000$, $M = 2,000$, and D to be the Euclidean distance metric. To compare the structure of these two datasets, principal components were generated for $\mathbf{X_T}$ and weighted principal components were generated for the 2,000 data nuggets. All principal components were found using the "DN.PCA" function from the R package "datanugget". Note that the weights entered for the "DN.PCA" function when creating the principal components for the entire dataset were all equal to 1 while the weighted principal components for the data nuggets were weighted according to the weights of the data nuggets. Also note that some principal components have been multiplied by -1 to make the similarity in the structures of the data more apparent.



(a) (b)

Figure 4: PCA Plots of $\mathbf{X_T}$ vs WPCA Plots of Data Nuggets

In Figure 4, pairwise combinations of the first, second, and third principal components of the entirety of $\mathbf{X_T}$ (left column of plots) are shown beside the same pairwise combinations of the first, second, and third weighted principal components of the 2,000 data nuggets (right column of plots). The color of each data nugget corresponds to the weight of the data nugget. Lighter green indicates a large weight while darker green indicates a low weight. Clearly the data nuggets reproduce the structure of the first three principal components of $\mathbf{X_T}$ in its entirety.

4 Application

Lastly we applied this algorithm to the analysis of a high-dimensional $(1,048,575 \times 9)$ flow cytometry dataset generated following protein immunization in mice.

We begin by using **Algorithm 1**, choosing data nugget centers chosen to be the mean, $R = 5,000$, $C = .01$, $M_{init} = 10,000$, $M = 2,000$, and D to be the Euclidean distance metric to create 2,000 data nuggets. We then refine the data nuggets with **Algorithm 2**, choosing $\nu = .25$ and $N_{min} = 2$ which resulted in 3,183 data nuggets.

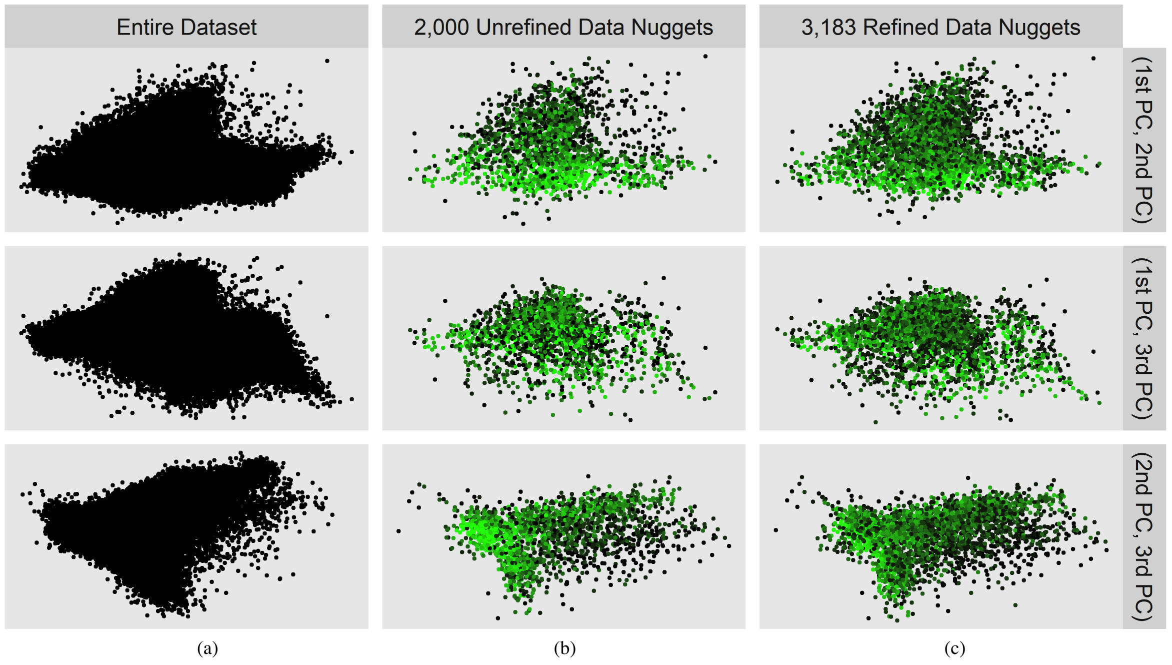


Figure 5: PCA Plots of Entire Dataset vs WPCA Plots of Data Nuggets

Pairwise combinations of the first, second, and third principal components of the entire dataset (left column of plots) are shown beside the same pairwise combinations of the first, second, and third weighted principal components of the initial 2,000 data nuggets (middle column of plots) and the refined 3,183 data nuggets (right column of plots) are given in Figure 5 for a comparison of the resulting data structures. Once again, note that the weights entered for the "DN.PCA" function when creating the principal components for the entire dataset were all equal to 1 while the principal components for the data nuggets were weighted according to the weights of the data nuggets. Also note that some of the component scores for some principal components have been multiplied by -1, swapped, and/or shifted to make the similarity in the structures of the data more apparent. Once again, the color of each data nugget corresponds to the weight of the data nugget. Lighter green indicates a large weight while darker green indicates a low weight.

Observe how the structure of the data regarding the first three principal components is moderately recovered with the original 2,000 data nuggets and strongly recovered with the 3,183 refined data nuggets. Recall that the original dataset contains over 1 million observations, so the fact that less than 1% of these observations can be chosen and still produce a relatively strong representation of the structure of the data is noteworthy.

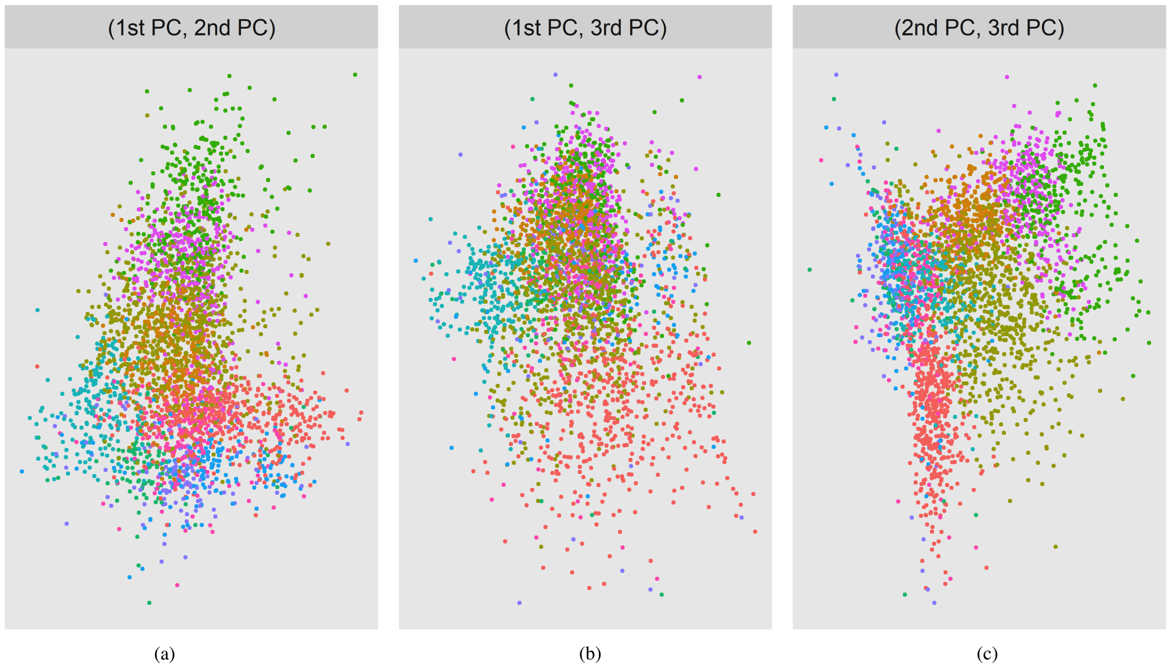


Figure 6: Weighted PCA Plots of Refined Data Nuggets After Clustering

Next, we configure the refined data nuggets into 10 clusters using **Algorithm 3** with $p_{\text{reassign}} = 1$ to perform weighted K-means clustering. We use 10 initial centers and choose the cluster configuration with the least weighted within cluster sum of squares. The pairwise combinations of the first, second, and third weighted principal components of the refined data nuggets separated into 10 clusters, each represented by a different color, is shown in Figure 6.

Finally, we created box plots for each cluster which summarize the level of expression within the cluster for each protein (measured via each component of the data nugget center) to search for whether any clusters show any visually significant levels of expression of any proteins. These box plots are given in Figure 7. These box plots could then be examined by the scientists to see if any meaningful clusters have appeared.

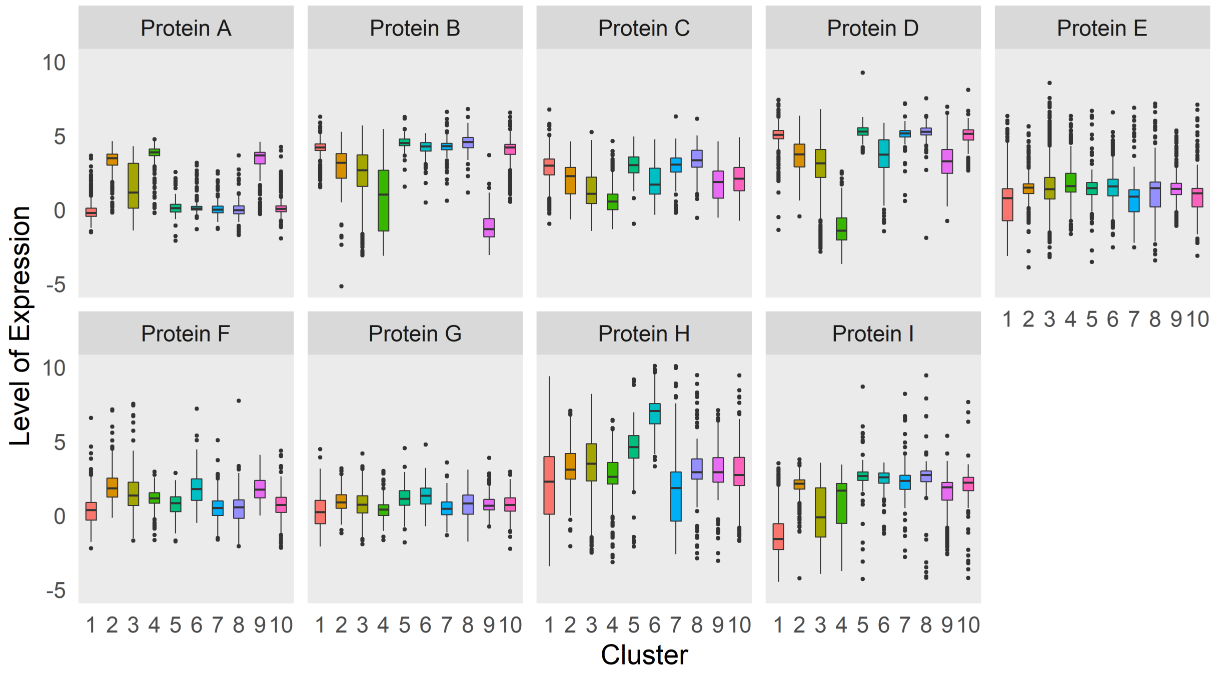


Figure 7: Levels of Expression for Each Protein and Cluster Combination

5 Conclusion

We have detailed a method for reducing "Big Data" using data nuggets. We offer a weighted K-means algorithm to cluster these data nuggets and provide simulation results which show that this algorithm outperforms the K-means clustering algorithm for data nuggets yielded from binary data. We also displayed the distinction between data nuggets and support points in the context of quantile bias at the upper tail of a normal distribution using a simulation, showing that there is a greater level of bias when these quantiles are calculated with support points. Further, we used a simulation to demonstrate that even for datasets with large P, data nuggets are capable of capturing the structure of the dataset.

The R package "datanugget" has been developed to execute the methods described in this paper. It includes functions for generating, refining, and clustering data nuggets using weighted K-means clustering. It has been submitted for publication on CRAN.

Future work could be done to show how well the data nuggets work when other main-stream statistical techniques are applied. We have already shown how well data nuggets can work when unsupervised methods such as principal components and clustering are applied. Another unsupervised method of interest that could be applied is projection pursuit (**?**). The efficacy of data nuggets could also be observed in the context of supervised methods such as logistic regression and linear regression.

In the case of logistic regression, the response for each data nugget would be the number of "successful" and "unsuccessful" observations contained in the data nugget. In the case of linear regression, the response for each data nugget would be the mean of the responses of the observations contained in the data nugget and weighted least squares regression could be applied. The weight of each data nugget (potentially combined with the variance of the response variable for each data nugget) would be used as the weight in the regression model.

An important area of improvement for this method would be to find the optimal number of data nuggets for any given sample size. Simulations involving large classified continuous datasets could also be created to determine how much better weighted K-means clustering performs compared to K-means clustering of data nuggets in a continuous setting.

Another area of interest is showing that the results of the simulation in Section 3.2 hold for higher dimensions. Work could also be done to provide a correction for the constant bias for estimating the quantiles with data nuggets. Research into asymptotic results regarding how well the probability distribution can be returned through estimation of the mean and covariance of data nuggets generated from a random sample of this probability distribution as the number of data nuggets increases to infinity would be useful as well.

SUPPLEMENTARY MATERIAL

R package for algorithms: R package "datanugget" containing functions that perform algorithms described in this paper. (tar.gz file)

Flow Cytometry Dataset used in Section 4: Flow cytometry dataset that has been masked, permuted, and had random noise added to it. (.RData file)

R Code for output: Code that produces all figures and tables found in this paper. (.R file)