Development of a Scalable Method for Creating Food Groups Using the NHANES Dataset and MapReduce

Michael R. Wyatt II
University of Delaware
Computer & Info. Sciences
Newark, DE 19716
m.r.wyatt.ii@gmail.com

Travis Johnston
University of Delaware
Computer & Info. Sciences
Newark, DE 19716
j.travis.johnston@gmail.edu

Mia Papas
University of Delaware
Behavioral Health & Nutrition
Newark, DE 19716
mpapas@udel.edu

Michela Taufer
University of Delaware
Computer & Info. Sciences
Newark, DE 19716
taufer@udel.edu

ABSTRACT

In this paper we tackle the need for meaningful food group classifications in dietary datasets such as the National Health and Nutrition Examination Survey (NAHNES) that are less subjective in nature by defining a new objective method of identifying food groups exclusively based on the food's micro- and macro-nutrient content. We first perform extensive preprocessing of the NHANES raw data to mitigate impacts of missing nutrient values, redundancies, and different food intake quantities and scales. We then utilize an unsupervised learning clustering algorithm to create food groups within the preprocessed NHANES data and identify food groups with similar nutrient content. Finally we parallelize our method to benefit from the scalable MapReduce paradigm. Our results show that our method identifies food groups with smaller diameter and larger cluster separation distances than the standard, expert-informed, method of grouping food items.

Categories and Subject Descriptors

J.3 [**Applied Computing**]: Life and medical sciences—*Consumer health, Health care information systems, Health informatics*

General Terms

Algorithms, Management, Measurement

Keywords

Clustering methods, data processing, Apache Spark, dietary data, micro- and macro-nutrients

BCB'16, October 2–5, 2016, Seattle, WA, USA.

http://dx.doi.org/10.1145/2975167.2975179.

1. INTRODUCTION

The diet of an individual or population is complex and difficult to model. The complexity comes from the large number of food items available as well as the interaction of foods in the body after consumption. Although complex, developing models for measuring the totality of the diet is critical to enhance our understanding of the influence of diet on overall health and well-being. Identifying dietary patterns, instead of focusing on single nutrients or single foods, is a method that has been used to attempt to capture the complexity of human consumption. Dietary patterns can reveal information that can be used to develop guidelines for how humans should eat to reduce the risk of disease [1].

Two main methods of identifying dietary patterns have emerged in an effort to reduce the complexity inherent in assessing dietary exposures. The first method measures how well a diet conforms to pre-established dietary guidelines. An index or score is typically derived by comparing adherence of the measured diet to that of the pre-established diet [2]. Examples of these scores include the Healthy Eating Index-2010 or the Mediterranean Diet Score [3]. This method relies heavily on the appropriateness of the score to capture the overall dietary quality of the individual or population. The second method uses a more data-driven approach such as factor or cluster analysis, examining how well individual food items are correlated or identifying clusters of individuals with similar dietary intakes. This method becomes computationally intense as the number of food items to be clustered increases. Both of these methods, whether criterion or data-driven, rely on defining pre-established food groups in order to reduce the amount of information that must be effectively analyzed. However, the food groups defined prior to analysis are subjective, typically created based on the opinion or interpretation of human experts [2, 3, 4, 5]. This leads to methods that are seldom consistent from study to study. New methods are needed in order to develop meaningful food group classifications that are less subjective in nature.

In this paper we address this need by defining a new objective method of identifying food groups based on the micro- and macro-nutrient values of the food items. Our method works under the assumption that food groups should con-

sist of food items that are highly similar in nutrient content. To this end, we first consider a well-known dietary dataset, the National Health and Nutrition Examination Survey (NHANES) dataset, and perform extensive preprocessing of the NHANES raw data. Our preprocessing ensures that our method minimizes the introduction of subjectivity into the food grouping. Specifically, all of the nutrient features used in the analysis have equal weight. We then utilize an unsupervised learning clustering algorithm in order to create food groups within the NHANES data. The algorithm is able to identify natural clusters of food items in a dataset. In other words, food items are grouped together based on the their nutrient content similarity. Finally, our method is developed with scalability in mind. The ever-increasing amount of collected data must be processed in parallel in order to maintain the ability for time efficient analysis. Our solution relies on the MapReduce paradigm for the parallel analysis of the food's nutrient values. The goal of our method is to identify high quality food groups which have smaller diameter (more internally similar) and larger cluster separation (more externally different) than the standard, expert-informed, method of grouping food items.

Our method is unique and novel as it relies on all available nutritional information of each food item and only on that nutritional information. We are not aware of any other work produced to date that has used extensive data analysis, unsupervised learning algorithms, and the MapReduce paradigm to identify food groups within NHANES-like data.

2. THE DATA AND PREPROCESSING

In this section we describe the targeted dataset and our preprocessing approach that we define and apply in order to transform the raw data into a suitable analytic format. Preprocessing is necessary to address four issues common in datasets such as NHANES.

2.1 Targeted Dataset

The National Health and Nutritional Examination Survey (NHANES) [6] dataset contains demographic, medical, and dietary data for thousands of American respondents and has been collected biennially since 1999. The Centers for Disease Control and Prevention (CDC) have made a total of seven sets of data available (1999-2012) to the public via their website.

We target the dietary dataset contained in NHANES which measures consumption for 105,263 Americans over a 13-year period. Dietary data are collected using a 24-hour dietary recall that allows participants to document every food item consumed during the past 24 hours. This method assumes that the diet of an individual can be represented by the intakes over an average 24-hour period. Data collected in 1999-2000 and 2001-2002 contain information about the food intake of participants for a single day. Collections from 2003-2012 contain information about the food intake of participants for two non-consecutive days. Every collection has a file which maps food item descriptions to an 8-digit integer food code generated by the United States Department of Agriculture (USDA); each row of the file contains a food code and description of that food item. The USDA grouping is the standard, expert-informed, method of grouping food items that we compare our method with.

Collections have a file for each day of recorded dietary intake. Every row of the file is an *entry* in our dataset and contains an identification number for the participant recording the food intake, the 8-digit food code of what the participant ate, metadata about the entry (e.g., date, time), and nutrient content of the food (also called *features*). Figure 1 shows an example of a food entry structure. There

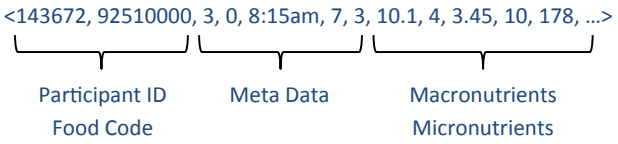


Figure 1: Example of a food entry with its food code, metadata, and features.

are up to 65 nutrient features for each food item identified across each year of dietary data collection. Forty-six features (71%; 46/65) are common to the entire NHANES dataset and thus targeted in our study. These 46 nutrient features can be split into two categories: macro-nutrients (e.g., fat, carbohydrates) and micro-nutrients (e.g., vitamins, minerals). There is an average of 15 food entries per participant and each participant can have multiple entries of the same food, that is they consume the same food multiple times a day. Additionally, the nutrient content of each entry is proportional to the weight of the entry, which means that two entries with the same food code can have different nutrient content values depending upon the amount of that food item consumed.

Food codes provide more than just a means of uniquely identifying food items. They also provide information about the standard, expert-informed, method of grouping food items. The first four digits of a food code provide the groups to which a food item belongs. Each digit places the food item in an increasingly specific subgroup. Figure 2 shows an example of how to read the food code for fruit juice drink to obtain the group which the food item belongs to [7]. The

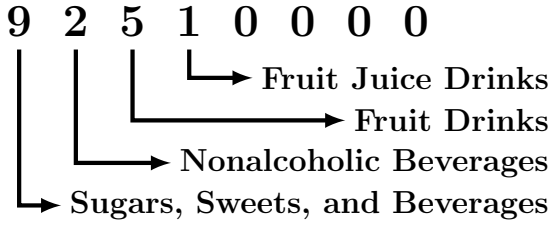


Figure 2: Example of a food code found in the NHANES dataset and how to identify the food group.

last four digits of the food code specify individual foods in the same group. For example, the food code 92510720 refers to a "fruit juice drink" because the first 4 digits (9251) place it in that category. The final 4 digits (0720) specify this food entry as "fruit punch made with fruit juice and soda."

To make the NHANES dataset usable for our analysis, it has to be transformed and processed because the current raw data suffers from four problems: (1) missing nutrient values for some food entries; (2) different weights for the same food item in different food entries; (3) redundant food entries; and (4) different nutrient features with different scales (e.g.,

Table 1: Example of data snapshot from an NHANES individual foods file.

	USDA Food Code	Description	Weight (g)	Protein (g)	Calcium (mg)	Caffeine (mg)
1:	11111000	Milk	76.25	2.4	86	
2:	11111000	Milk	122	3.84	138	
3:	11111000	Milk	518.5	16.33	586	0
4:	11111000	Milk	2.54	0.08	3	0
5:	92191100	Coffee	1.5	0.18	2	47
6:	92191100	Coffee	1.35	0.17	2	42
7:	92191100	Coffee	0.9	0.11	1	28
8:	92191100	Coffee	0.45	0.06	1	14

grams and milligrams) in a food entry. Table 1 provides a sample of the data that can be found in the individual foods file of the dataset. The table shows four (of many) entries for two different foods: whole milk (from a cow) and regular ground coffee. In Rows 1 and 2, in the caffeine column, there are missing nutrient values whereas later in Rows 3 and 4 we see that caffeine has a "0" value. Missing values and zero values are a common occurrence in the dataset and must be properly handled. Across the rows in the table, protein quantities are measured in grams whereas both calcium and caffeine are measured in milligrams. We want to be able to group these nutrients, but it is unclear whether a unit change in each of these columns has the same effect. For this reason we need to normalize the data so that they use the same metrics. Rows 1-4 have the same food code and thus provide redundancy for milk, as do Rows 5-8 for coffee. When we create our food groups, we do not need to have milk as an entry listed hundreds of times in the group, a single milk entry suffices. Last, Row 1 says 2.4 grams but Row 3 records 16.33 grams for the same food item. The discrepancy associated to the quantity of milk consumed (76.25 grams in Row 1 and 518.5 grams in Row 3) should be reconciled. To make sure that missing or zero values, redundancies, and differing scales and weights do not bias food grouping, we preprocess the data by applying a rigorous set of data cleaning steps. The preprocessing tasks are not unique to the cleaning of the NHANES dataset and our methods are adaptable to any similarly structured raw data.

2.2 Missing Values

Missing values for the 46 common nutrient features exist in 9,586 of the total 1,587,750 food entries in the NHANES dataset. A missing value exists in the comma-separated files as a blank entry or a single space entry. The naïve method for handling missing values is to replace the missing values with 0. However, using this approach introduces artificial data into the dataset. If too many missing values are filled with artificial data, the analysis becomes increasingly biased. For example, if several of the entries have many missing values, replacing the missing values with a 0 can increase the similarity between these entries and skew average values towards zero. This can cause food items which are dissimilar to then be grouped together [8].

The analysis of the 9,586 food entries with missing values indicates that nearly all the entries have food code 11000000 (i.e., "milk, human"). Nearly every entry for this food item has no information about its nutrient content. We speculate this is due to nutritional values for human milk (breast milk) being dependent on the diet of the mother. We removed food entries that have values missing for the 46 features. Due to the redundant nature of the food item entries, and the existence of at least one record for "milk, human" which does contain nutrient information, no unique food items are lost.

2.3 Dependence on Quantity of Food

A fair comparison of food items in the NHANES raw data is not possible due to the nutrient content values being scaled to the weight of the intake food item. Without accounting for size of the reported food item, foods consumed in high quantities are grouped with other foods consumed in high quantities. Groups should be built around the nutrient content of food items and not the amount of the actual reported intakes. A fair comparison of nutrient content can be achieved using the nutrient density of each food item. We scale all foods to nutrient content per one gram in order to obtain the nutrient density values. Each of the 46 nutrient features for an entry are divided by the weight (in grams) of that entry.

Some of the food entries are reported to have a weight of zero grams. With a weight of zero, the nutrient densities cannot be obtained since division by zero is an undetermined value. In the preprocessed NHANES dataset, 10,983 entries are removed from the dataset due to having a weight of zero grams. Some of the items removed due to missing weights coincide with the set of items removed due to missing values. From the 10,983 entries removed, no unique food items are ultimately removed from our dataset.

2.4 Entry Redundancy

Each food item has several entries in the dataset; however we require only one entry per food item to create food groups. We average the top five entries for each food item when sorted by reported weight. Averaging entries with a large weight avoids the rounding error present in the raw data. Table 2 shows an example of the weight, caffeine content, and calculated caffeine/gram values for several entries of 92191100 (i.e., "coffee") in our dataset. The caffeine content, like many other micro- and macro-nutrients, is rounded to the nearest integer in the raw NHANES dataset. The consequence of rounding is large differences in the calculated nutrient density values. In general, by using only entries with large reported weights, we can reduce the variance in calculated nutrient densities across food items in the dataset.

Additionally, in the raw NHANES dataset there exists a modification code for each food entry. This code indicates if a food item is prepared in a different way (e.g., added fat) and also indicates that the nutrient density values do not match with other entries of the same food with a different modification code. Table 3 shows an example of varying nutrient values for different modification codes of food item

Table 2: Example of nutrient value rounding.

USDA Food Code	Modification Code	Weight (g)	Caffeine (mg)	Caffeine Density (mg/g)
92191100	0	1.35	5	3.70
92191100	0	.23	1	4.35
92191100	0	.45	2	4.44
92191100	0	.11	0	0.00
92191100	0	1.5	5	3.33
			Standard Deviation	1.83

Table 3: Example of different nutrient values due to modification codes.

USDA Food Code	Modification Code	Weight (g)	kcal	kcal/gram
58148110	0	88.5	179	2.02
58148110	206782	22.13	29	1.31
58148110	205349	44.25	72	1.63
58148110	206181	99.56	173	1.74
			Standard Deviation	0.29

58148110 (i.e., "macaroni or pasta salad"). We remove entries with a modification code that is not 0 before averaging the density values. As a results of this last modification, a total of 39,072 food entries and 32 unique food items are removed from the dataset.

2.5 Different Scales

We standardize the data to remove the effects of nutrient features being measured on different scales. For example, fat is measured in grams while caffeine content is measured in milligrams. There are 1,000 milligrams in a gram, so the scale of the caffeine is exaggerated. A 200 milligram caffeine difference between two food items can impact distances greater than a 0.2 gram fat difference. In order to determine the best method for standardizing data, the distribution of nutrient density features across all unique entries is examined. Many of the features appeared to be either normally or log-normally distributed. As an example, Figure 3 shows the feature standardized distribution for cholesterol density. Similar distributions are obtained for the other macro- and micro-nutrients. A quantitative measure of the likelihood of a feature having a known distribution is obtained by quantile-quantile (QQ) plots [9], as seen in Figure 4. On the x-axis there are theoretical quantile values of a log-normal distribution and the y-axis is the observed quantiles for the cholesterol values. The closer the data points are to the line the better the fit (in the case to the log-normal distribution). In this case, the plotted points appear very linear indicating that the two datasets, cholesterol level and a theoretical log-normal distribution, come from a common distribution. Similar conclusions were observed for other food items.

The presence of log-normal and normal-like distributions among nutrient density features indicates that Z-Score is an applicable standardization method. Z-Score standardization converts original values into values indicating how many standard deviations above or below a value is from the mean of the original distribution. For example, Figure 5 shows the distribution of protein density before and after standardization for the protein distribution. Similar patterns are observed for the other macro- and micro-nutrients' distributions. The similarity in distribution indicates that our method of standardization has minimal effect on the shape of a distribution.

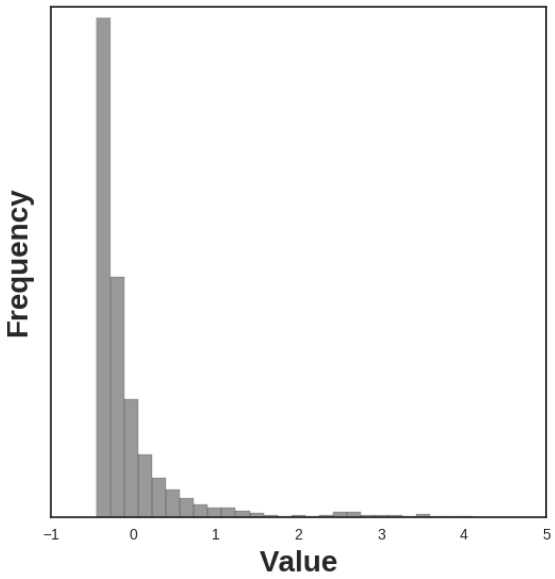


Figure 3: Distribution of cholesterol content.

2.6 Preprocessed Data

The preprocessing and cleaning of the raw data described above removes 50,056 of the 1,587,750 food entries. The number of unique food items that we are grouping is reduced from 7,494 to 7,462. Table 4 shows the number of food entries and unique foods removed for each collection of the NHANES dataset as well as the entirety of the NHANES dataset. Our method of preprocessing removes a minimal amount of food items from the original dataset (i.e., 32 unique food items lost). These removed food items represent a very small percentage (0.4%) of the original data; most of the removed food items are slightly modified versions of food items which remain in the dataset (e.g., Frozen white and yellow corn, 75216182, is removed, but canned white and yellow corn, 75216183, remains in the dataset). Preprocessed data are normalized and standardized to pro-

Table 4: Comparison of data collections before and after preprocessing and cleaning.

			Raw Data		Processed Data		Difference	
Year	**Day**	**Features**	**Entries**	**Unique Items**	**Entries**	**Unique Items**	**Entries**	**Unique Items**
1999	-	46	127840	4311	120025	4159	7815	152
2001	-	60	143004	4268	140533	4254	2471	14
2003	1	63	131164	3961	126717	3883	4447	78
2003	2	63	120871	3893	116497	3825	4374	68
2005	1	64	146940	4052	142764	3980	4176	72
2005	2	64	132151	3808	127975	3731	4176	77
2007	1	65	145703	4255	141019	4184	4684	71
2007	2	65	121341	3985	117011	3892	4330	93
2009	1	65	150991	4409	146577	4335	4414	74
2009	2	65	129141	4146	125086	4060	4055	86
2011	1	65	126503	4607	123834	4534	2669	73
2011	2	65	112101	4261	109656	4182	2445	79
All	-	46	1587750	7494	1537694	7462	50056	32

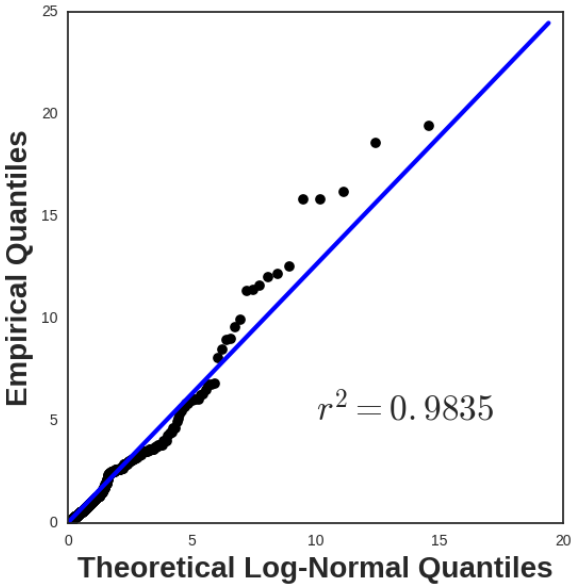


Figure 4: A QQ-Plot for cholesterol content against a log-normal distribution shows that the cholesterol distribution follows a log-normal distribution.

vide an objective comparison of nutrient values between all food items, exclusively based on the nutritional characteristics of the food items. Our data-driven grouping method presented in the next section is applied to the preprocessed NHANES dataset.

3. CLUSTERING ALGORITHM

There are many unsupervised learning techniques to cluster data. In this section we motivate our selection of the DBSCAN algorithm for clustering over simpler and more common clustering techniques such as K-means or hierarchical clustering. We also detail our implementation of DBSCAN using Apache Spark, a MapReduce framework.

3.1 Selection of Clustering Algorithm

To build a framework which groups food items based on their nutritional content, we need a clustering algorithm that takes the food item's features (i.e., the nutrient values) and groups items with similar values effectively and in a scalable way. We examine the suitability of three clustering algorithms: K-Means clustering, hierarchical clustering, and Density Based Spatial Clustering of Applications with Noise (or DBSCAN) across six desired properties. A suitable algorithm must be efficient with large datasets and be insensitive to noisy data. We prefer an algorithm which is able to distinguish noise or outliers from actual clusters. And we do not want a clustering algorithm which preferentially creates globular (roughly spherical) clusters. In addition, we do not want to impose any bias on the clustering by forcing any specific number of clusters to appear. Therefore, we need a clustering algorithm which does not require *a priori* knowledge of the number of clusters. Table 5 summarizes whether the three algorithms meet the set of desired properties [10, 11]. We are dealing with a 46-dimensional dataset (i.e., 46 macro- and micro-nutrient features common across all food items) and we expect to deal with even larger data dimensions in the future. We need an algorithm that is efficient with any N-dimensional datasets. While K-means is efficient on large datasets, the outliers from the long tails of our log-normally distributed features is problematic. In addition, its dependence on *a priori* knowledge of the number of clusters and bias towards globular clusters makes K-means unsuitable. Hierarchical clustering does not suffer from the same setbacks that K-means clustering does. However, hierarchical clustering is not a scalable technique and is therefore not suitable for our analysis. We chose to use DBSCAN since it meets all the desired characteristics of a clustering method as seen in Table 5.

3.2 Implementation in MapReduce

Sequential DBSCAN is not able to handle large amounts of data due to the memory limitations of a single compute node. When dealing with datasets containing hundreds of thousands of data points in a high-dimensional space DBSCAN must be distributed across multiple computational nodes of a cluster. We implemented DBSCAN in Apache Spark by converting the clustering problem into a problem of determining the connected components of a simple graph determine by pairwise distances. We implemented a solu-

Table 5: Comparison of common clustering algorithms in terms of six desired properties.

Algorithm \Feature	K-means	Hierarchical Clustering	DBSCAN
Efficient with large dataset	✓	×	✓
Not affected by noisy data	×	×	✓
Can distinguish outliers	×	×	✓
Can find globular clusters	✓	✓	✓
Can find non-globular clusters	×	✓	✓
Requires no knowledge of the number of clusters	×	✓	✓

tion to this problem using a MapReduce framework which allows us to find the connected components in parallel.

The MapReduce paradigm is a parallel programming model that facilitates the processing of large distributed datasets. It was originally proposed by Google to index and annotate data on the internet [12]. In this paradigm, the programmer specifies two functions: map and reduce. The map function takes as input a key and value pair, performs the map function, and outputs a list of intermediate key and value pairs that can be different from the input. The runtime system automatically groups all the values associated with the same key in a step that is called shuffle to form the input to the reduce function. The reduce function takes as input a key and a list of all values associated with that key. It then performs the reduce function on the collection of values and outputs one or more new key value pairs. Apache Spark is a software implementation of the paradigm that offers several benefits including: a more diverse set of functions beyond the traditional map, shuttle and reduce; and the in-memory operations that do not require repeated disk access. Among the additional functions, it is worth mentioning the `FlatMap` function that returns a sequence for each element in the list, and flattening the results into the original list [13] and the `Join` operator that, when called on two sets of key-value pairs with the same key but different values (e.g., (K, V) and (K, W)), returns a set of key-value pairs where the values are tuples (e.g., (K, (V, W)).

The sequential version of DBSCAN cycles through each data point (i.e., food item) and counts the number of points or food items within a distance of ϵ in the 46-D space. Two data points are neighbors if the distance between them is less than ϵ. DBSCAN identifies three type of points: core, border, and noise points. A core point is one which has more than *min_pts* neighbors. Each core is part of a cluster. A cluster is expanded by continually adding neighbors of core points to the cluster. Data points which are not core points, but have a core point neighbor, are called border points. These points are included in a cluster, but DBSCAN does not expand the cluster to its neighboring points. Non-core data points which do not have a neighboring core point are classified as noise. The algorithm cycles through each point in a dataset, applying these rules, until all points are either part of a cluster or noise. Two parameters play a key role in defining a good or bad cluster: the number of *min_pts* neighbors and the distance ϵ. The optimal selection of these parameters result in dense, well-separated groups of food items.

In the parallel version, we redefine DBSCAN to interpret its clusters as a graph of connected components. Each food item with its 46 features (i.e., 46-dimensional point) becomes a node of a graph where every core point has an edge to all of the neighbor points; each border point is connected to exactly one of its core point neighbors; and noise points have no connecting edges. By using this interpretation of the food items as nodes in the graph, clusters are created by solving the connected components problem with the algorithm in [14].

In the MapReduce-based DBSCAN, we implement a connected component solver in Spark to work with our food items. Our algorithm works as a chain of two MapReduce jobs. In the first MR job, the graph of food items is represented in a Spark resilient distributed dataset (RDD) as a key-value pair. The key is the food code and the value is a list of food items with a connected edge to the food item given by the key. A FlatMap operation converts this graph to an additional key-value pair RDD, where the key is the neighboring food code and the value is the food code of the node. A third RDD is created from the original graph RDD by replacing the list of connected food items with the minimum valued food code in that list. Each node keeps track of the minimum food code which it can reach through connected nodes.

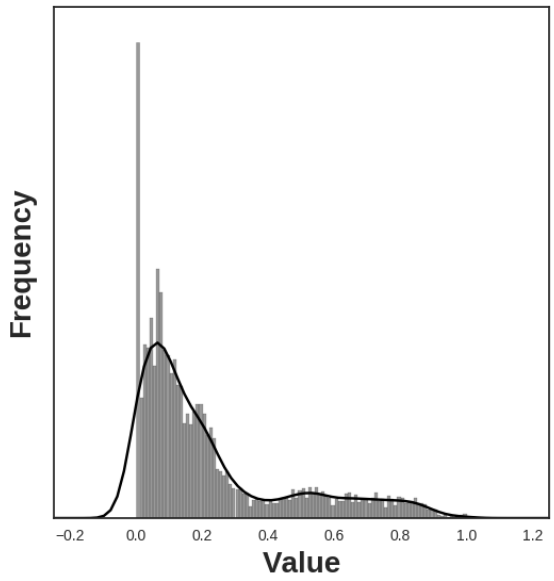
In the second MR job, a map-join job is iteratively executed; its input is the two RDDs established from the original RDD representing the graph structure. At each iteration, the two RDDs are joined on keys. This operation allows each node to see the minimum valued food code which can be reached by all its connected neighbors. The RDD which tracks the minimum valued food code reachable by each food item is updated and the iterative MR job starts again. When the values of the minimum valued food code RDD no longer change, the clustering has converged and all connected components have been found. At this stage, the algorithm exits the iterative map-join chain and returns the list of food codes and associated group.

4. GROUPING FOOD ITEMS

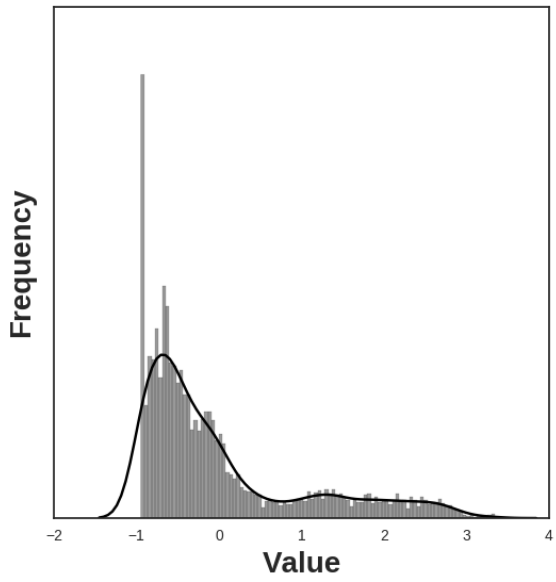
Within this section we define the specific method used to select the most suitable DBSCAN parameters and compare our data-driven grouping of food items to the standard, expert-informed, method of grouping food items. Comparisons are made in terms of the diameter of the resulting groups and group separation distances.

4.1 Selecting DBSCAN Parameters

As outlined in Section 3.2, the DBSCAN clustering algorithm is characterized by two key parameters: *min_pts* and ϵ. Together, *min_pts* and ϵ are a proxy for density. The proper selection of the two parameters substantially impacts the quality of the resulting food groups. The high dimensional nature of the data incurs the curse of dimensionality and measuring distances is not a trivial task. Common metrics such as Euclidean distance and cosine distance metrics have both strengths and weaknesses. Euclidean distance



(a) **Before**

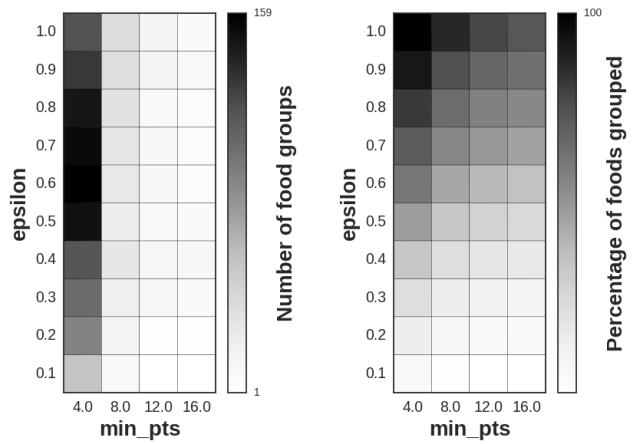


(b) **After**

Figure 5: The distribution of protein density before (a) and after (b) the Z-score standardization for the protein density.

typically performs poorly in high dimensional space while cosine similarity typically performs better [15, 16]. However, what defines high or low dimensionality is subjective to the type of data and communities using the data. Therefore, we study the impact of both the Euclidean distance and cosine similarity metrics with DBSCAN. For the latter, we transform the cosine similarity into a distance metric using: $d(i, j) := 1 - \text{CosSimilarity}(i, j)$. Using both metrics, we sweep across a range of *min_pts* and ϵ values.

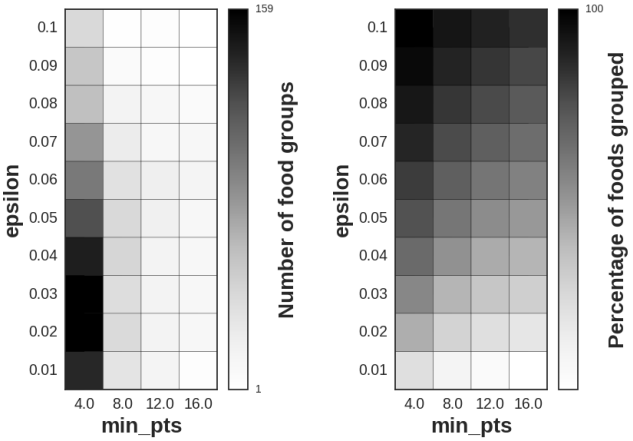
The first step involves searching for a suitable value of *min_pts*. Figures 6a and 6b show the number of resulting clusters (Figure 6a) and the percentage of points clustered (see Figure 6b) when the Euclidean distance is used. Figures 7 shows similar results when the distance metric is



(a) Number of clusters (b) Clustered items

Figure 6: Heat maps showing the number of clusters (a) and the percentage of clustered food items (b) when the Euclidean distance is used.

instead derived from cosine similarity (i.e., Figures 7a shows the number of resulting clusters and Figures 7b shows the percentage of points clustered). The figures demonstrate



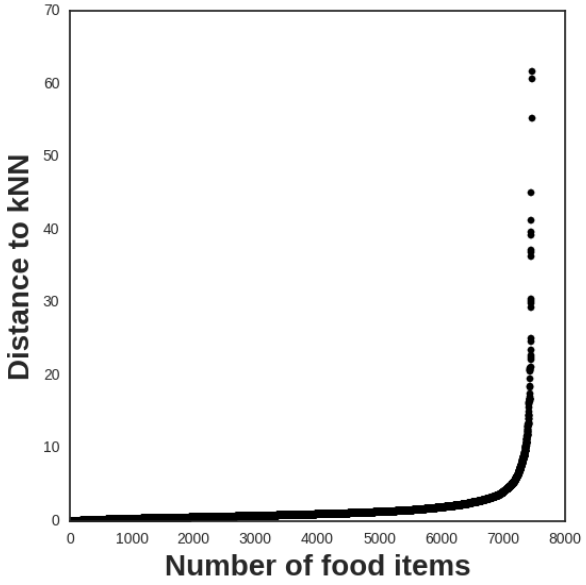
(a) Number of clusters (b) Clustered items

Figure 7: Heat maps showing the number of clusters (a) and the percentage of clustered food items (b) when the cosine similarity distance is used.

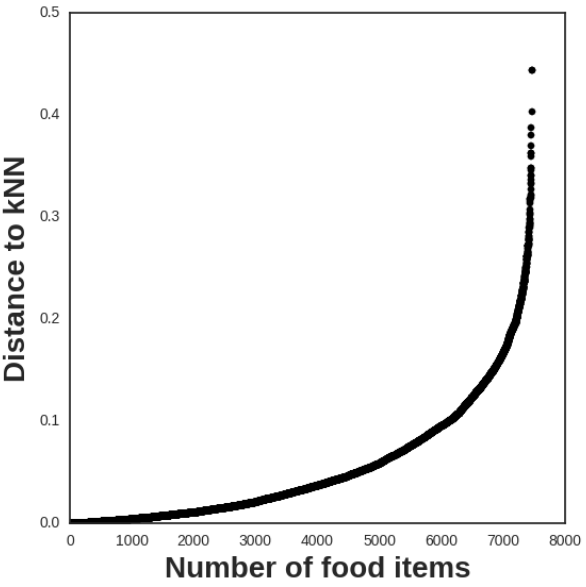
that a *min_pts* value of 4 produces a clustering that incorporates the most food items while still producing multiple clusters. Thus we use *min_pts* equal to 4 in the rest of our study.

Determining a proper ϵ value requires finding a balance between number of points clustered and the number of clusters. As the number of clustered points increases, the number of clusters decreases, and vice versa. To determine a

proper ϵ value, we plot the distance to the 4th nearest neighbor for all food items. The food items are sorted along the x-axis according to this distance. Figure 8a and Figure 8b show these plots for Euclidean and cosine distance metrics respectively. Optimal values for ϵ are typically found near



(a) Euclidean distance



(b) Cosine similarity

Figure 8: Food items sorted by distance to 4th nearest neighbor when using the Euclidean distance (a) and the cosine similarity (b). The position of the elbow indicates good candidate values for ϵ.

the elbow of the resulting curve, where the slope changes dramatically [17]. Food items with forth nearest neighbors past the elbow are border and noise points. These plots help determine an ϵ value which captures all core points and avoids noise points. For Euclidean distance, the elbow is well defined and indicates that an ϵ value between 1.0 and 3.0 is ideal. The elbow of the curve for cosine distance is less well defined, but indicates an ϵ value between 0.08 and 0.10 is ideal.

The parameter sweep for Euclidean distance do not consider ϵ values beyond 1.0. Therefore, we fix *min_pts* to 4 and examine the results of ϵ values up to 3.0. Figure 9 shows that the number of clusters quickly declines as ϵ increases. However, the number of points clustered grows as

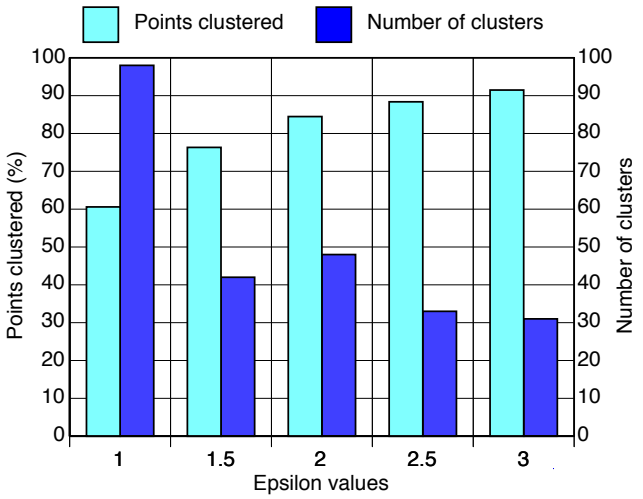


Figure 9: Percentage of points clustered and number of clusters as a function of ϵ when min_pts is 4.

ϵ increases. Based on our criteria for cluster quality (i.e., dense clusters that have significant separation), we chose an ϵ value of 1.0 for the Euclidean metric and an ϵ value of 0.08 for the cosine metric. These values maintain a balance between the number of clusters and the number of points clustered. Overall, our empirical observations indicate that the Euclidean distance metric outperforms the cosine distance metric when clustering a larger percentage of food groups into many clusters. Therefore, in the next session, we use the Euclidean distance for comparing our approach with the USDA food code grouping.

4.2 Comparison of Group Approaching

The quality of food grouping obtained with traditional approaches and our data-driven approach can be assessed in different ways. Common methods of internal evaluation, like the Dunn Index, are biased toward globular groups of food items [17] and do not capture non-globular groups. In our work we rely on three metrics of success that, when combined, provide a unique indicator of the quality of food groups. First, a significant portion of the food items should be classified into food groups (i.e., food items are not considered noise). Second, the diameter of each food group should be small. The diameter is the maximum distance between two points in the same food group. Having a smaller diameter means that there are no two foods in a food group that have dramatically different nutritional content. Finally, the distances between two food groups should be maximized. This property ensures that foods which are grouped into dif-

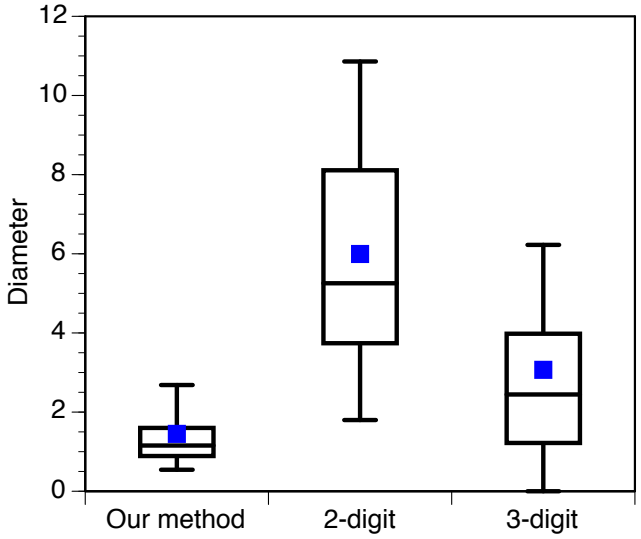
ferent food groups actually have distinctly different nutrient content.

Given this notion of quality, we compare food groups generated in three different ways. First, we cluster the foods based on their nutritional information using our DBSCAN implementation and the Euclidean distance with ϵ set to 1.0 and *min_pts* set to 4. Then, we cluster foods into groups by the first two digits of their USDA food code. For example, food code 92510000 (from Figure 2) is in food group 92 which indicates non-alcoholic beverages. Last we cluster foods into groups by the first three digits of their USDA food code. Using 3 digits (instead of 2) provides a refinement of the grouping, i.e. it subdivides each cluster into several smaller clusters. If we were to further subdivide the groups, many would have only a single food item. For this reason, we only used 2 or 3 digits. Using DBSCAN with the parameters just specified we generate 98 clusters and cluster 61% of the unique foods into groups. Using the same 61% of unique food items and the first two and three digits of the associated USDA food code, we generated 46 and 190 clusters, respectively. Figures 10a and 10b show the results on cluster diameter and cluster separation comparing these three methods.

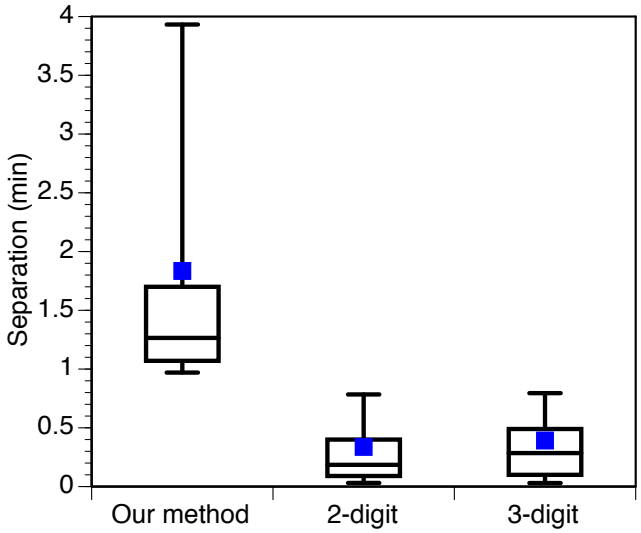
Figure 10a displays the diameters of the groups resulting from the two different clustering schemes (i.e., our approach on the left and the USDA food code, two or three digits, in the center and on the right). When interpreting these graphs, it is important to keep in mind that lower diameters are optimal. Each box plot consists of six different statistical pieces of information related to the 98, 46, and 190 groups obtained with our method, the 2-digit USDA code, and the 3-digit USDA code, respectively. The whiskers on the bottom extend from the 10th percentile to the top 90th percentile. The top, bottom, and line through the middle of the box correspond to the 75th percentile (top), 25th percentile (bottom), and 50th percentile (middle). The blue dot is the average diameter. When using DBSCAN, the average diameter is 1.45 as opposed to 5.99 and 3.07 when using the USDA food codes. The median (50th percentile) is 1.15 opposed to 5.25 and 2.44. 90% of the groups have a diameter smaller than 2.67 whereas the 2-digit method has 75% of the clusters with a diameter above 3.80 and the 3-digit method has 50% of its diameters larger than 2.44.

Figure 10b shows the separation across groups resulting from the two different clustering schemes (i.e., our approach on the left and the USDA food code groupings center and right). In this case, the higher the separation, the better. When using DBSCAN, the average separation is 1.83 as opposed to 0.34 and 0.39 when using the USDA food codes. Furthermore, 90% of the groups formed with DBSCAN are separated by at least 0.97 units. Using 2 and 3 digits of the USDA food code, 90% of the groups have a separation of at most 0.72 and 0.79 units respectively.

The smaller diameter of our groups combined with higher degrees of separations for our grouping indicates how a data-driven grouping of food items can provide a more accurate classification of food intake based on nutrition values only compared to standard, expert-informed, method of grouping food items currently used by USDA.



(a) Cluster Diameters (lower better)



(b) Cluster Separation (higher better)

Figure 10: Measurements of the quality of three types of food groups: defined by DBSCAN based off of nutrient content (left), defined by the USDA food code with two digits (center), and three digits (right).

5. CONCLUSION

In this paper we present a data-driven, scalable approach to creating food groups which are objectively measured. We provide a method of processing dietary data and an implementation of DBSCAN in a MapReduce framework suitable for processing datasets containing hundreds of thousands of records. We measure the quality of food groups based off of the nutritional similarity (closeness) of foods within a group and by the nutritional dissimilarity (separation) of foods from distinct groups. In the first case of cluster similarity, we show an improvement from a mean cluster diameter of 5.99 or 3.07 down to 1.45 units. In the second case of cluster separation (or dissimilarity) we show an improvement from an average of 0.33 or 0.39 up to 1.84. We reduce the average differences from within a single food group by a factor of at least 2 and we increase the average difference between foods in different clusters by a factor of almost 6.

Because objective and exclusively based on nutritional values, our grouping method is particularly suitable to support broader studies that target the identification of dietary patterns and the development of models for measuring the influence of specific food groups on the population's overall health and well-being.

6. ACKNOWLEDGEMENTS

This work is supported by NSF grant #1318445, NSF grant #1513025, and University of Delaware Research Foundation.

7. REFERENCES

[1] A. K. Kant. Dietary Patterns and Health Outcomes. *J Am Diet Assoc*, 104(4):615–635, Apr 2004.

[2] J. Reedy, E. Wirfalt, A. Flood, P. N. Mitrou, S. M. Krebs-Smith, V. Kipnis, D. Midthune, M. Leitzmann, A. Hollenbeck, A. Schatzkin, and A. F. Subar. Comparing 3 Dietary Pattern Methods–Cluster Analysis, Factor Analysis, and Index Analysis–With Colorectal Cancer Risk: The NIH-AARP Diet and Health Study. *Am. J. Epidemiol.*, 171(4):479–487, Feb 2010.

[3] C. Niclis, M. Román, A. Osello, A. Eynard, and M. del Pilar Diaz. Traditional Dietary Pattern Increases Risk of Prostate Cancer in Argentina: Results of a Multilevel Modeling and Bias Analysis from a Case-Control Study. *Journal of Cancer Epidemiology*, 2015, 2015.

[4] A. Betoko, M. A. Charles, R. Hankard, A. Forhan, M. Bonet, M. J. Saurel-Cubizolles, B. Heude, B. de Lauzon-Guillain, M. A. Charles, A. Forhan, M. de Agostini, B. Heude, P. Ducimetiere, M. Kaminski, M. J. Saurel-Cubizolles, P. Dargent, X. Fritel, B. Larroque, N. Lelong, L. Marchand, C. Nabet, I. Annesi-Maesano, R. Slama, V. Goua, R. Hankard, G. Magnin, O. Thiebaugeorges, M. Schweitzer, B. Foliguet, and N. Job-Spira. Infant Feeding Patterns Over the First Year of Life: Influence of Family Characteristics. *Eur J Clin Nutr*, 67(6):631–637, Jun 2013.

[5] S. Robinson, L. Marriott, J. Poole, S. Crozier, S. Borland, W. Lawrence, C. Law, K. Godfrey, C. Cooper, and H. Inskip. Dietary Patterns in Infancy: the Importance of Maternal and Family Influences on Feeding Practice. *Br. J. Nutr.*, 98(5):1029–1037, Nov 2007.

[6] NHANES-National Health and Nutrition Examination Survey. `http://www.cdc.gov/nchs/nhanes/index.htm`. Accessed: 2016-05-12.

[7] Food and Nutrient Database for Dietary Studies, 5.0. http://www.ars.usda.gov/ba/bhnrc/fsrg, 2012. Accessed: 2016-05-12.

[8] Yongsong Qin, Shichao Zhang, Xiaofeng Zhu, Jilian Zhang, and Chengqi Zhang. Semi-parametric Optimization for Missing Data Imputation. *Applied Intelligence*, 27(1):79–88, 2007.

[9] B. Das and S. I. Resnick. QQ Plots, Random Sets and Data from a Heavy Tailed Distribution. *Stochastic Models*, 24(1):103–132, 2008.

[10] Maryam Bakhshi, Mohammad-Reza Feizi-Derakhshi, and Elnaz Zafarani. Review and Comparison between Clustering Algorithms with Duplicate Entities Detection Purpose. *International Journal of Computer Science & Emerging Technologies*, 3(3), 2012.

[11] Narendra Sharma, Aman Bajpai, and Ratnesh Litoriya. Comparison the Various Clustering Algorithms of Weka Tools. *Int J Emerg Tech Adv Eng*, 2(5), 2012.

[12] Jeffrey Dean and Sanjay Ghemawat. MapReduce: Simplified Data Processing on Large Clusters. *Commun. ACM*, 51(1):107–113, 2008.

[13] Matei Zaharia, Patrick Wendell, Andy Konwinski, and Holden Karau. *Learning Spark*. O'Reilly Media, Inc, 2015.

[14] Jonathan Cohen. Graph Twiddling in a MapReduce World. *IEEE Comput. in Science & Engin.*, 11:29–41, 2009.

[15] Levent Ertöz, Michael Steinbach, and Vipin Kumar. Finding Clusters of Different Sizes, Shapes, and Densities in Noisy, High Dimensional Data. In *Proceedings of Second SIAM International Conference on Data Mining*, 2003.

[16] Eamonn Keogh and Abdullah Mueen. *Encyclopedia of Machine Learning*, chapter Curse of Dimensionality, pages 257–258. Springer US, Boston, MA, 2010.

[17] Pang-Ning Tan, Michael Steinbach, and Vipin Kumar. *Introduction to Data Mining, (First Edition)*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2005.

[18] Matei Zaharia, Mosharaf Chowdhury, Michael J. Franklin, Scott Shenker, and Ion Stoica. Spark: Cluster Computing with Working Sets. In *Proceedings of the 2nd USENIX Conference on Hot Topics in Cloud Computing*, HotCloud'10, pages 10–10, Berkeley, CA, USA, 2010. USENIX Association.