Journal of Statistical Software

February 2018, Volume 83, Issue 13. doi: 10.18637/jss.v083.i13

kamila: Clustering Mixed-Type Data in R and Hadoop

Alexander H. Foss
University at Buffalo

Marianthi Markatou
University at Buffalo

Abstract

In this paper we discuss the challenge of equitably combining continuous (quantitative) and categorical (qualitative) variables for the purpose of cluster analysis. Existing techniques require strong parametric assumptions, or difficult-to-specify tuning parameters. We describe the **kamila** package, which includes a weighted k-means approach to clustering mixed-type data, a method for estimating weights for mixed-type data (Modha-Spangler weighting), and an additional semiparametric method recently proposed in the literature (KAMILA). We include a discussion of strategies for estimating the number of clusters in the data, and describe the implementation of one such method in the current R package. Background and usage of these clustering methods are presented. We then show how the KAMILA algorithm can be adapted to a map-reduce framework, and implement the resulting algorithm using **Hadoop** for clustering very large mixed-type data sets.

Keywords: clustering, **Hadoop**, kernel density estimator, mixed data, mixed-type data, mixture model, R, semiparametric, unsupervised learning.

1. Introduction

Cluster analysis, also referred to as unsupervised learning, comprises a set of techniques that seek to identify structure in a data set without reference to a known set of labeled "training" data vectors (Hastie, Tibshirani, and Friedman 2009). In this paper we introduce software for clustering data consisting of mixed continuous and categorical variables. Although there are various existing approaches for clustering mixed-type data, they suffer from significant drawbacks including loss of information due to discretization, arbitrary weighting of continuous versus categorical variables (e.g., as in dummy coding or the similarity metric of Gower 1971), or strong parametric assumptions (e.g., as in parametric mixture models). We review these existing approaches in Section 2, and in Section 3 discuss an often overlooked technique

introduced by Modha and Spangler (2003) for estimating appropriate weights for combining continuous and categorical variables. In Section 4 we describe the KAMILA (k-means for mixed large data) algorithm and underlying data model, and in Section 5 we introduce the R (R Core Team 2017) package **kamila** (Foss and Markatou 2018) implementing a weighted k-means algorithm, Modha-Spangler weighting, and the KAMILA algorithm for clustering mixed-type data. Package **kamila** is available from the Comprehensive R Archive Network (CRAN) at `https://CRAN.R-project.org/package=kamila`. In Section 6 we introduce software for implementing KAMILA on a distributed file system using **Hadoop** (Apache Software Foundation 2016a). Section 7 offers conclusions, while Appendix A discusses the **Hadoop** implementation of the KAMILA algorithm.

2. Related work

Clustering techniques can generally be divided into hierarchical methods and partitioning methods. Hierarchical methods involve constructing a series of nested partitions of the data set; these can either involve iteratively splitting a cluster from the previous step (divisive techniques) or iteratively merging two clusters from the previous step (agglomerative techniques). The `diana` function in R package **cluster** (Maechler, Rousseeuw, Struyf, Hubert, and Hornik 2015) implements divisive clustering. The function `agnes` in package **cluster** (Maechler *et al.* 2015) and the function `hclust` implemented in each of the packages **stats** (R Core Team 2017), **fastcluster** (Müllner 2013), and **flashClust** (Langfelder and Horvath 2012) are commonly used functions for agglomerative clustering. A compromise between agglomerative and divisive clustering is developed by Chipman and Tibshirani (2006), and implemented in R package **hybridHclust** of Chipman, Tibshirani, and Hastie (2015).

Partitioning methods generally involve a single split of the data set into mutually exclusive and exhaustive clusters. A well-known example are the k-means algorithms (Hartigan and Wong 1979; MacQueen 1967) or alternatives such as the k-medoids algorithms, e.g., the PAM (partitioning around medoids) algorithm of Kaufman and Rousseeuw (1990). The Hartigan-Wong, Lloyd, and MacQueen k-means algorithms are implemented by the `kmeans` function in the **stats** package (R Core Team 2017). Kernel k-means uses kernels to project the data into a non-linear feature space before applying k-means, and is implemented in the `kkmeans` function in the **kernlab** package (Karatzoglou, Smola, Hornik, and Zeileis 2004). The function `kcca` in package **flexclust** (Leisch 2006) implements generalizations of k-means using arbitrary centroid statistics and distance measures. Trimmed k-means, which trims a specified proportion of extreme values in each cluster, is implemented in packages **trimcluster** (Hennig 2012) and **tclust** (Fritz, García-Escudero, and Mayo-Iscar 2012). The `pam` function in package **cluster** (Maechler *et al.* 2015) implements k-medoids. The `clara` function in the same package implements a computationally efficient version of k-medoids in which the partitioning steps are executed on sub-sampled data and then applied to the entire data set. This efficient algorithmic structure makes `clara` well suited for large data sets that can be stored in RAM. At the end of this section we discuss existing software for data sets too large to store on a single compute node.

The finite mixture model (e.g., Lindsay 1995; McLachlan and Basford 1988; McLachlan and Peel 2000; Titterington, Smith, and Makov 1985) is another partitioning method in which each cluster is assumed to follow some parametric distribution, the parameters of which are then typically estimated using the EM (expectation-maximization) algorithm (Dempster,

Laird, and Rubin 1977). Various finite mixture models are implemented in R. Finite Gaussian mixture models can be fit using the packages **EMCluster** (Chen and Maitra 2015), **mclust** (Fraley, Raftery, Murphy, and Scrucca 2012), and **mixture** (Browne, ElSherbiny, and McNicholas 2015); multinomial or gamma mixtures can be fit using package **mixtools** (Benaglia, Chauveau, Hunter, and Young 2009), and mixtures of skew-normals can be fit using package **mixsmsn** (Prates, Cabral, and Lachos 2013). For an overview on packages available on CRAN for cluster analysis and mixture models see the corresponding CRAN Task View (Leisch and Grün 2017).

Although there is an abundance of clustering techniques designed for a single data type, fewer exist for handling mixed-type data. In fact, many of the most common approaches to clustering mixed-type data involve imperfect usages of techniques designed for a single data type. One common strategy is to first dummy code the categorical variables, and then apply some technique designed for continuous data, such as k-means. For a categorical variable X with ℓ distinct categorical levels, the dummy coding step generally involves creating ℓ distinct $0 - c$ indicator variables, one for each of the levels, where 0 indicates absence and $c \in \mathbb{R}$ denotes presence of that particular level in X. A problem arises in the selection of c; an excessively large c will overemphasize the categorical variables over the continuous variables in the final clusters, whereas an excessively small c will under-emphasize the categorical variables. As demonstrated in Foss, Markatou, Ray, and Heching (2016), the selection of c is a difficult problem without a clear solution; even in very simple mixed-type data sets the ideal choice of c can fluctuate based on the number of variables used in the analysis and the degree of separation between the underlying clusters in the data. The common approach of arbitrarily setting $c = 1$ is by no means an acceptable solution to the mixed-type data clustering problem. Another common approach of standardizing all variables (in particular, each dummy-coded variable) to unit variance is also inadequate; it amounts to an equally arbitrary choice of $c_j \approx 1/\sqrt{p_j(1-p_j)}$, where p_j is the frequency of observations with that categorical level present in the j-th dummy-coded variable. This has the effect of up-weighting categories the further they deviate from $p_j = 0.5$. While it is trivial to imagine an application where this could be appropriate, it is equally trivial to imagine scenarios where the opposite effect is desired (i.e., two near-majority categories of interest, with a small proportion of rare categories of minimal interest). See Section 5.1 for an illustration of the difficulty in selecting appropriate weights.

A second common strategy for clustering mixed-type data is to use a distance metric compatible with mixed data, such as Gower's distance (Gower 1971), and then use a clustering method that depends only on the distances between the data points (e.g., `agnes`, `pam`, or various other functions in the **cluster** package). However, each variable in Gower's distance must be assigned a user-specified weight determining its relative contribution to the distance, which presents essentially the same dilemma as the choice of c in the dummy coding approach above. A poor choice of weights will result in certain variables being over- or under-emphasized, and in particular, it is unclear how to properly weigh the continuous variables relative to the categorical variables for a given data set. In Table 2 of Foss *et al.* (2016), a range of weighting strategies for Gower's distance and dummy coding were compared in a simulation study using three variables (one continuous, two categorical). The continuous weight was fixed at 1.0 while the categorical weights spanned a broad range of values. Two models were used to generate data; one in which the continuous variable contained more useful information regarding cluster structure, and one in which the categorical variables contained more useful

information. Although the top performing method in this simulation (KAMILA; no choice of weights required as described in Section 4) achieved adjusted Rand index (ARI; Hubert and Arabie 1985) scores of 0.99 in both conditions, weighting strategies for Gower's distance and dummy coding could only perform well in one condition at a time. For example, the best score for Gower's distance in condition 1 (ARI = 0.984) used categorical weights of 0.1, but could only achieve an ARI of 0.688 in condition 2. The best score for Gower's distance in condition 2 (ARI = 0.849) used categorical weights of 0.4, but could only achieve an ARI of 0.872 in condition 1. Similar trade-offs were apparent for dummy coding. We note that the optimal weight could fluctuate unpredictably based on features such as the number of variables and the number of categorical levels; the best weights mentioned above do not generalize to other data conditions.

Numerous other existing strategies for clustering mixed-type data involve this same problem of requiring a user-specified weight determining the relative contribution of continuous versus categorical variables. The k-prototypes method of Huang (1998) uses a distance metric equal to the squared Euclidean distance between the continuous components plus the user-specified constant γ times the matching distance between the categorical components (Huang 1998, p. 291, Equation 9). The package **clustMixType** (Szepannek 2016) implements k-prototypes clustering in R. In Azzalini and Menardi (2014) (with associated package **pdfCluster**), a novel density-based clustering scheme is proposed for continuous data, and a possible extension to mixed-type data using a distance such as Gower's distance is mentioned briefly. The method of Friedman and Meulman (2004) uses a Gower-like distance metric, and introduces an interesting strategy for selecting variable weights, but explicitly relies on the assumption that setting each weight to $1/p$ is equivalent to giving the same influence to all attributes regardless of data type (here p denotes the number of variables of any type). In Hennig and Liao (2013), a weighting scheme is proposed for mixed-type data that takes into account the number of levels in each categorical variable (and optionally the number of observations in each categorical level). The method does not take into account other crucial aspects of the data that influence the adequate choice of weights, such as number and quality of the variables. This leads to this method performing worse than other methods for mixed-type data (see Foss *et al.* (2016) for more extensive discussion and examples). As described above and discussed in Foss *et al.* (2016), the proper choice of weights fluctuates in a data-dependent manner, rendering any fixed choice of weights unable to balance continuous and categorical components in any general sense.

The clustering technique of Modha and Spangler (2003) offers a potential solution to this dilemma, and is described in Section 3. The algorithm is structured similarly to k-prototypes, using a weighted combination of squared Euclidean distance and cosine distance for the continuous and categorical variables, respectively. The weighting between continuous and categorical variables is identified through a brute-force search. In various conditions Modha-Spangler weighting performs well. Figure 4 in Foss *et al.* (2016) depicts simulation results suggesting that Modha-Spangler weighting performs poorly relative to a competing method (KAMILA clustering, described in Section 4) when clusters are poorly separated in the continuous variables but well-separated in the categorical. Furthermore, unlike finite mixture models, this method is unable to adjust the relative contribution of individual variables of the same type. To our knowledge, the method of Modha & Spangler has not previously been implemented in R, nor in any other statistical software package.

Finite mixture modeling is another technique that does not require user-specified weights

for the continuous versus categorical variable contribution. For mixed-type data, a popular model is the joint normal-multinomial mixture model (Hunt and Jorgensen 2011). Finite mixture models are often able to achieve a favorable balance between continuous and categorical variables when their parametric assumptions are reasonably accurate; however, they often perform poorly when their parametric assumptions are strongly violated. For examples of this behavior, see Foss *et al.* (2016), in which the performance of mixture models is studied in a series of Monte Carlo studies that systematically vary the severity of parametric violations. Specifically, Foss *et al.* (2016) show that the normality assumption is particularly vulnerable to cluster distributions that are skewed as well as heavy-tailed distributions. Normal-multinomial mixture models are implemented in the R packages **clustMD** (McParland 2015), **fpc** (Hennig 2015a), and **Rmixmod** (Langrognet, Lebret, Poli, and Iovleff 2016).

In a manner similar to finite mixture models, the KAMILA method of Foss *et al.* (2016) is able to achieve a favorable balance between continuous and categorical variables without requiring user-specified weights. In order to decrease the susceptibility to violations of parametric assumptions, the continuous components are modeled using a general class of elliptical distributions. Categorical variables are modeled as mixtures of multinomial random variables, thus not requiring dummy coding. Unlike the method of Modha and Spangler (2003), KAMILA does not require a brute-force search to identify an appropriate balance between continuous and categorical variables. We describe KAMILA in detail in Section 4.

Existing clustering software for very large data sets relies heavily on methods designed for continuous data only, and on k-means clustering in particular; they are thus vulnerable to the drawbacks enumerated above when used with mixed-type data. See the results of the simulation in Section 5.1 for an illustration of the limitations of using k-means for mixed-type data, and see Foss *et al.* (2016) for additional results and discussion of the limitations. The Mahout project (Apache Software Foundation 2016c) implements k-means, fuzzy k-means, and streaming k-means, as well as a spectral clustering algorithm that involves running k-means on eigenvectors of the graph Laplacian of the original data. The In-Memory Statistics for **Hadoop** Suite (SAS Corporation 2016) implements both k-means and DBSCAN, a clustering technique which also requires the choice of a suitable distance-based metric. The Microsoft Azure Machine Learning Studio (Microsoft Corporation 2016) and the **ScaleR** package (Revolution Analytics 2016b) both rely on the standard k-means algorithm for clustering data sets. The R package **pmclust** (Chen and Ostrouchov 2016), built on the **pbdMPI** framework (Ostrouchov, Chen, Schmidt, and Patel 2012), implements Gaussian mixture modeling for continuous variables on large computing clusters using the single program/multiple data (SPMD) programming style. In Section 6 we introduce an alternative clustering strategy for large mixed-type data sets that implements the KAMILA clustering algorithm in **Hadoop** streaming mode. In addition to achieving a superior balance between continuous and categorical variables compared to standard k-means approaches, our method can drastically reduce storage and computational requirements by not requiring dummy coding of variables. For example, the airline data set analyzed in Section 6 would require over 25 times the number of variables if its categorical variables were dummy-coded.

3. Modha-Spangler clustering

The work of Modha and Spangler (2003) presents a general framework for clustering data sets with multiple data types, although it focuses primarily on the special case of mixed

continuous and categorical variables. Let $\mathbf{x}_i = (\mathbf{F}_{(i,1)}^\top, \mathbf{F}_{(i,2)}^\top, \ldots, \mathbf{F}_{(i,m)}^\top)^\top$ denote the i-th data vector with m different data types and $i = 1, 2, \ldots, N$. Modha and Spangler (2003) defines a weighted distortion measure for mixed-type data as $D^\alpha(\mathbf{x}_1, \mathbf{x}_2) = \sum_{\ell=1}^m \alpha_\ell D_\ell(\mathbf{F}_{(1,\ell)}, \mathbf{F}_{(2,\ell)})$, where each of the m data types is assigned its own distance metric D_ℓ and a corresponding weight in $\alpha = (\alpha_1, \alpha_2, \ldots, \alpha_m)$, $\alpha_\ell \geq 0$, and $\sum_{\ell=1}^m \alpha_\ell = 1$.

For a fixed weighting α, the proposed algorithm in Modha and Spangler (2003) seeks to partition the data vectors into the k partitions $\{\pi_u^*\}_{u=1}^k$ such that

$$\{\pi_u^*\}_{u=1}^k = \underset{\{\pi_u\}_{u=1}^k}{\arg\min} \left(\sum_{u=1}^k \sum_{\mathbf{x} \in \pi_u} D^\alpha(\mathbf{x}, \mathbf{c}_u) \right),$$

where $\mathbf{c}_u$ denotes the centroid corresponding to the u-th cluster, whose definition depends on the particular choices of distance metrics D_ℓ for each data type. Modha and Spangler (2003) use a method analogous to the k-means algorithm to approximate the optimal partition $\{\pi_u^*\}_{u=1}^k$.

In order to select the optimal weights α^*, Modha and Spangler (2003) define the average within-cluster distortion for the ℓ-th data type as $W_\ell(\alpha) = \sum_{u=1}^k \sum_{\mathbf{x} \in \pi_u^*(\alpha)} D_\ell(\mathbf{F}_\ell, \mathbf{c}_{(u,\ell)}^*(\alpha))$, and the average between-cluster distortion as $B_\ell(\alpha) = \sum_{i=1}^N D_\ell(\mathbf{F}_{(i,\ell)}, \bar{\mathbf{c}}_\ell) - W_\ell(\alpha)$, where $\bar{\mathbf{c}}_\ell$ denotes the centroid of the ℓ-th data type taken across all N data vectors.

Finally, assuming no missing data, the method of Modha and Spangler (2003) aims to identify the weighting vector α^* such that

$$\alpha^* = \underset{\alpha}{\arg\min} \left\{ \frac{W_1(\alpha)}{B_1(\alpha)} \times \frac{W_2(\alpha)}{B_2(\alpha)} \times \cdots \times \frac{W_m(\alpha)}{B_m(\alpha)} \right\}.$$

The weighting is identified through a brute-force search: the algorithm is run repeatedly for a grid of the possible weightings α with selection as described above. We focus primarily on the case where $m = 2$, with the first data type consisting of continuous variables and the second of 0–1 dummy-coded categorical variables. In this case, Modha and Spangler (2003) use squared Euclidean distance for D_1 and cosine distance for D_2.

In the current R package we provide a flexible implementation of the Modha-Spangler method in which the underlying clustering technique can be specified by the user. We devolve the issues of initialization and stopping rules to the underlying clustering technique used. We leave for future investigation the interesting question regarding how optimal settings for initialization and stopping rules are altered when moving from a simple application of a clustering method to the Modha-Spangler approach. The granularity of the brute-force search over α_1 can be specified by the user, with the default of ten equal subdivisions of $[0, 1]$.

4. KAMILA clustering

4.1. Model

As described in Foss *et al.* (2016), we assume that our data set consists of N independent and identically distributed observations of a $(P+Q)$-dimensional vector of random variables $(\mathbf{V}^\top, \mathbf{W}^\top)^\top$ that follow a finite mixture distribution with G components, where $\mathbf{V}$ is a P-dimensional vector of continuous random variables and $\mathbf{W}$ is a vector of Q categorical random

variables, and where the q-th element of $\mathbf{W}$ has L_q categorical levels denoted 1, 2, ..., L_q, with $q = 1, 2, \ldots, Q$. The vectors $\mathbf{V}$ and $\mathbf{W}$ may be dependent, but under the local independence assumption (e.g., see Hennig and Liao 2013; Hunt and Jorgensen 2011), $\mathbf{V}$ and $\mathbf{W}$ are independent within any particular cluster.

Given membership in the g-th cluster, we model $\mathbf{V}$ as a vector following a finite mixture of elliptical distributions with individual component density functions $f_{\mathbf{V},g}(\mathbf{v}; \mu_g, \Sigma_g)$, where g indexes cluster membership, μ_g denotes the g-th centroid, and Σ_g the g-th scaling matrix. The specific form of the density function $f_{\mathbf{V},g}$ is described below. Given membership in the g-th cluster, we model $\mathbf{W}$ as a vector following a finite mixture of multinomials with individual component probability mass functions $f_{\mathbf{W},g}(\mathbf{w}) = \prod_{q=1}^{Q} m(w_q; \theta_{gq})$, where $m(\cdot;\cdot)$ is the multinomial probability mass function, and θ_{gq} is the multinomial parameter vector for the g-th component of the q-th categorical variable. Given cluster membership in the g-th cluster, under the local independence assumption, the joint density of $(\mathbf{V}^\top, \mathbf{W}^\top)^\top$ is

$$f_{\mathbf{V},\mathbf{W},g}(\mathbf{v}, \mathbf{w};\ \mu_g, \Sigma_g, \theta_{gq}) = f_{\mathbf{V},g}(\mathbf{v}; \mu_g, \Sigma_g) \prod_{q=1}^{Q} m(w_q; \theta_{gq}),$$

with the overall density unconditional on cluster membership given by

$$f_{\mathbf{V},\mathbf{W}}(\mathbf{v}, \mathbf{w}) = \sum_{g=1}^{G} \pi_g f_{\mathbf{V},\mathbf{W},g}(\mathbf{v}, \mathbf{w};\ \mu_g, \Sigma_g, \theta_{gq}), \tag{1}$$

where π_g denotes the prior probability of observing the g-th cluster.

4.2. Radial kernel density estimation

The Proposition 2 in Foss *et al.* (2016) states that if $\mathbf{X} \in \mathbb{R}^p$ follows a spherically symmetric distribution with center μ, then

$$f_{\mathbf{X}}(\mathbf{x}) = \frac{f_R(r)\,\Gamma(\frac{p}{2}+1)}{p r^{p-1} \pi^{p/2}}, \tag{2}$$

where $r = \sqrt{(\mathbf{x}-\mu)^\top(\mathbf{x}-\mu)}$, $R = \sqrt{(\mathbf{X}-\mu)^\top(\mathbf{X}-\mu)}$, and f_R is the probability density of R. We proceed to construct $\hat{f}_R$ using a (univariate) kernel density estimation scheme, which is then substituted into (2) in place of f_R. Note that $\mathbf{X}$ corresponds to the vector $\mathbf{V}$ above *within a particular cluster*, and that using a scaling matrix Σ_g this result can be extended to elliptical distributions. The `kamila` function currently uses Σ_g equal to the identity matrix; future work will investigate how best to extend KAMILA clustering to allow for more flexible specifications, such as $\Sigma_g = \Sigma_{g'}$ for all g, g', i.e., all scaling matrices being equal across clusters, and $\Sigma_g \neq \Sigma_{g'}$ for all $g \neq g'$.

This univariate kernel density estimation scheme avoids the drawbacks of a multivariate kernel density estimator: namely that multivariate kernel density estimation is computationally expensive, and tends to over-fit data points (Scott 1992, Chapter 7).

4.3. Algorithm description

KAMILA proceeds by estimating the unknown parameters of (1) via an iterative process similar to the EM algorithm, as shown in Algorithm 1 of Foss *et al.* (2016). At the t-th

iteration of the algorithm, let $\hat{\boldsymbol{\mu}}_g^{(t)}$ denote the estimator of the centroid of population g, and let $\hat{\boldsymbol{\theta}}_{gq}^{(t)}$ denote the estimator of the parameters of the multinomial distribution corresponding to the q-th discrete random variable drawn from population g.

These parameter estimates can be initialized through random uniform draws, draws from the observed data points, or with initial centroids selected through some other mechanism (e.g., a preliminary clustering round). Using centroids identified through previous rounds of clustering may be appropriate in certain circumstances, but in general, replacing a small sub-problem (i.e., initialization) with the entire original problem (clustering) seems needlessly complex. Any additional computational resources may be better spent on extending the number of initializations and iterations of KAMILA rather than spending it on an entirely separate clustering algorithm. We have found that initializing $\hat{\boldsymbol{\mu}}_g^{(0)}$ for each $g = 1, 2, \ldots, G$ with random draws from the observed continuous data vectors appears to offer a modest advantage over random draws from a uniform distribution with marginal ranges equal to the sample ranges of the data. Initializing $\hat{\boldsymbol{\theta}}_{gq}^{(0)}$ for each g and each $q = 1, 2, \ldots, L_q$ is done with a draw from a Dirichlet distribution (Kotz, Balakrishnan, and Johnson 2000) with shape parameters all equal to one, i.e., a uniform draw from the simplex in $\mathbb{R}^{L_q}$.

The estimation procedure proceeds iteratively, with each iteration consisting of two broad steps: the partition and the estimation step. The partition step assigns each observation to a cluster, and the estimation step re-estimates the parameters of interest using the new cluster memberships.

Given $\hat{\boldsymbol{\mu}}_g^{(t)}$ and $\hat{\boldsymbol{\theta}}_{gq}^{(t)}$ at the t-th iteration, the Euclidean distance from observation i to each of the $\hat{\boldsymbol{\mu}}_g^{(t)}$'s is $d_{ig}^{(t)} = \sqrt{\sum_{p=1}^{P}[(v_{ip} - \hat{\mu}_{gp}^{(t)})]^2}$. The minimum distance is then calculated for the i-th observation as $r_i^{(t)} = \min_g(d_{ig}^{(t)})$. The kernel density estimate of the minimum distances is constructed as

$$\hat{f}_R^{(t)}(r) = \frac{1}{Nh^{(t)}} \sum_{\ell=1}^{N} k\left(\frac{r - r_\ell^{(t)}}{h^{(t)}}\right), \tag{3}$$

where $k(\cdot)$ is a kernel function and $h^{(t)}$ is the corresponding bandwidth at iteration t. We currently use the Gaussian kernel, with bandwidth $h = 0.9An^{-1/5}$, where $A = \min(\hat{\sigma}, \hat{q}/1.34)$, $\hat{\sigma}$ is the sample standard deviation, and $\hat{q}$ is the sample interquartile range (Silverman 1986, p. 48, Equation 3.31). The function $\hat{f}_R^{(t)}$ is used to construct $\hat{f}_{\mathbf{V}}^{(t)}$ as shown in Section 4.2.

We assume that the Q categorical variables are independent within a given population g (i.e., local independence), and calculate the probability of observing the i-th vector of categorical variables given population membership as $c_{ig}^{(t)} = \prod_{q=1}^{Q} \mathrm{m}(w_{iq};\ \hat{\boldsymbol{\theta}}_{gq}^{(t)})$, where $\mathrm{m}(\cdot;\cdot)$ is the multinomial probability mass function.

We assign the i-th object to the population g that maximizes the function

$$H_i^{(t)}(g) = \log\left[\hat{f}_{\mathbf{V}}^{(t)}(d_{ig}^{(t)})\right] + \log\left[c_{ig}^{(t)}\right]. \tag{4}$$

In each iteration t, the latest partition of the N observations is used to calculate $\hat{\mu}_{gp}^{(t+1)}$ and $\hat{\boldsymbol{\theta}}_{gq}^{(t+1)}$ for all g, p, and q. If $\Omega_g^{(t)}$ denotes the set of indices of observations assigned to population g at iteration t, we can then calculate the parameter estimates by

$$\hat{\boldsymbol{\mu}}_g^{(t+1)} = \frac{1}{\left|\Omega_g^{(t)}\right|} \sum_{i \in \Omega_g^{(t)}} \mathbf{v}_i,$$

$$\hat{\theta}_{gq\ell}^{(t+1)} = \frac{1}{\left|\Omega_g^{(t)}\right|} \sum_{i \in \Omega_g^{(t)}} I\{w_{iq} = \ell\},$$

where $I\{\cdot\}$ denotes the indicator function and $|A|$ denotes the cardinality of the set A.

The **kamila** R package uses the straightforward rule of stopping a run when group membership remains unchanged from one iteration to the next. When running KAMILA with particularly large data sets, e.g., in a map-reduce framework (see Section 6), this stopping rule requires the storage and comparison of cluster memberships for two consecutive iterations at a time which can be computationally expensive; a stopping rule which avoids this computational cost uses the quantities

$$\epsilon_{\text{con}} = \sum_{g=1}^{G} \sum_{p=1}^{P} \left| \hat{\mu}_{g,p}^{(t)} - \hat{\mu}_{g,p}^{(t-1)} \right|^k, \quad \epsilon_{\text{cat}} = \sum_{g=1}^{G} \sum_{q=1}^{Q} \sum_{\ell=1}^{L_q} \left| \hat{\theta}_{g,q,\ell}^{(t)} - \hat{\theta}_{g,q,\ell}^{(t-1)} \right|^k, \tag{5}$$

and stops when both are less than some chosen threshold(s). If membership remains unchanged from one iteration to the next, then ϵ_{con} and ϵ_{cat} will both be zero. Thus, selecting very low thresholds will be similar to the initially proposed stopping rule. We have found that using $k = 1$ and thresholds of around 0.01 or lower yield satisfactory results with data sets with variables numbering in the dozens.

After all runs have ended, we select the best partitioning based on an objective function. Equation 4 above suggests selecting the partition that maximizes $\sum_{i=1}^{N} \max_g \{H_i^{(\text{final})}(g)\}$ over all runs. An alternative objective function, similar in structure to the Modha-Spangler objective described in Section 2, is to minimize the product of the categorical negative log-likelihood and the within- to between-cluster distance ratio of the continuous variables, that is, selecting the partition that minimizes $Q_{\text{con}} \times \left[-\log c_{ig}^{(t)} \right]$. The quantity Q_{con} is defined as $W_{\text{con}}/B_{\text{con}}$, where $W_{\text{con}} = \sum_{i=1}^{N} \|\mathbf{x}_i - \hat{\boldsymbol{\mu}}_{z(i)}^{(\text{final})}\|_2$, $z : \{1, 2, \ldots, N\} \to \{1, 2, \ldots, G\}$ is a function that maps the observed data point index to its assigned cluster index, and $B_{\text{con}} = T_{\text{con}} - W_{\text{con}}$, where T_{con} is the total sum of squares $\sum_{i=1}^{N} \|\mathbf{x}_i - \hat{\boldsymbol{\mu}}\|_2$ with $\hat{\boldsymbol{\mu}}$ denoting the sample mean taken across all continuous variables. The latter objective function is currently the default used in the **kamila** package, although future work will address in greater detail the best possible objective functions that can be used for KAMILA clustering.

5. The R package kamila

The **kamila** package provides the three clustering functions `wkmeans`, a simple wrapper for the `stats::kmeans` function modified to handle mixed-type data; `kamila`, described in detail in Section 4, and `gmsClust`, a flexible version of Modha-Spangler clustering described in Section 3 that can be modified to use any base clustering method and objective function. The `gmsClust` function defaults to using `wkmeans` with dummy coding of categorical variables and squared Euclidean distance. Only the `kamila` and `gmsClust` functions are intended to be used routinely for clustering; `wkmeans` is included primarily to be used as the default clustering technique of `gmsClust`, while also serving to illustrate fundamental challenges associated with using manual weights for clustering mixed-type data.

One heuristic approach to clustering mixed data is to use k-means with dummy coded categorical variables, implemented here with the `wkmeans` function. While the default of 0−1

dummy coding is most common, this choice of weights is not optimal in any general sense; unlike the use of dummy coding in a regression context, the choice of weights can yield completely different clustering results. Generally speaking, the higher the scalar c used in 0–c coding, the greater influence the categorical variables will have on the resulting clustering. In the `wkmeans` function, the `conWeight` parameter specifies the relative weighting of both the continuous and categorical variables. The `conWeight` parameter must be an element of $[0, 1]$; the continuous variables are each multiplied by `conWeight`, while the 0−1 dummy coded categorical variables are each multiplied by (1−`conWeight`). The "best" choice of weight in this weighted k-means approach can change dramatically based on the characteristics of the data set, and is generally difficult to specify (Foss *et al.* 2016). In the next section, we illustrate this difficulty by showing the performance of two choices of weights on two different data sets.

Selecting the optimal clustering technique remains a challenge, in part due to the fact that the optimal clustering technique is highly dependent upon the data set and a user's analytic objectives (Hennig 2015b). Expected characteristics of the clusters can guide the selection of clustering technique. If the clusters are well-approximated by the normal distribution, then the normal-multinomial model is recommended. If the clusters depart from normality, however (e.g., due to skewness or heavy tails), then a model relying on normality should be avoided (Foss *et al.* 2016) in favor of KAMILA or the Modha-Spangler technique. KAMILA is particularly well-suited to handle large data sets (hence the acronym), both in terms of algorithmic efficiency and its ability to identify useful latent structure. Depending on the scenario, Modha-Spangler may be more appropriate if the sample size is small, particularly if the underlying clusters are not well separated (well-separated clusters tend to be identified easily by any method). In particular, KAMILA (and other techniques relying on multinomial mixture models) can suffer if the sample size is small compared to the number of levels in the categorical variables (e.g., sample sizes of 100–250 with variables containing 10+ categorical levels and overlapping clusters). Modha-Spangler often is more robust in this type of scenario with categorical sparsity (small sample size relative to the number of categorical levels). The `kamila` function incorporates a categorical smoother which may be used to reduce the ill-effects of categorical sparsity; if a user suspects that categorical sparsity is adversely affecting performance the parameter `catBw` may be increased to 0.1 or 0.2; a trade-off is that an increased bandwidth when there is no categorical sparsity will reduce the impact of categorical variables.

The selection of the number of clusters remains a fundamental challenge in clustering (see, e.g., Cooper and Milligan 1988; Hennig and Lin 2015; Milligan and Cooper 1985; Tibshirani and Walther 2005). Recognizing that no method is superior in all circumstances, we include an implementation of the prediction strength method of Tibshirani and Walther (2005) in the `kamila` function. Strengths and weaknesses of the prediction strength method have been discussed in the general case (Hennig and Lin 2015; Tibshirani and Walther 2005) as well as in the context of KAMILA clustering by Foss *et al.* (2016, Section 3.3). The prediction strength method tends to favor a small number of clusters, although this depends on a user-specified threshold. While a threshold of 0.8−0.9 is recommended in Tibshirani and Walther (2005) for well-separated data (default in the `kamila` function is 0.8), it is less clear what threshold should be used if greater cluster overlap is expected. A lower threshold will, *ceteris paribus*, yield an equal or larger number of clusters. In Foss *et al.* (2016) a threshold of 0.6 yields useful results in a particular application, and Hennig and Lin (2015) suggests using a simulation-based approach in the spirit of Tibshirani, Walther, and Hastie (2001) to calibrate

the selection process. In any of the clustering functions in the **kamila** package, the number of clusters can also be manually selected by the user.

Theoretical analysis of the run-time of k-means analysis has proven surprisingly difficult (e.g., see Arthur, Manthey, and Röglin 2011), to say nothing of more complex algorithms such as Modha-Spangler and finite mixture models. However, in typical applied conditions these algorithms seem to consistently display linear run-times with respect to sample size (e.g., Figure 7 in Foss *et al.* 2016). The brute-force optimization of the Modha-Spangler algorithm yields a linear run-time with a larger slope than k-means and KAMILA. This discrepancy is increased in our R implementations of the Modha-Spangler (`gmsClust`) and KAMILA (`kamila`) algorithms, due to the fact that all of the computationally intensive portions of `kamila` are written in C++.

5.1. Basic usage and simulation

We begin by illustrating the basic usage and performance of the methods in the **kamila** package. In order to demonstrate the properties of the methods, we first use the `genMixedData` function to create a simple data generation function. This function allows the creation of mixed-type data sets in which we carefully control the degree of cluster separation/overlap separately in the continuous and categorical variables. We parameterize the overlap between two clusters as the overlapping area under their densities, with zero corresponding to complete separation and one corresponding to complete overlap; `genMixedData` specifies univariate overlap for each variable separately. The parameter `nConVar` controls the number of continuous variables, `nConWithErr` controls how many of these variables have an overlap of level `conErrLev`, while the remaining continuous variables have overlap of 0.01. The categorical variables are controlled analogously with the parameters `nCatVar`, `nCatWithErr`, and `catErrLev`. The parameter `popProportions` is a vector of probabilities controlling the relative frequency of observations drawn from each cluster. Currently only two-cluster data sets are supported, as controlling overlap between clusters in k-cluster simulated data sets is less straightforward (e.g., see Maitra and Melnykov 2010). We note that simulated mixed-type data with more intricate dependency structure between variables might be generated using the **PoisBinOrdNor** package (Amatya and Demirtas 2015; Demirtas, Hu, and Allozi 2016) to simulate data separately for each cluster.

```
R> suppressMessages(library("kamila"))
R> suppressMessages(library("mclust"))
R> ari <- adjustedRandIndex
R> genSimpleData <- function(conErrLev, catErrLev) {
+    tmpDat <- genMixedData(sampSize = 200, nConVar = 2, nCatVar = 2,
+      nCatLevels = 4, nConWithErr = 2, nCatWithErr = 2,
+      popProportions = c(.5, .5), conErrLev = conErrLev,
+      catErrLev = catErrLev)
+    tmpDat <- within(tmpDat, conVars <- as.data.frame(scale(conVars)))
+    tmpDat <- within(tmpDat, catVarsFac <- as.data.frame(lapply(
+      as.data.frame(catVars), factor)))
+    within(tmpDat, catVarsDum <- as.data.frame(dummyCodeFactorDf(
+      catVarsFac)))
+  }
```

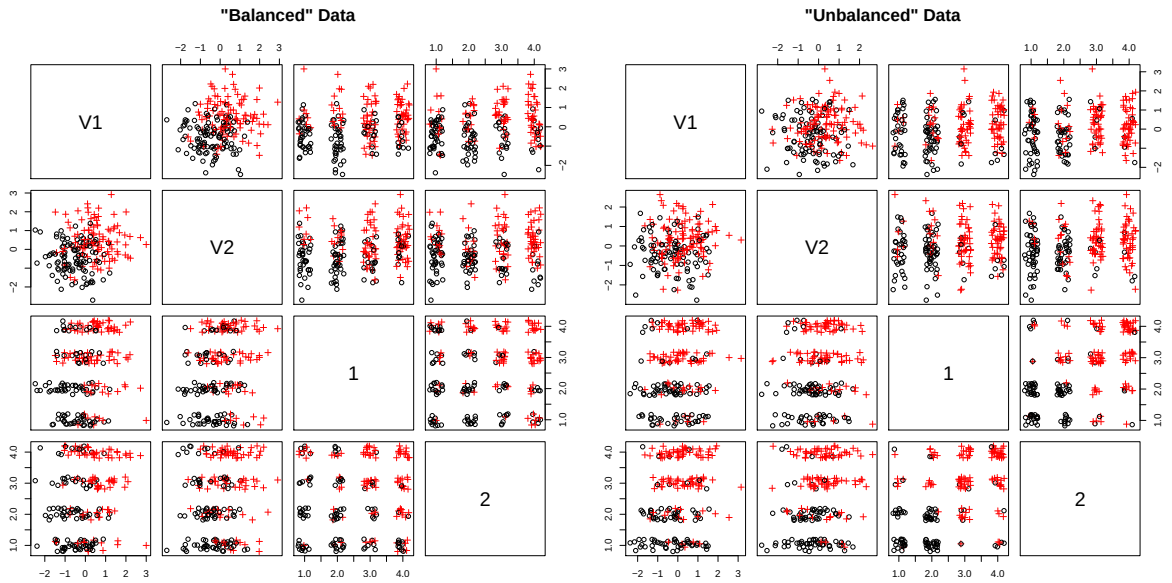

Figure 1: A pairwise scatterplot of the "Balanced" data structure (left), in which both continuous and categorical variables contain useful information regarding the cluster structure and the "Unbalanced" data structure (right), in which the categorical variables contain the most useful information regarding the cluster structure. Black circles denote cluster 1, while red pluses denote cluster 2. Variables 3 and 4 correspond to the categorical variables, whose levels have been arbitrarily numbered one through four, and jittered to avoid over-plotting.

The balanced and unbalanced data sets illustrated in Figure 1 are constructed using:

```
R> set.seed(1)
R> with(data = genSimpleData(conErrLev = 0.5, catErrLev = 0.5),
+    expr = pairs(cbind(conVars, jitter(catVars)), pch = c(1, 3)[trueID],
+    col = trueID, main = "\"Balanced\" Data"))
R> set.seed(2)
R> with(data = genSimpleData(conErrLev = 0.7, catErrLev = 0.3),
+    expr = pairs(cbind(conVars, jitter(catVars)), pch = c(1, 3)[trueID],
+    col = trueID, main = "\"Unbalanced\" Data"))
```

We focus on two conditions; first, one in which both variable types have approximately comparable cluster overlap (e.g., Figure 1, left) and next, a data structure in which the continuous variables have substantially more overlap compared to the categoricals (i.e., the categorical variables are more useful for purposes of clustering, Figure 1, right). We compare the performance of `wkmeans` with two weightings, Modha-Spangler and KAMILA clustering.

```
R> set.seed(3)
R> nrep <- 25
R> res1 <- replicate(n = nrep, expr = {
+    balData <- genSimpleData(conErrLev = 0.5, catErrLev = 0.5)
+    kmWgt5_balData <- wkmeans(conData = balData$conVars,
+      catData = balData$catVarsDum, conWeight = 0.5, nclust = 2)
```

```
+    kmWgt2_balData <- wkmeans(conData = balData$conVars,
+      catData = balData$catVarsDum, conWeight = 0.2, nclust = 2)
+    gms_balData <- gmsClust(conData = balData$conVars,
+      catData = balData$catVarsDum, nclust = 2)
+    kam_balData <- kamila(conVar = balData$conVars,
+      catFactor = balData$catVarsFac, numClust = 2, numInit = 10)
+    unbalData <- genSimpleData(conErrLev = 0.7, catErrLev = 0.3)
+    kmWgt5_unbalData <- wkmeans(conData = unbalData$conVars,
+      catData = unbalData$catVarsDum, conWeight = 0.5, nclust = 2)
+    kmWgt2_unbalData <- wkmeans(conData = unbalData$conVars,
+      catData = unbalData$catVarsDum, conWeight = 0.2, nclust = 2)
+    gms_unbalData <- gmsClust(conData = unbalData$conVars,
+      catData = unbalData$catVarsDum, nclust = 2)
+    kam_unbalData <- kamila(conVar = unbalData$conVars,
+      catFactor = unbalData$catVarsFac, numClust = 2, numInit = 10)
+    c(ari(kmWgt5_balData$cluster, balData$trueID),
+      ari(kmWgt2_balData$cluster, balData$trueID),
+      ari(gms_balData$results$cluster, balData$trueID),
+      ari(kam_balData$finalMemb, balData$trueID),
+      ari(kmWgt5_unbalData$cluster, unbalData$trueID),
+      ari(kmWgt2_unbalData$cluster, unbalData$trueID),
+      ari(gms_unbalData$results$cluster, unbalData$trueID),
+      ari(kam_unbalData$finalMemb, unbalData$trueID))})
R> rownames(res1) <- rep(c("W5/5", "W2/8", "MS", "KAM"), times = 2
R> boxplot(t(res1), ylab = "Adjusted Rand Index",
+    col = rep(c(0, 8), c(4, 4)))
```

Results are shown in Figure 2, with performance measured using the adjusted Rand index (ARI; Hubert and Arabie 1985). Although `wkmeans` with equal weighting (0.5 for both variable types) outperforms an alternative weighting (0.2 and 0.8 for the continuous and categorical variables, respectively) in the balanced data condition, this performance benefit is reversed in the unbalanced condition. In both cases, Modha-Spangler and KAMILA clustering perform equal or better than both `wkmeans` weightings. KAMILA appears to outperform Modha-Spangler in the unbalanced data condition, which matches the findings of Foss *et al.* (2016), namely that KAMILA tends to outperform Modha-Spangler weighting when categorical variables are more informative than continuous.

5.2. Byar data analysis

We illustrate our R package **kamila** on a real data set consisting of fifteen variables measured on 475 patients with prostate cancer. We cluster the data set with `gmsClust` with a k-medoids algorithm specified as the base clustering algorithm, and with `kamila`.

We begin by dropping the height, weight, patient ID, and outcome variables, and converting categorical variables from integers to factors.

```
R> data("Byar", package = "clustMD")
R> Byar$Serum.prostatic.acid.phosphatase <- log(
+    Byar$Serum.prostatic.acid.phosphatase)
```

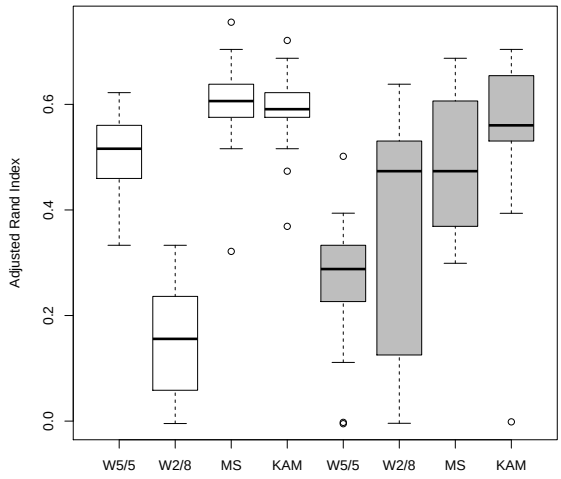

Figure 2: Performance of k-means, Modha-Spangler, and KAMILA clustering algorithms on two different data structures. White boxplots denote results from a balanced data set in which both the continuous and categorical variables contain useful information regarding cluster structure, and gray boxplots denote results from an unbalanced data set in which the categorical variables are more useful than the continuous. "W5/5" denotes the k-means algorithm with a weight of 0.5 applied to both continuous and categorical variables during clustering, and "W2/8" denotes weights of 0.2 and 0.8 applied to the continuous and categorical variables, respectively. MS denotes the Modha-Spangler algorithm, and KAM denotes the KAMILA algorithm.

```
R> conInd <- c(5, 6, 8:11)
R> conVars <- Byar[, conInd]
R> summary(conVars)
```

```
 Systolic.Blood.pressure Diastolic.blood.pressure Serum.haemoglobin
 Min.   : 8.00           Min.   : 4.000           Min.   : 59.0
 1st Qu.:13.00           1st Qu.: 7.000           1st Qu.:122.5
 Median :14.00           Median : 8.000           Median :137.0
 Mean   :14.38           Mean   : 8.158           Mean   :134.2
 3rd Qu.:16.00           3rd Qu.: 9.000           3rd Qu.:147.0
 Max.   :30.00           Max.   :18.000           Max.   :182.0
 Size.of.primary.tumour Index.of.tumour.stage.and.histolic.grade
 Min.   : 0.00          Min.   : 5.0
 1st Qu.: 5.00          1st Qu.: 9.0
 Median :10.00          Median :10.0
 Mean   :14.29          Mean   :10.3
 3rd Qu.:21.00          3rd Qu.:11.0
 Max.   :69.00          Max.   :15.0
 Serum.prostatic.acid.phosphatase
 Min.   :0.000
 1st Qu.:1.609
 Median :1.946
```

```
 Mean   :2.639
 3rd Qu.:3.384
 Max.   :9.210

R> catVars <- Byar[, -c(1:2, conInd, 14, 15)]
R> catVars[] <- lapply(catVars, factor)
R> summary(catVars)

 Performance.rating Cardiovascular.disease.history
 0:428              0:268
 1: 32              1:207
 2: 13
 3:  2

 Electrocardiogram.code Bone.metastases Stage
 0:161                  0:398           3:273
 1: 23                  1: 77           4:202
 2: 50
 3: 25
 4:145
 5: 70
 6:  1
```

We use the **kamila** package to cluster the data using Modha-Spangler clustering. In the Modha-Spangler optimization framework, the optimal choices of distance metric and clustering method depend on the data set at hand and the user's clustering goals; we do not wish to endorse one particular method over others. Thus, we allow the user to input any custom distance metric and clustering algorithm into our Modha-Spangler implementation, and we illustrate this functionality here. Instead of the default k-means algorithm, we use the Modha-Spangler framework to optimize the k-medoids algorithm PAM (partitioning around medoids; Kaufman and Rousseeuw 1990) using Gower's distance (Gower 1971). This requires using continuous variables that have been normalized to the range $[0, 1]$ by subtracting the minimum value and dividing by the range.

```
R> rangeStandardize <- function(x) {
+    (x - min(x)) / diff(range(x))
+  }
R> conVars <- as.data.frame(lapply(conVars, rangeStandardize))
```

We then write functions implementing the L1 distance for continuous variables and matching distance for categorical (the two distances required by Gower's distance). These functions must accept as input two rows of a data frame and return a scalar distance measure. Note that the default behavior of a data frame of factors differs substantially if it contains one versus multiple variables; using `as.integer` to convert factors before comparison avoids comparing incompatible types.

```
R> L1Dist <- function(v1, v2) {
+    sum(abs(v1 - v2))
```

```
+  }
R> matchingDist <- function(v1, v2) {
+    sum(as.integer(v1) != as.integer(v2))
+  }
```

Finally, we write a wrapper for the `pam` function from the **cluster** package, using the `daisy` function to implement Gower's distance. This function will be an input argument to the `gmsClust` function, and must have the method signature below (including optional arguments to be passed to the clustering function). It must return a list containing at least the three named slots `cluster` containing an integer vector denoting cluster membership of the data points; `conCenters` containing a data frame with each row denoting continuous coordinates of a centroid; and `catCenters` containing a data frame with each row denoting categorical coordinates of a centroid.

```
R> library("cluster")
R> pammix <- function(conData, catData, conWeight, nclust, ...) {
+    conData <- as.data.frame(conData)
+    catData <- as.data.frame(catData)
+    distMat <- daisy(x = cbind(conData, catData), metric = "gower",
+      weights = rep(c(conWeight, 1 - conWeight),
+        times = c(ncol(conData), ncol(catData))))
+    clustRes <- pam(x = distMat, k = nclust, diss = TRUE, ...)
+    return(list(cluster = clustRes$clustering,
+      conCenters = conData[clustRes$id.med, , drop = FALSE],
+      catCenters = catData[clustRes$id.med, , drop = FALSE]))
+  }
```

We now run Modha-Spangler clustering using PAM as the base clustering function with Gower's distance, specifying three clusters. Note that the distance function used in the `clustFun` argument is independent of the distance functions used to construct the objective function (i.e., arguments to `conDist` and `catDist`).

```
R> set.seed(4)
R> msRes <- gmsClust(conData = conVars, catData = catVars, nclust = 3,
+    clustFun = pammix, conDist = L1Dist, catDist = matchingDist)
R> with(msRes, plot(weights, objFun, xlab = "Continuous Weight",
+    ylab = "Objective Function"))
```

Inspecting the objective function at each possible weighting (Figure 3), we see that a weight of 0.82 appears best.

We now run a KAMILA clustering procedure on the same data.

```
R> set.seed(5)
R> kmRes <- kamila(conVars, catVars, numClust = 3, numInit = 10,
+    maxIter = 50)
```

Comparing the partitions, we see that Modha-Spangler clustering identifies a substantially similar structure compared to KAMILA. We note, however, that KAMILA uses a more efficient algorithmic structure that does not require a brute-force search to optimize the continuous and categorical contribution to the resulting clustering.

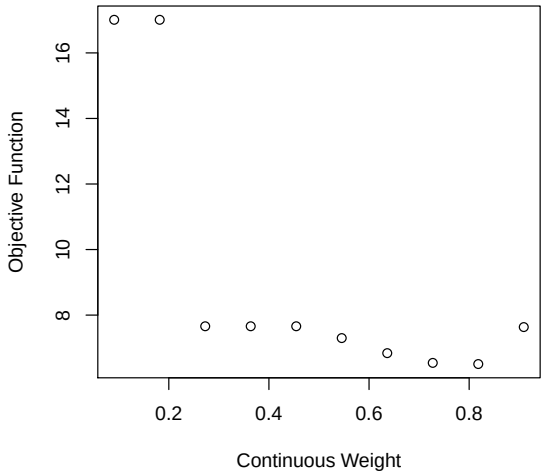

Figure 3: Modha-Spangler objective function values from the Byar prostate cancer data analysis. Values of the objective function are plotted against the continuous weight used in the Modha-Spangler clustering using PAM and Gower's distance.

```
R> table(kamila = kmRes$finalMemb, modhaSpangler = msRes$results$cluster)

      modhaSpangler
kamila   1   2   3
     1 101  29   0
     2  19   9 174
     3  13 125   5
```

There appears to be a strong relationship between the identified cluster structure and the outcome variable (survival, which was not used in constructing the clusters), which is supported by a chi-squared goodness-of-fit test:

```
R> ternarySurvival <- factor(Byar$SurvStat)
R> survNames <- c("Alive", "DeadProst", "DeadOther")
R> levels(ternarySurvival) <- survNames[c(1, 2, rep(3, 8))]
R> kamila3clusters <- factor(kmRes$finalMemb, 1:3, paste("Cluster", 1:3))
R> (kamilaSurvTab <- table(kamila3clusters, ternarySurvival))

               ternarySurvival
kamila3clusters Alive DeadProst DeadOther
      Cluster 1    24        11        95
      Cluster 2    46        90        66
      Cluster 3    67        20        56

R> chisq.test(kamilaSurvTab)


        Pearson's Chi-squared test
```

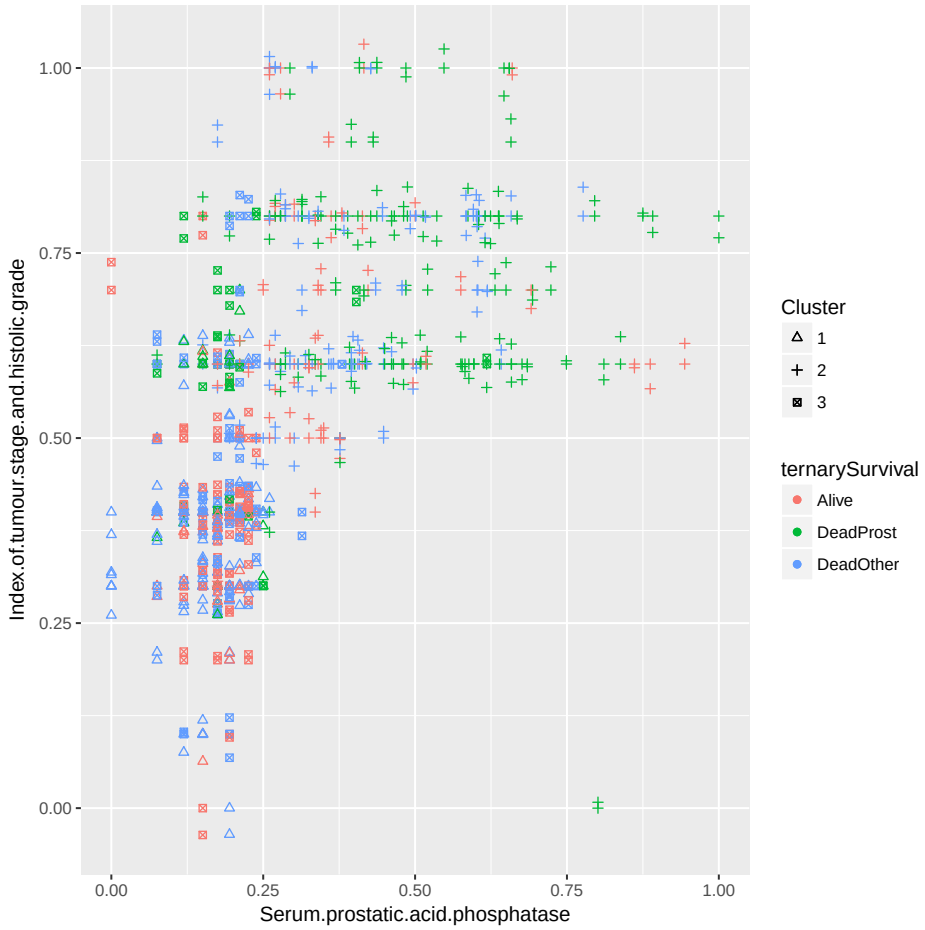

Figure 4: Results of the Byar prostate cancer data analysis. Values of tumor severity are plotted against serum prostatic acid phosphatase, with patient outcome depicted by plotting color and KAMILA cluster membership depicted by plotting character.

```
data:  kamilaSurvTab
X-squared = 104.74, df = 4, p-value < 2.2e-16
```

Cluster 2 is composed of a high proportion of patients who died due to prostate cancer (90/202; 74% of all patients in the entire data set who died due to prostate cancer), while cluster 1 is primarily composed of patients who died due to reasons other than prostate cancer (95/130). Cluster 3 contains primarily individuals who survived or died due to reasons unrelated to prostate cancer (123/143). These figures suggest that the clustering captures disease-specific mortality.

This is further supported by results shown in Figure 4, in which we see that cluster 2 appears to contain a preponderance of patients with high indices of tumor stage/grade, while clusters 1 and 3 appear to include predominantly patients with low serum prostatic acid phosphatase and low indices of tumor stage/grade.

```
R> suppressMessages(library("ggplot2"))
```

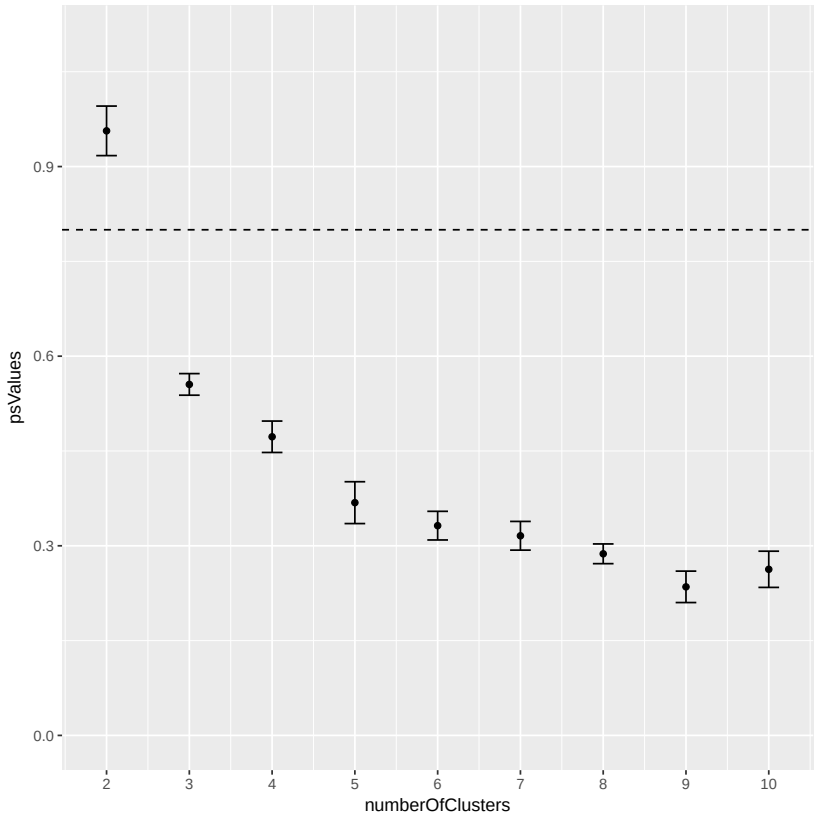

Figure 5: Prediction strength values for the KAMILA clustering of the Byar prostate cancer data. Prediction strength values are plotted against the number of clusters, with error bars denoting plus or minus one standard error. The horizontal dotted line at $y = 0.8$ denotes the default threshold for determining the number of clusters.

```
R> plotDatKam <- cbind(conVars, catVars, Cluster = factor(kmRes$finalMemb))
R> plotDatKam$Bone.metastases <- ifelse(plotDatKam$Bone.metastases == "1",
+    yes = "Yes", no = "No")
R> bonePlot <- ggplot(plotDatKam, aes(x = Serum.prostatic.acid.phosphatase,
+    y = Index.of.tumour.stage.and.histolic.grade, color = ternarySurvival,
+    shape = Cluster))
R> (bonePlot + geom_point() + scale_shape_manual(values = c(2, 3, 7))
+    + geom_jitter())
```

The three-way table below of KAMILA cluster membership, tumor stage, and bone metastasis shows suggestive relationships with other variables. For example, every individual in KAMILA cluster 2 had a tumor of stage 4, while all individuals in clusters 1 and 3 had tumors of stage 3. All except for one individual with metastatic spread of the prostate tumor to the bone was allocated to cluster 2.

```
R> ftable(Metastasis = ifelse(catVars$Bone.metastases == "1", yes = "Yes",
+    no = "No"), "Tumor Stage" = paste("Stage", 3:4)[catVars$Stage],
+    Cluster = kmRes$finalMemb)
```

```
                        Cluster   1   2   3
Metastasis Tumor Stage
No         Stage 3              130   0 142
           Stage 4                0 126   0
Yes        Stage 3                0   0   1
           Stage 4                0  76   0
```

We now run KAMILA clustering again, using the prediction strength method to estimate the number of clusters.

```
R> set.seed(6)
R> numberOfClusters <- 2:10
R> kmResPs <- kamila(conVars, catVars, numClust = numberOfClusters,
+    numInit = 10, maxIter = 50, calcNumClust = "ps")
```

Prediction strength values for each of number of clusters tested are shown in Figure 5. The selected number of clusters is two. In the table below, we see that the two-cluster KAMILA solution is identical to the three-cluster KAMILA solution except that the two clusters with the lowest proportion of deaths due to prostate cancer are merged.

```
R> psPlot <- with(kmResPs$nClust, qplot(numberOfClusters, psValues) +
+    geom_errorbar(aes(x = numberOfClusters, ymin = psValues - stdErrPredStr,
+    ymax = psValues + stdErrPredStr), width = 0.25))
R> psPlot <- psPlot + geom_hline(yintercept = 0.8, lty = 2)
R> psPlot + scale_x_continuous(breaks = numberOfClusters) + ylim(0, 1.1)
R> table(kamila3 = kmRes$finalMemb, kamila2 = kmResPs$finalMemb)
```

```
       kamila2
kamila3   1   2
      1 130   0
      2   0 202
      3 143   0
```

6. Hadoop implementation

We illustrate here how to implement KAMILA on very large data sets stored on distributed file systems. We use the map-reduce computing model (Dean and Ghemawat 2008) as implemented in **Hadoop** (Apache Software Foundation 2016a; White 2009). We also make use of the **Hadoop** streaming utility (Apache Software Foundation 2016b), available in the standard **Hadoop** distribution. **Hadoop** streaming allows mappers and reducers to be written in any programming language to be integrated into an otherwise standard **Hadoop** map-reduce run; this allows for rapid development of custom applications. In our case, it offers a way to implement map-reduce runs that have access to the rich set of objects and methods available in R.

There are multiple existing R packages that facilitate computing with data sets stored on distributed file systems. The **pbdR** project (Ostrouchov *et al.* 2012) is a collection of packages for high-performance computing on distributed systems, including interfaces to MPI

(The MPI Forum 2015) and **ScaLAPACK** (Blackford *et al.* 1997). Interfacing with **Hadoop** can be accomplished with the packages **RHIPE** (Guha *et al.* 2012; Guha 2012), **rmr2** (Revolution Analytics 2016a), or **hive** (Feinerer and Theussl 2015), while interfacing with **Spark** (Apache Software Foundation 2016d) can be accomplished using the R package **sparklyr** (Luraschi, Ushey, Allaire, and The Apache Software Foundation 2016) or the **SparkR** package (Venkataraman 2013) included in **Spark** (Apache Software Foundation 2016d) versions ≥ 1.4. Although we opted to use a bare-bones implementation of **Hadoop** streaming in the current work to minimize software dependencies, future implementations may draw upon the packages mentioned above in order to allow distributed computations to be executed from within the R environment.

The algorithm used in the **Hadoop** implementation of KAMILA is shown in Algorithm 1. Unless otherwise noted, it is structured in fundamentally the same manner as detailed in Section 4 and in Algorithm 1 in Foss *et al.* (2016), although certain modifications were necessary to accommodate map-reduce operations. Initializations of the continuous variables are not drawn from the entire data set, but a subset of the continuous data points (around 2,000 to 5,000 points are adequate) to reduce computation time. In the REDUCE 1 step of Algorithm 1, values of $r_i^{(t)}$ are binned to avoid the computation required to store a vector of length N between the first and second map-reduce runs. For the calculation of the radial kernel density estimate between map-reduce runs 1 and 2, calculations are performed assuming the points are drawn from the mid-point of the bins; the variance of the binned values is computed using Sheppard's correction (Stuart and Ord 1998), in which a correction factor of $h^2/12$ is subtracted from the standard variance estimator, where h is the bin width. The stopping rule given in Equation 5 is used in order to avoid the costly storage and comparison of two vectors of length N every iteration.

Compared to KAMILA, k-means and Modha-Spangler algorithms pose problems in a high-performance computing environment with large mixed-type data sets. In addition to substantial problems balancing continuous and categorical contribution described in Section 2, k-means may cause difficulties with large data sets due to the computational demands of dummy coding categorical variables. It is not uncommon to encounter large data sets with categorical variables containing a large number of levels; since each dummy-coded level requires its own variable the corresponding storage and run-time demands can quickly become challenging to handle. Each iteration of the KAMILA algorithm requires two passes through the continuous variables and one through the categorical, and thus has run-time approximately proportional to $2p + q$, where p is the number of continuous variables and q the number of categorical variables. One k-means iteration has run-time $p + q^*$, where q^* is the number of dummy-coded categorical levels. If $p = q$ and there are ℓ levels per categorical variable, the ratio of these run-times is $(p + q^*)/(2p + q) = (\ell + 1)/3$. Thus, if every level is given its own dummy variable, then computation times will be comparable if all variables are binary. Since variables often have many levels, KAMILA will generally outperform k-means in this context. (We note that simplex coding (McCane and Albert 2008) can reduce this ratio slightly to $\ell/3$.) Since the standard formulation of Modha-Spangler is based on k-means, it suffers from these same computational challenges, but with run-times multiplied by the number of evaluations in its brute-force optimization scheme.

This setup requires Java 1.6.0_22, **Hadoop** 2.5.1, **myhadoop** 0.30b (`https://github.com/glennklockwood/myhadoop/tree/v0.30b`), R 3.0.0, and the SLURM workload manager version 16.05.3 (Yoo, Jette, and Grondona 2003). More detailed instructions for installing

Algorithm 1 KAMILA map-reduce algorithm

for User-specified number of initializations **do**
  Initialize $\hat{\boldsymbol{\mu}}_g^{(0)}, \hat{\boldsymbol{\theta}}_{gq}^{(0)}$ for $g = 1, 2, \ldots, G$ clusters and $q = 1, 2, \ldots, Q$ categorical variables.
  repeat
    MAP 1
      $d_{ig}^{(t)} \leftarrow \text{dist}(\mathbf{v}_i, \hat{\boldsymbol{\mu}}_g^{(t)})$ for $i = 1, 2, \ldots, N$ observed data vectors
      $r_i^{(t)} \leftarrow \min_g(d_{ig}^{(t)})$
      Key: $\operatorname{argmin}_g(d_{ig}^{(t)})$
      Value: $r_i^{(t)}$
    REDUCE 1: Calculate $r_i^{*(t)}$ as binned $r_i^{(t)}$ values.
    $\hat{f}_{\mathbf{V}}^{(t)} \leftarrow \text{RadialKDE}(\mathbf{r}^{*(t)})$
    MAP 2
      $c_{ig}^{(t)} \leftarrow \widehat{\mathsf{P}}(\mathbf{w}_i \,|\, \text{observation } i \in \text{population } g)$
      $H_i^{(t)}(g) \leftarrow \log\left[\hat{f}_{\mathbf{V}}^{(t)}(d_{ig}^{(t)})\right] + \log\left[c_{ig}^{(t)}\right]$
      Key: $\operatorname{argmax}_g\{H_i^{(t)}(g)\}$
      Value: Original data vector $(\mathbf{v}_i, \mathbf{w}_i)$
    REDUCE 2: Calculate $\hat{\boldsymbol{\mu}}_g^{(t+1)}$ and $\hat{\boldsymbol{\theta}}_{gq}^{(t+1)}$.
  until Stopping rule is satisfied.
end for
Output partition that maximizes the chosen objective function.

Hadoop can be found in Appendix A of Theußl, Feinerer, and Hornik (2012). The environment modules package (`http://www.modules.sourceforge.net/`) is used to manage packages and environment variables. We do not expect all interested parties to have the same setup, and thus some adaptation may be required to implement our programs; however, we have separated out the components such that the mappers and reducers can be conveniently integrated into alternative setups using **Hadoop** streaming.

6.1. Running the analysis in Hadoop

Here we give a brief overview of the code implementing KAMILA in **Hadoop**. First we navigate to the relevant directory where the supplementary material is contained and list the content:

```
~$ ls -1F --gr kamilaStreamingHadoop/

BASH/
csv/
py/
R/
Rnw/
kamila.slurm
kmeans.slurm
```

```
LICENSE
preprocessing.slurm
README.md
```

Note that the sources may also be downloaded from GitHub (https://github.com/ahfoss/kamilaStreamingHadoop.git).

As stated above, our methods may require adaptation to the particular platform of interest. The file `preprocessing.slurm` is a SLURM batch submission script showing an example preprocessing pipeline using **SQLite** (Hipp 2013, version 3.7.17). Three data files are required to run the clustering program. In addition to the main data file (which currently must be in CSV format), the program requires a TSV file with one row per categorical variable describing basic metadata including column indices, number of levels, and the names and counts of the categorical levels. Required formatting is described in `README.md`, and the script `py/preprocKamila.py` gives example code for generating required inputs from a raw data set. Categorical variables should not be dummy coded. The levels of each categorical variable should be replaced by the integers 1–M, where M is the number of categorical levels in that variable. Finally, as described above, a small random subset (without replacement; about 2,000 to 5,000 should suffice) of the *continuous* data points should be drawn and placed in a separate data file. The file `kamila.slurm` is a SLURM batch submission script that implements KAMILA clustering in streaming **Hadoop**. The SLURM options and BASH variables defined in the initial lines of `kamila.slurm` must be set to match the specifications of the job at hand. Detailed instructions for these settings can be found in the document `README.md` and in Appendix A.

Once `kamila.slurm` and the data set are set up, the job can be submitted to the SLURM job manager using the `sbatch` command. In order to illustrate the data format required, a sample analysis can be executed from the package as follows from a linux terminal. First, a toy data set `csv/sample.csv` can be generated using an included R script:

```
~$ cd kamilaStreamingHadoop/
~/kamilaStreamingHadoop$ Rscript R/genData.R
~/kamilaStreamingHadoop$ ls -1sh csv/

total 604K
604K sample.csv
```

A preprocessing script formats the data and generates the required metadata as described above. A **SQLite** data base is created containing other useful summary information.

```
~/kamilaStreamingHadoop$ sh preprocessing.slurm
~/kamilaStreamingHadoop$ ls -1sh csv/

total 998K
604K sample.csv
4.0K sample_KAM_rmvna_catstats.tsv
300K sample_KAM_rmvna_norm.csv
 80K subsampled_2000_sample_KAM_rmvna_norm.csv
```

```
~/kamilaStreamingHadoop$ ls -1sh db/

total 760K
760K sample.db
```

Note that the `preprocessing.slurm` script can be instead run using the SLURM job manager:

```
~/kamilaStreamingHadoop$ slurm preprocessing.slurm
```

Next, cluster the data:

```
~/kamilaStreamingHadoop$ slurm kamila.slurm
```

Results of the analysis will be stored in the directory `output-kamila-####`, where `####` denotes the SLURM job ID. See Appendix A for a more detailed description of the raw output file structure.

Finally, a sample script can be used to generate useful summary information about the clustering procedure and results using the script `Rnw/kamilaSummary.Rnw`. The `JOBID` variable in the section "User-Supplied Values" of `kamilaSummary.Rnw` must be changed to the job ID used by SLURM to run `kamila.slurm`.

```
~/kamilaStreamingHadoop$ cd Rnw/
~/kamilaStreamingHadoop/Rnw$ Rscript -e "knitr::knit(\"kamilaSummary.Rnw\")"
~/kamilaStreamingHadoop/Rnw$ pdflatex kamilaSummary.tex
~/kamilaStreamingHadoop/Rnw$ !!
~/kamilaStreamingHadoop/Rnw$ evince kamilaSummary.pdf &
```

6.2. Airline data set

We illustrate our **Hadoop** software on the airline data set used in the ASA Data Expo 2009 (American Statistical Association 2016, http://stat-computing.org/dataexpo/2009/, accessed April 2016). All files necessary for downloading the data and conducting the analysis are part of the supplementary material. In addition they can be found on GitHub

Variable name	Mean	Standard deviation	Min	Max
Year	2.00E+03	4.04E+00	2.00E+03	2.01E+03
DepTime	1.32E+01	4.79E+00	0.00E+00	2.40E+01
ActualElapsedTime	1.24E+02	7.00E+01	0.00E+00	1.88E+03
AirTime	1.03E+02	6.74E+01	0.00E+00	3.51E+03
ArrDelay	7.42E+00	3.33E+01	−1.43E+03	2.60E+03
DepDelay	8.79E+00	3.08E+01	−1.41E+03	2.60E+03
Distance	7.30E+02	5.64E+02	8.00E+00	4.96E+03

Table 1: Summary statistics for continuous variables in the airline data set. A negative delay corresponds to an event that occurred earlier than the originally scheduled time. Note: statistics are calculated before standardizing to mean 0 and standard deviation 1.

Variable name	Level name	Counts	Percent
Month	8	7370927	8.8
Month	7	7326866	8.7
Month	3	7233892	8.6
Month	5	7181463	8.5
Month	10	7109262	8.4
Month	6	7077369	8.4
Month	4	7013408	8.3
Month	1	6985085	8.3
Month	12	6901993	8.2
Month	11	6766812	8.0
Month	9	6756115	8.0
Month	2	6464065	7.7
DayOfWeek	Mon	12370335	14.7
DayOfWeek	Fri	12360530	14.7
DayOfWeek	Thu	12325489	14.6
DayOfWeek	Wed	12312472	14.6
DayOfWeek	Tue	12262869	14.6
DayOfWeek	Sun	11774948	14.0
DayOfWeek	Sat	10780614	12.8
UniqueCarrier	WN	13029547	15.5
UniqueCarrier	DL	10203541	12.1
UniqueCarrier	AA	9411379	11.2
UniqueCarrier	UA	8569494	10.2
UniqueCarrier	US	8065539	9.6
UniqueCarrier	NW	6752509	8.0
UniqueCarrier	CO	4888454	5.8
UniqueCarrier	MQ	3789853	4.5
UniqueCarrier	OO	3020263	3.6
UniqueCarrier	XE	2290458	2.7
UniqueCarrier	HP	2176097	2.6
UniqueCarrier	AS	2104490	2.5
UniqueCarrier	TW	1839203	2.2
UniqueCarrier	EV	1626952	1.9
UniqueCarrier	OH	1409415	1.7
UniqueCarrier	FL	1249313	1.5
UniqueCarrier	YV	822290	1.0
UniqueCarrier	B6	799117	0.9
UniqueCarrier	DH	669645	0.8
UniqueCarrier	9E	504508	0.6
UniqueCarrier	F9	334842	0.4
UniqueCarrier	HA	272834	0.3
UniqueCarrier	TZ	206007	0.2
UniqueCarrier	AQ	151507	0.2

Table 2: Summary statistics for categorical variables in the airline data set. Carriers are denoted by IATA code; see the Supplemental Data Sources tab on the website given in the American Statistical Association (2016) citation for full carrier names by IATA code.

Variable name	Level name	Counts	Percent
Origin	ORD	4448835	5.3
Origin	ATL	4335383	5.1
Origin	DFW	3863123	4.6
Origin	LAX	2857305	3.4
Origin	PHX	2484176	3.0
Origin	DEN	2212864	2.6
Origin	IAH	2179533	2.6
Origin	DTW	2037127	2.4
Origin	LAS	1986152	2.4
Origin	MSP	1916848	2.3
Origin	EWR	1791026	2.1
Origin	SFO	1777866	2.1
Origin	STL	1724370	2.0
Origin	CLT	1665273	2.0
Origin	BOS	1531311	1.8
Origin	PHL	1490657	1.8
Origin	SLC	1486335	1.8
Origin	LGA	1459035	1.7
Origin	SEA	1416031	1.7
Origin	CVG	1407283	1.7
Origin	MCO	1381332	1.6
Origin	BWI	1234570	1.5
Origin	PIT	1159364	1.4
Origin	DCA	1157242	1.4
Origin	SAN	1088436	1.3
Origin	IAD	979922	1.2
Origin	JFK	971538	1.2
Origin	CLE	952584	1.1
Origin	MDW	927936	1.1
Origin	MIA	906114	1.1
Origin	TPA	896608	1.1
Origin	OAK	885349	1.1
Origin	MCI	808648	1.0
Origin	SJC	791986	0.9
Origin	HOU	775095	0.9
Origin	PDX	761713	0.9
Origin	MEM	757583	0.9
Origin	BNA	748108	0.9
Origin	FLL	735799	0.9
Origin	DAL	652207	0.8
Origin	MSY	636294	0.8
Origin	RDU	608314	0.7
Origin	SMF	605927	0.7
Origin	SNA	592273	0.7
Origin	AUS	577642	0.7
Origin	SAT	546237	0.6
Origin	IND	520262	0.6
Origin	ABQ	519070	0.6
Origin	CMH	513464	0.6
Origin	ONT	501304	0.6
Origin	Other	15923803	18.9

Table 3: Summary statistics for categorical variables in the airline data set: flight origin.

Variable name	Level name	Counts	Percent
Dest	ORD	4436438	5.3
Dest	ATL	4328774	5.1
Dest	DFW	3851797	4.6
Dest	LAX	2857883	3.4
Dest	PHX	2479383	2.9
Dest	DEN	2212456	2.6
Dest	IAH	2173218	2.6
Dest	DTW	2035856	2.4
Dest	LAS	1989516	2.4
Dest	MSP	1913408	2.3
Dest	EWR	1787315	2.1
Dest	SFO	1775794	2.1
Dest	STL	1723698	2.0
Dest	CLT	1663261	2.0
Dest	BOS	1533501	1.8
Dest	PHL	1491679	1.8
Dest	SLC	1487740	1.8
Dest	LGA	1457226	1.7
Dest	SEA	1415457	1.7
Dest	CVG	1403933	1.7
Dest	MCO	1383621	1.6
Dest	BWI	1235708	1.5
Dest	DCA	1158146	1.4
Dest	PIT	1156441	1.4
Dest	SAN	1089037	1.3
Dest	IAD	978939	1.2
Dest	JFK	971166	1.2
Dest	CLE	952784	1.1
Dest	MDW	927501	1.1
Dest	MIA	907612	1.1
Dest	TPA	898406	1.1
Dest	OAK	885841	1.1
Dest	MCI	809835	1.0
Dest	SJC	792899	0.9
Dest	HOU	774311	0.9
Dest	PDX	762550	0.9
Dest	MEM	757828	0.9
Dest	BNA	749699	0.9
Dest	FLL	737673	0.9
Dest	DAL	650729	0.8
Dest	MSY	637334	0.8
Dest	RDU	609729	0.7
Dest	SMF	606564	0.7
Dest	SNA	592762	0.7
Dest	AUS	578469	0.7
Dest	SAT	546991	0.6
Dest	IND	521711	0.6
Dest	ABQ	520139	0.6
Dest	CMH	515299	0.6
Dest	ONT	502140	0.6
Dest	Other	15957060	19.0

Table 4: Summary statistics for categorical variables in the airline data set: flight destination.

Cluster number	Count	Total Euclidean distance to centroid	Total categorical log-likelihood
1	6748386	1.38E+07	−8.84E+07
2	12877112	1.55E+07	−1.74E+08
3	13275607	1.37E+07	−1.81E+08
4	10721486	1.56E+07	−1.44E+08
5	13810601	1.58E+07	−1.86E+08
6	12522120	1.28E+07	−1.68E+08
7	11019742	1.54E+07	−1.54E+08
8	3212083	9.12E+06	−4.46E+07

Table 5: Airline data: Cluster-specific information for the best KAMILA run, including number of observations in each cluster, the summed Euclidean distance of each point to its corresponding centroid (continuous variables only), and the log-likelihood of the multinomial model with respect to the categorical variables.

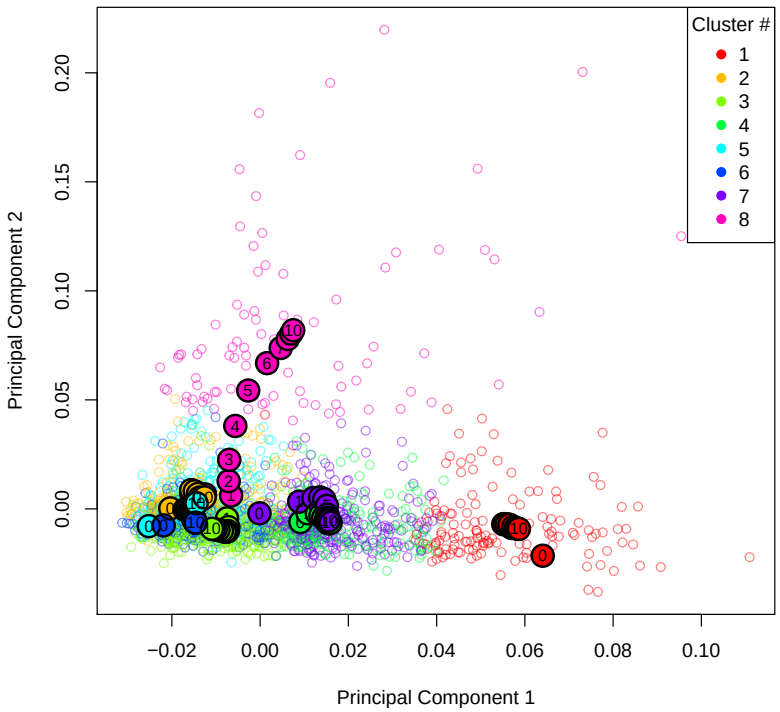

Figure 6: Results of clustering the airline data using KAMILA. A subset of the data is plotted using the first two principal components of the sub-sample. Cluster number is depicted by color, with the number within the plotted points depicting the iteration number of the best k-means run. Loadings of the variables on the PCs are shown in Table 6.

(`https://github.com/ahfoss/airline-hadoop-analysis`); the analysis can be reproduced by running the commands found in the README file.

Summary statistics for the variables used are given in Tables 1, 2, 3, and 4. If a row had a missing entry for any of the variables used it was removed from the analysis. Continuous

Variable index	Variable name	PC1	PC2
1	Year	0.043	0.000
2	DepTime	-0.001	0.270
3	ActualElapsedTime	0.575	−0.045
4	AirTime	0.572	−0.070
5	ArrDelay	0.082	0.674
6	DepDelay	0.077	0.679
7	Distance	0.572	−0.074

Table 6: Airline data: Loadings of the continuous variables on principal components 1 and 2 in Figure 6. The first two principal components account for 70.5 % of the variance.

Cluster number	Min	Mean	Max
1	13.000	265.821	1958.000
2	1.000	66.317	612.000
3	0.000	70.217	1615.000
4	4.000	147.483	694.000
5	0.000	60.255	3508.000
6	1.000	62.366	516.000
7	1.000	144.916	2582.000
8	0.000	98.292	1651.000

Table 7: Cluster-specific minimum, mean, and maximum for the continuous variable describing length of time the flight was off the ground (AirTime).

Cluster number	Min	Mean	Max
1	−1188.000	5.116	1504.000
2	−1302.000	7.174	2453.000
3	−950.000	−2.143	1942.000
4	−1426.000	3.798	1655.000
5	−978.000	3.820	1438.000
6	−212.000	0.781	2598.000
7	−1298.000	2.079	1925.000
8	−1035.000	124.524	1724.000

Table 8: Cluster-specific minimum, mean, and maximum for the continuous arrival delay in minutes (ArrDelay). A negative delay corresponds to an event that occurred earlier than the originally scheduled time.

variables were z-normalized, and categorical variables were re-coded using unique integers for each categorical level. There were a total of 335 and 331 terminals in the origin and destination variables, respectively; we restricted the analysis to the 50 most prevalent terminals in both cases and grouped other terminals in an "other" category. We note that if we had been forced to rely on a method requiring dummy coding, we would require 145 variables to represent the categorical variables instead of five, a 2800% increase in the number of variables required. Based on the approximate calculations described in the opening of Section 6, this corresponds roughly to an 8× increase in computation time for k-means over KAMILA.

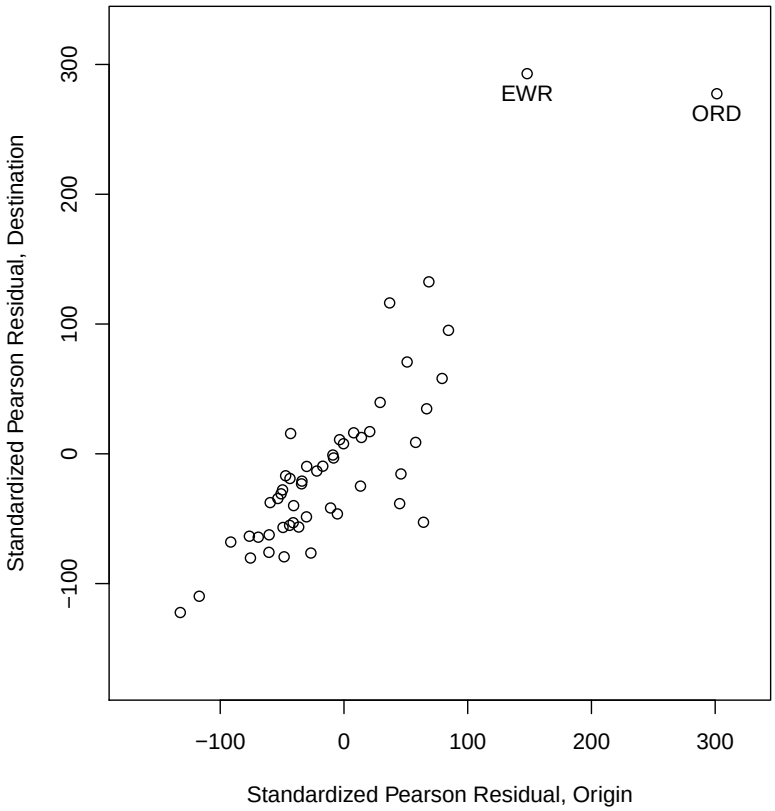

Figure 7: The most prominent airlines in cluster 8 of the airline data analysis. Pearson residuals were calculated for the cross-tabulations of cluster membership and airport, separately for origin and destination airports. Residuals for only cluster 8 were then plotted. Higher numbers represent increased counts compared to the expected counts if the cluster were unrelated to the variable in question.

6.3. Results of the airline analysis

We ran the KAMILA analysis using 16 initializations with a maximum of 10 iterations per initialization (as shown in Algorithm 1 each iteration includes two map-reduce steps). Four runs of four initializations were run in parallel, with the data split into 80 chunks for the map-reduce steps. Our computing setup used 320 cores and ran for a wall time of 72 hours, a little more than two and a half years of CPU time.

A plot of the clustering results for the airline data are shown in Figure 6. This plot shows a subset of the data plotted on the first two principal components of the sub-sampled continuous variables. Loadings of the continuous variables on these principal components are given in Table 6, and account for 70.5% of the variance in the subset.

Cluster-specific minimum, maximum, and mean values are shown for the air time variable in Table 7, and for arrival delay in Table 8. Figure 6 shows that the most prominent clusters with regard to continuous variables are cluster 1, which appears to capture longer flights, and cluster 8, which appears to capture flights with larger arrival and departure delays. This is

Terminal	Mean departure time (minutes)	Terminal	Mean departure time (minutes)
ORD	13.87	SMF	13.08
EWR	13.66	CMH	13.08
JFK	13.63	TPA	13.08
PHL	13.63	RDU	13.08
ATL	13.60	MSP	13.07
MDW	13.56	CVG	13.06
MIA	13.50	BNA	13.05
SFO	13.47	PIT	13.05
HOU	13.44	IAH	12.98
IAD	13.44	MEM	12.97
LAS	13.42	MCI	12.96
DTW	13.39	ABQ	12.96
PHX	13.35	SAN	12.96
DFW	13.33	PDX	12.94
FLL	13.31	MSY	12.93
DEN	13.30	ONT	12.90
BWI	13.27	Other	12.90
SEA	13.27	SLC	12.89
BOS	13.27	SJC	12.88
STL	13.26	IND	12.87
DAL	13.26	CLE	12.83
LGA	13.22	SAT	12.82
CLT	13.18	AUS	12.79
LAX	13.14	SNA	12.75
MCO	13.12	DCA	12.70
OAK	13.08		

Table 9: Mean departure delay by airport in the airline data set.

	Mon	Tue	Wed	Thu	Fri	Sat	Sun
Cluster 1	972438	971981	971373	966847	962540	969449	933758
Cluster 2	1902033	1893070	1881387	1888707	1887205	1933275	1491435
Cluster 3	1960129	1959462	1988005	1984145	1968617	1686060	1729189
Cluster 4	1551125	1528441	1536946	1547241	1550589	1506646	1500498
Cluster 5	2047288	2017011	1992459	2035587	2040594	2034428	1643234
Cluster 6	1858126	1812502	1830218	1852937	1866861	1602267	1699209
Cluster 7	1600030	1592269	1593626	1591320	1584428	1584879	1473190
Cluster 8	479151	585774	531455	445678	402020	457924	310081

Table 10: Counts of flights by cluster and day of week.

confirmed by inspecting variable means in Tables 7 and 8.

In order to investigate which departure terminals were most prominently associated with cluster 8, we calculated Pearson residuals over the cross-tabulation of cluster membership (in all clusters 1–8) and departure terminal. Pearson residuals were calculated using the output value `residuals` from `chisq.test` in R, calculated as $(O - E)/\sqrt{E}$, where O denotes observed counts and E denotes expected counts under an independence model. Note that the clusters are data-dependent and thus a chi-squared goodness-of-fit test is invalid; we use Pearson residuals as descriptive statistics only to adjust for differences in overall cluster size

	Mon	Tue	Wed	Thu	Fri	Sat	Sun
Cluster 1	−19.2	−18.9	−16.7	−20.2	−20.6	26.3	74.9
Cluster 2	7.2	1.8	−2.8	3.9	8.4	98.5	−122.7
Cluster 3	6.8	7.4	31.8	30.6	25.1	−125.3	22.4
Cluster 4	−19.3	−36.4	−26.1	−16.6	−8.9	5.8	108.9
Cluster 5	12.6	−7.5	−20.7	11.1	20.4	74.0	−94.2
Cluster 6	13.4	−19.2	−2.3	15.9	31.7	−112.7	75.6
Cluster 7	−15.1	−20.2	−15.5	−16.0	−16.4	35.1	52.2
Cluster 8	10.4	166.3	89.2	−35.2	−96.3	12.9	−157.9

Table 11: Pearson residuals for counts of flights by cluster and day of week.

	Mon	Tue	Wed	Thu	Fri	Sat	Sun
WN	1933234	1944177	1928090	1937827	1936815	1788337	1561067
DL	1478438	1472266	1470108	1469804	1462119	1455216	1395590
AA	1370741	1369607	1362268	1365368	1365670	1332848	1244877
UA	1246535	1240902	1240207	1244582	1244501	1203927	1148840
US	1191480	1185983	1188343	1187252	1184223	1135607	992651
NW	992370	989472	989432	991630	991502	931525	866578
CO	734457	734909	734896	721185	712921	671245	578841
MQ	557737	556454	555244	558987	557170	532900	471361
OO	442043	440401	440530	439745	435809	425281	396454
XE	355997	355773	357069	340979	331282	306530	242828
HP	316420	316720	315098	314681	314354	309713	289111
AS	307114	307718	305790	304230	305227	292767	281644
TW	270786	269008	268888	270515	269526	253542	236938
EV	236848	238506	236643	237286	232317	232806	212546
OH	209664	209450	210652	209643	201544	191859	176603
FL	180229	180992	179704	177368	176639	179833	174548
YV	119200	119844	119277	118780	117962	118499	108728
B6	115345	115279	114559	113598	113923	113445	112968
DH	96915	96769	96849	96912	96536	93631	92033
9E	74563	73370	73458	74682	74350	69241	64844
F9	49356	48759	48528	48401	48838	47253	43707
HA	38358	40767	38054	38171	38117	38729	40638
TZ	30844	30893	30391	29613	30423	28658	25185
AQ	21661	22511	21411	21233	21101	21556	22034

Table 12: Counts of flights by carrier IATA code and day of week. Carriers are denoted by IATA code; see the Supplemental Data Sources tab on the website given in the American Statistical Association (2016) citation for full carrier names by IATA code.

and terminal size. Pearson residuals were calculated in the same way for arrival terminals, and departure and arrival values for cluster 8 were plotted against each other in Figure 7. It appears that Chicago O'Hare (ORD) and Newark Liberty International Airport (EWR) are the most over-represented airports in cluster 8. An inspection of mean departure delays by airport (Table 9) confirms that ORD and EWR have the highest mean departure delays of

	Mon	Tue	Wed	Thu	Fri	Sat	Sun
WN	13.5	22.5	14.8	23.4	28.2	−25.2	−83.2
DL	−17.0	−21.1	−19.4	−18.4	−19.8	23.5	77.8
AA	−10.3	−10.4	−13.3	−9.4	−4.4	14.4	36.2
UA	−11.3	−15.4	−12.9	−7.8	−3.4	4.9	49.1
US	5.8	1.6	6.9	7.1	8.7	7.1	−39.5
NW	0.2	−2.0	0.8	4.1	8.0	−13.3	2.0
CO	19.1	20.3	22.7	7.4	1.0	−15.1	−59.6
MQ	1.2	0.0	0.5	6.3	6.9	3.9	−20.0
OO	−2.6	−4.6	−2.5	−3.0	−6.2	4.4	15.6
XE	33.5	33.6	37.5	10.4	−4.1	−24.4	−93.2
HP	−5.9	−4.9	−6.2	−6.3	−4.7	9.7	19.8
AS	−3.8	−2.3	−4.2	−6.4	−2.4	−2.9	23.4
TW	1.0	−2.0	−0.7	2.9	3.1	−7.3	2.9
EV	−4.5	−0.7	−3.2	−1.3	−9.6	11.0	9.2
OH	5.6	5.5	9.5	7.7	−8.3	−11.9	−9.1
FL	−7.8	−5.7	−7.5	−12.5	−12.5	12.2	36.4
YV	−4.7	−2.6	−3.2	−4.3	−5.2	10.3	10.6
B6	−6.1	−6.0	−7.1	−9.6	−7.3	5.0	33.3
DH	−4.7	−4.9	−3.8	−3.3	−3.2	−0.1	21.5
9E	1.6	−2.6	−1.5	3.3	3.2	−5.0	0.9
F9	0.7	−1.8	−2.2	−2.6	0.3	1.9	4.0
HA	−8.6	3.5	−9.5	−8.7	−8.1	2.9	30.5
TZ	3.3	3.7	1.3	−3.0	2.4	−0.9	−7.4
AQ	−4.0	1.8	−5.2	−6.2	−6.5	2.5	18.9

Table 13: Pearson residuals for counts of flights by carrier IATA code and day of week. Carriers are denoted by IATA code; see the Supplemental Data Sources tab on the website given in the American Statistical Association (2016) citation for full carrier names by IATA code.

all airports in the data set.

Counts of flights by day of the week for each cluster are shown in Table 10, with associated Pearson residuals in Table 11. An interesting observation is that counts for Saturday and Sunday are consistently more often over- or under-represented than this is the case for any other day of the week, despite the fact that Saturday and Sunday are the least frequent days for flying overall (see Table 2). Inspecting counts of flights for each carrier by day of the week (Table 12) and associated Pearson residuals (Table 13), it appears that flights on Sunday depart the most dramatically from an independence model. This is especially apparent in Figure 8, a boxplot depicting Pearson residuals of carrier by day counts (i.e., Table 13) aggregated across carriers. This suggests that airlines are markedly different in terms of how many flights they offer on weekends, and on Sunday in particular. While more work would be needed to identify the precise cause of this, one possible reason could be that some airlines do not choose to offer many Sunday flights due to lower demand on weekends. The difference could also be related to contrasting company policies regarding flight attendants' work on Sunday.

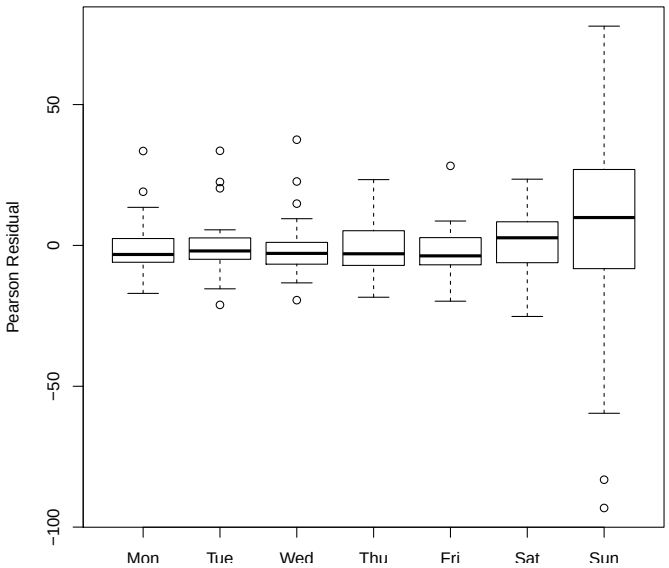

Figure 8: Pearson residuals for the cross-tabulation of flights by carrier and day of week. Pearson residuals are aggregated across carrier and plotted.

7. Conclusion

Although mixed-type data is quite common across many fields, effective strategies for clustering such data sets are surprisingly difficult to find. As discussed above, this is true even in the context of R, but it is particularly true when the data at hand is too large to be clustered in a standard R environment. Existing methods use arbitrary strategies for controlling the continuous and categorical contribution to the overall clustering, often resulting in undesirable solutions dominated by one type or the other. In this paper we introduced the R package **kamila** that offers multiple solutions to this problem. Additionally, we provide examples using **Hadoop** illustrating the implementation of KAMILA using a map-reduce model on a large data set stored on a distributed file system. Clustering mixed-type data continues to be a challenging problem; as these methods are refined, and as other effective strategies are discovered, we intend for the **kamila** package to serve as a centralized platform for implementing the most effective techniques in this area.

Acknowledgments

We thank the Center for Computational Research at the University at Buffalo for access to high-performance computing resources. Further, we thank two anonymous reviewers for thoughtful and useful feedback.

References

Amatya A, Demirtas H (2015). "Simultaneous Generation of Multivariate Mixed Data with Poisson and Normal Marginals." *Journal of Statistical Computation and Simulation*, **85**(15), 3129–3139. doi:10.1080/00949655.2014.953534.

American Statistical Association (2016). *Airline On-Time Performance. American Statistical Association Sections on Statistical Computing and Statistical Graphics*. Accessed April 2016, URL http://stat-computing.org/dataexpo/2009/.

Apache Software Foundation (2016a). *Apache* ***Hadoop***. Accessed April 2016, URL https://hadoop.apache.org/.

Apache Software Foundation (2016b). *Apache* ***Hadoop*** *Streaming*. Accessed April 2016, URL https://hadoop.apache.org/docs/r1.2.1/streaming.html.

Apache Software Foundation (2016c). *Apache* ***Mahout****: Scalable Machine-Learning and Data-Mining Library*. Accessed April 2016, URL https://mahout.apache.org/users/basics/algorithms.html.

Apache Software Foundation (2016d). *Apache* ***Spark****: Ligntning-Fast Cluster Computing*. Accessed October 2016, URL http://spark.apache.org/.

Arthur D, Manthey B, Röglin H (2011). "Smoothed Analysis of the k-Means Method." *Journal of the ACM*, **58**(5), 1–31. doi:10.1145/2027216.2027217. Article number 19.

Azzalini A, Menardi G (2014). "Clustering via Nonparametric Density Estimation: The R Package **pdfCluster**." *Journal of Statistical Software*, **57**(11), 1–26. doi:10.18637/jss.v057.i11.

Benaglia T, Chauveau D, Hunter DR, Young D (2009). "**mixtools**: An R Package for Analyzing Finite Mixture Models." *Journal of Statistical Software*, **32**(6), 1–29. doi:10.18637/jss.v032.i06.

Blackford LS, Choi J, Cleary A, D'Azeuedo E, Demmel J, Dhillon I, Hammarling S, Henry G, Petitet A, Stanley K, Walker D, Whaley RC (1997). ***ScaLAPACK*** *User's Guide*. Society for Industrial and Applied Mathematics, Philadelphia. ISBN 0-89871-397-8.

Browne RP, ElSherbiny A, McNicholas PD (2015). ***mixture****: Mixture Models for Clustering and Classification*. R package version 1.4, URL https://CRAN.R-project.org/package=mixture.

Chen WC, Maitra R (2015). ***EMCluster****: EM Algorithm for Model-Based Clustering of Finite Mixture Gaussian Distribution*. R package version 0.2-5, URL https://CRAN.R-project.org/package=EMCluster.

Chen WC, Ostrouchov G (2016). ***pmclust****: Parallel Model-Based Clustering Using Expectation-Gathering-Maximization Algorithm for Finite Mixture Gaussian Model*. R package version 0.1-9, URL https://CRAN.R-project.org/package=pmclust.

Chipman H, Tibshirani R (2006). "Hybrid Hierarchical Clustering with Applications to Microarray Data." *Biostatistics*, **7**(2), 286–301. doi:10.1093/biostatistics/kxj007.

Chipman H, Tibshirani R, Hastie T (2015). ***hybridHclust****: Hybrid Hierarchical Clustering*. R package version 1.0-5, URL https://CRAN.R-project.org/package=hybridHclust.

Cooper MC, Milligan GW (1988). "The Effect of Measurement Error on Determining the Number of Clusters in Cluster Analysis." In W Gaul, M Schader (eds.), *Data, Expert Knowledge and Decisions: An Interdisciplinary Approach with Emphasis on Marketing Applications*, pp. 319–328. Springer-Verlag, Berlin, Heidelberg.

Dahl DB (2016). ***xtable**: Export Tables to LaTeX or HTML*. R package version 1.8-2, URL https://CRAN.R-project.org/package=xtable.

Dean J, Ghemawat S (2008). "MapReduce: Simplified Data Processing on Large Clusters." *Communications of the ACM*, **51**(1), 107–113. doi:10.1145/1327452.1327492.

Demirtas H, Hu Y, Allozi R (2016). ***PoisBinOrdNor**: Data Generation with Poisson, Binary, Ordinal and Normal Components*. R package version 1.2, URL https://CRAN.R-project.org/package=PoisBinOrdNor.

Dempster AP, Laird NM, Rubin DB (1977). "Maximum Likelihood from Incomplete Data via the EM Algorithm." *Journal of the Royal Statistical Society B*, **39**(1), 1–38.

Feinerer I, Theussl S (2015). ***hive**: **Hadoop** InteractiVE*. R package version 0.2-0, URL https://CRAN.R-project.org/package=hive.

Foss A, Markatou M (2018). ***kamila**: Methods for Clustering Mixed-Type Data*. R package version 0.1.1.2, URL https://CRAN.R-project.org/package=kamila.

Foss A, Markatou M, Ray B, Heching A (2016). "A Semiparametric Method for Clustering Mixed Data." *Machine Learning*, **105**(3), 419–458. doi:10.1007/s10994-016-5575-7.

Fraley C, Raftery AE, Murphy TB, Scrucca L (2012). "**mclust** Version 4 for R: Normal Mixture Modeling for Model-Based Clustering, Classification, and Density Estimation." *Technical Report 597*, Department of Statistics, University of Washington.

Friedman JH, Meulman JJ (2004). "Clustering Objects on Subsets of Attributes." *Journal of the Royal Statistical Society B*, **66**(4), 815–849. doi:10.1111/j.1467-9868.2004.02059.x.

Fritz H, García-Escudero LA, Mayo-Iscar A (2012). "**tclust**: An R Package for a Trimming Approach to Cluster Analysis." *Journal of Statistical Software*, **47**(12), 1–26. doi:10.18637/jss.v047.i12.

Gower JC (1971). "A General Coefficient of Similarity and Some of Its Properties." *Biometrics*, **27**(4), 857–871. doi:10.2307/2528823.

Guha S (2012). ***RHIPE**: R and **Hadoop** Integrated Programming Environment*. Accessed October 2016, URL https://github.com/saptarshiguha/RHIPE.

Guha S, Hafen R, Rounds J, Xia J, Li J, Xi B, Cleveland WS (2012). "Large Complex Data: Divide and Recombine with **RHIPE**." *Stat*, **1**(1), 53–67. doi:10.1002/sta4.7.

Hartigan JA, Wong MA (1979). "Algorithm AS 136: A K-Means Clustering Algorithm." *Journal of the Royal Statistical Society C*, **28**(1), 100–108. doi:10.2307/2346830.

Hastie T, Tibshirani R, Friedman J (2009). *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. 2nd edition. Springer-Verlag, New York. doi:10.1007/978-0-387-21606-5.

Hennig C (2012). ***trimcluster**: Cluster Analysis with Trimming*. R package version 0.1-2, URL https://CRAN.R-project.org/package=trimcluster.

Hennig C (2015a). ***fpc**: Flexible Procedures for Clustering*. R package version 2.1-10, URL https://CRAN.R-project.org/package=fpc.

Hennig C (2015b). "What Are the True Clusters?" *Pattern Recognition Letters*, **64**, 53–62. doi:10.1016/j.patrec.2015.04.009.

Hennig C, Liao TF (2013). "How to Find an Appropriate Clustering for Mixed-Type Variables with Application to Socio-Economic Stratification." *Journal of the Royal Statistical Society C*, **62**(3), 309–369. doi:10.1111/j.1467-9876.2012.01066.x.

Hennig C, Lin C (2015). "Flexible Parametric Bootstrap for Testing Homogeneity Against Clustering and Assessing the Number of Clusters." *Statistics and Computing*, **25**(4), 821–833. doi:10.1007/s11222-015-9566-5.

Hipp DR (2013). ***SQLite**: A Relational Database Management System*. Version 3.7.17, URL http://sqlite.org/.

Huang Z (1998). "Extensions to the k-Means Algorithm for Clustering Large Data Sets with Categorical Values." *Data Mining and Knowledge Discovery*, **2**(3), 283–304. doi:10.1023/a:1009769707641.

Hubert L, Arabie P (1985). "Comparing Partitions." *Journal of Classification*, **2**(1), 193–218. doi:10.1007/bf01908075.

Hunt L, Jorgensen M (2011). "Clustering Mixed Data." *WIREs Data Mining and Knowledge Discovery*, **1**(4), 352–361. doi:10.1002/widm.33.

Karatzoglou A, Smola A, Hornik K, Zeileis A (2004). "**kernlab** – An S4 Package for Kernel Methods in R." *Journal of Statistical Software*, **11**(9), 1–20. doi:10.18637/jss.v011.i09.

Kaufman L, Rousseeuw PJ (1990). *Finding Groups in Data: An Introduction to Cluster Analysis*. John Wiley & Sons, New York. doi:10.1002/9780470316801.

Kotz S, Balakrishnan N, Johnson NL (2000). *Continuous Multivariate Distributions: Models and Applications*, volume 1 of *Continuous Multivariate Distributions*. 2nd edition. John Wiley & Sons. doi:10.1002/0471722065.

Langfelder P, Horvath S (2012). "Fast R Functions for Robust Correlations and Hierarchical Clustering." *Journal of Statistical Software*, **46**(11), 1–17. doi:10.18637/jss.v046.i11.

Langrognet F, Lebret R, Poli C, Iovleff S (2016). ***Rmixmod**: Supervised, Unsupervised, Semi-Supervised Classification with MIXture MODelling (Interface of MIXMOD Software)*. R package version 2.1.1, URL https://CRAN.R-project.org/package=Rmixmod.

Leisch F (2006). "A Toolbox for K-Centroids Cluster Analysis." *Computational Statistics & Data Analysis*, **51**(2), 526–544. doi:10.1016/j.csda.2005.10.006.

Leisch F, Grün B (2017). *CRAN Task View: Cluster Analysis & Finite Mixture Models*. Version 2017-12-05, URL https://CRAN.R-project.org/view=Cluster.

Lindsay BG (1995). *Mixture Models: Theory, Geometry, and Applications*. Institute for Mathematical Statistics, Hayward.

Luraschi J, Ushey K, Allaire JJ, The Apache Software Foundation (2016). ***sparklyr**: R Interface to Apache **Spark***. R package version 0.4, URL https://CRAN.R-project.org/package=sparklyr.

MacQueen J (1967). "Some Methods for Classification and Analysis of Multivariate Observations." In *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, Volume 1: Statistics*, pp. 281–297. University of California Press, Berkeley.

Maechler M, Rousseeuw P, Struyf A, Hubert M, Hornik K (2015). ***cluster**: Cluster Analysis Basics and Extensions*. R package version 2.0.3.

Maitra R, Melnykov V (2010). "Simulating Data to Study Performance of Finite Mixture Modeling and Clustering Algorithms." *Journal of Computational and Graphical Statistics*, **19**(2), 354–376. doi:10.1198/jcgs.2009.08054.

McCane B, Albert M (2008). "Distance Functions for Categorical and Mixed Variables." *Pattern Recognition Letters*, **29**(7), 986–993. doi:10.1016/j.patrec.2008.01.021.

McLachlan GJ, Basford KE (1988). *Mixture Models: Inference and Applications to Clustering*. Marcel Dekker, New York.

McLachlan GJ, Peel D (2000). *Finite Mixture Models*. John Wiley & Sons, New York. doi:10.1002/0471721182.

McParland D (2015). ***clustMD**: Model Based Clustering for Mixed Data*. R package version 1.1, URL https://CRAN.R-project.org/package=clustMD.

Microsoft Corporation (2016). *Microsoft Azure Machine Learning Studio*. Accessed April 2016, URL https://azure.microsoft.com/en-us/documentation/articles/machine-learning-algorithm-cheat-sheet/.

Milligan GW, Cooper MC (1985). "An Examination of Procedures for Determining the Number of Clusters in a Data Set." *Psychometrika*, **50**(2), 159–179. doi:10.1007/bf02294245.

Modha DS, Spangler WS (2003). "Feature Weighting in k-Means Clustering." *Machine Learning*, **52**(3), 217–237. doi:10.1023/a:1024016609528.

Müllner D (2013). "**fastcluster**: Fast Hierarchical, Agglomerative Clustering Routines for R and Python." *Journal of Statistical Software*, **53**(9), 1–18. doi:10.18637/jss.v053.i09.

Ostrouchov G, Chen WC, Schmidt D, Patel P (2012). *Programming with Big Data in R*. Accessed October 2016, URL http://R-pbd.org/.

Prates MO, Cabral CRB, Lachos VH (2013). "**mixsmsn**: Fitting Finite Mixture of Scale Mixture of Skew-Normal Distributions." *Journal of Statistical Software*, **54**(12), 1–20. doi:10.18637/jss.v054.i12.

R Core Team (2017). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. URL https://www.R-project.org/.

Revolution Analytics (2016a). *The **RHadoop** Project*. Accessed October 2016, URL https://github.com/RevolutionAnalytics/RHadoop.

Revolution Analytics (2016b). *Revolution R Enterprise **ScaleR**: Transparent Parallelism Accelerates Big Data Analytics Easily*. Accessed April 2016, URL http://www.revolutionanalytics.com/revolution-r-enterprise-scaler.

SAS Corporation (2016). *SAS LASR Analytic Server 2.3: Reference Guide*, PROC IMSTAT, *Cluster Statement*. Accessed April 2016, URL http://support.sas.com/documentation/cdl/en/inmsref/67306/HTML/default/viewer.htm.

Scott DW (1992). *Multivariate Density Estimation*. John Wiley & Sons. doi:10.1002/9780470316849.

Silverman BW (1986). *Density Estimation*. Chapman and Hall, London.

SLURM Team (2016). *SLURM Workload Manager Version 16.05; Sbatch*. Accessed August 2016, URL http://www.slurm.schedmd.com/sbatch.html.

Stuart A, Ord J (1998). *Kendall's Advanced Theory of Statistics*, volume 1. 6th edition. Oxford University Press, New York.

Szepannek G (2016). ***clustMixType**: k-Prototypes Clustering for Mixed Variable-Type Data*. R package version 0.1-16, URL https://CRAN.R-project.org/package=clustMixType.

The MPI Forum (2015). *MPI: A Message-Passing Interface Standard, Version 3.1*. Accessed October 2016, URL http://mpi-forum.org/.

Theußl S, Feinerer I, Hornik K (2012). "A **tm** Plug-In for Distributed Text Mining in R." *Journal of Statistical Software*, **51**(1), 1–31. doi:10.18637/jss.v051.i05.

Tibshirani R, Walther G (2005). "Cluster Validation by Prediction Strength." *Journal of Computational and Graphical Statistics*, **14**(3), 511–528. doi:10.1198/106186005x59243.

Tibshirani R, Walther G, Hastie T (2001). "Estimating the Number of Clusters in a Data Set via the Gap Statistic." *Journal of the Royal Statistical Society B*, **63**(2), 411–423. doi:10.1111/1467-9868.00293.

Titterington DM, Smith AFM, Makov UE (1985). *Statistical Analysis of Finite Mixture Models*. John Wiley & Sons, Chichester.

Venkataraman S (2013). ***SparkR**: R Frontend for **Spark***. Package version 0.1, URL http://amplab-extras.github.io/SparkR-pkg.

White T (2009). ***Hadoop**: The Definitive Guide*. 1st edition. O'Reilly Media.

Xie Y (2015). ***knitr**: Elegant, Flexible and Fast Dynamic Report Generation with R*. Accessed August 2016, URL http://yihui.name/knitr.

Yoo AB, Jette MA, Grondona M (2003). "SLURM: Simple Linux Utility for Resource Management." In *Job Scheduling Strategies for Parallel Processing: 9th International Workshop, JSSPP 2003, Seattle*, pp. 44–60. Springer-Verlag, Berlin, Heidelberg. Revised Paper.

A. KAMILA Hadoop submission file

The main SLURM batch submission script `kamila.slurm` should be modified to fit your computing setup and data. The first set of choices involve options for the SLURM workload manager; consult the `sbatch` documentation (SLURM Team 2016) for further information.

The remainder of the user-specified options are BASH variables. The first three variables are the file names of the three data files described in Section 6.1:

- `DATANAME`: the file name of the primary CSV data file.
- `SUBSAMP_DATANAME`: the file name of the sub-sampled data file.
- `CATINFO_DATANAME`: the file name of the TSV file.

The fourth variable, `DATADIR`, is the directory of the data files, i.e, the full path of the data file is `./$DATADIR/$DATANAME`. The remainder of the environment variables control the behavior of the KAMILA algorithm:

- `INITIAL_SEED`: integer. A random seed for the random number generator for reproducibility.
- `NUM_CLUST`: integer. KAMILA partitions the data into this many clusters.
- `NUM_MAP`: integer. The default number of map tasks, passed to **Hadoop** streaming option `mapred.map.tasks`.
- `NUM_REDUCE`: integer. The default number of reduce tasks, passed to **Hadoop** streaming option `mapred.reduce.tasks`.
- `NUM_CHUNK`: integer. Each cluster is split into this many chunks, and each chunk is assigned its own key (i.e., about `NUM_CLUST` $\times$ `NUM_CHUNK` keys total). This is useful if the number of available reducers greatly exceeds `NUM_CLUST`.
- `NUM_INIT`: integer. The number of distinct initializations should be used.
- `MAX_NITER`: integer. The maximum number of initializations computed for each distinct initialization.
- `EPSILON_CON` and `EPSILON_CAT`: positive reals. Parameters controlling the stopping rule. The closer to zero the more stringent the rule and thus the more likely each initialization will simply run for `MAX_NITER` iterations. The run is stopped if the summed absolute deviation of the centroid parameters in both the continuous and categorical variables are less than `EPSILON_CON` and `EPSILON_CAT`, respectively, from one iteration to the next. A reasonable value is the total deviation you would accept in the estimated centroid parameters relative to the true parameters, which depends on the data and the user's analysis needs. See the software paper cited above for more information.
- `RBIN_HOME`: character. File path to R, e.g., `/home/blah/R/R-3.x.x/bin`.

A.1. KAMILA Hadoop output file structure

Output files are stored in the directory `output-kamila-####`, where the `####` denotes the job submission number. The file structure is organized as follows:

```
output-kamila-####/
    best_run/
    run_1/
        stats/
        iter_1/
        iter_2/
        iter_3/
        ...
    run_2/
        stats/
        iter_1/
        iter_2/
        iter_3/
        ...
    run_3/
    ...
```

The file `output-kamila-####/best_run/allClustQual.txt` contains a list of the objective criterion values used to select the best run. The directory `output-kamila-####/run_i/` contains information on the i-th run; within each run's directory `iter_j/` stores information on the j-th run. The primary results files, `output-kamila-####/run_i/stats/finalRunStats.RData`, contain two R lists: `runSummary` and `clustSummary`.

The `output-kamila-####/run_i/iter_j/` directories contain the centroids of the current run/iteration stored as an `RData` file, along with other output from the reducers for that iteration.

The `runSummary` object contains overall summary statistics for the clustering:

- `clusterQualityMetric`: the quantity used to select the best solution over all the runs.
- `totalEucDistToCentroid`: the total Euclidean distance from each continuous point to its assigned centroid.
- `totalCatLogLik`: the log-likelihood of the categorical centroids with respect to the categorical variables.

The `clustSummary` list contains one element for each cluster. Each cluster's element contains a list of length five with the elements:

- `count`: the number of observations assigned to this cluster.
- `totalDistToCentroid`: the total Euclidean distance from each member of the current cluster to the centroid (along the continuous variables only).
- `totalCatLogLik`: the log-likelihood of the categorical centroids of the current cluster with regard to the categorical variables.

- `con`: means, minima, and maxima along each continuous variable for members of the current cluster.
- `catfreq`: frequencies of observations of each level of each categorical variable within the current cluster. Frequencies are calculated by counting the number of observations in the current cluster at a particular level of a particular variable and dividing by the number of observations in the current cluster.

A.2. Reading Hadoop results files into R

The document `Rnw/kamilaSummary.Rnw` (written as a **knitr** document; Xie 2015) included in the `kamilaStreamingHadoop` repository provides code for generating plots and tables of summary statistics from the output objects described in Section A.1. The context and execution of this script is described in Section 6.1. In the interest of brevity we do not discuss every line of `Rnw/kamilaSummary.Rnw` comprehensively, but highlight certain sections of the code that illustrate strategies for querying **SQLite** data bases and iterating through the **Hadoop** results directory.

We begin by obtaining summary statistics from the **SQLite** data base with path and filename stored in the variable `dataBaseName`.

```
R> suppressMessages(library("RSQLite"))
R> sqlite <- dbDriver("SQLite")
R> currentDb <- dbConnect(sqlite, dataBaseName)
R> nLines <- dbGetQuery(currentDb, "SELECT count(*) FROM finalDataSet")
R> nLines <- as.integer(nLines)
R> conVarInfo <- dbGetQuery(currentDb, "SELECT * FROM conStats")
R> numConVar <- nrow(conVarInfo)
R> catVarInfo <- dbGetQuery(currentDb, "SELECT * FROM catSummary")
R> levCounts <- as.integer(unlist(strsplit(as.vector(catVarInfo$LevCounts),
+    split = ",")))
R> levPcts <- levCounts / nLines * 100
R> catInfoTable <- data.frame(VariableName = with(catVarInfo, rep(VarName,
+    times = NumLev)), LevelName = unlist(strsplit(as.vector(
+    catVarInfo$LevNames), split = ",")), Counts = levCounts,
+    Percent = levPcts)
```

Table 1 was then generated from the `conVarInfo` data frame using a call to the `xtable` function in the **xtable** package (Dahl 2016). Tables 2, 3, and 4 were generated similarly using the `catInfoTable` data frame (split up due to length).

We construct Figure 6 by projecting a subset of the data onto its first two principal components, with data points colored by final cluster membership. We assume that this subset has already been created and stored as a separate CSV file with path and file name `subsampleDataFile`. Upon this plot we superimpose the cluster centroids at each iteration of the clustering procedure, labeled by iteration number. This is accomplished with the following code.

```
R> suppressMessages(library("MASS"))
R> subsamp <- read.csv(subsampleDataFile)
```

```
R> datMeans <- colMeans(subsamp)
R> datSds <- apply(subsamp, 2, sd)
R> datSds[datSds == 0] <- 1
R> subsamp <- scale(subsamp, center = datMeans, scale = datSds)
R> subsampCon <- subsamp[, 1:numConVar, drop = FALSE]
R> svdDat <- svd(subsampCon)
R> transMat <- ginv(diag(svdDat$d) %*% t(svdDat$v))
R> transMat <- rbind(transMat, matrix(0, nrow = ncol(subsamp) - numConVar,
+    ncol = ncol(transMat)))
R> JOBID <- "1234567"
R> outputFileName <- paste("output-kamila-", JOBID, sep = "")
R> centroidFileNames <- list.files(file.path("..", outputFileName,
+    paste("run_", best_run, sep = "")), pattern = "currentMeans_i",
+    recursive = TRUE, full.names = TRUE)
R> centroidFileNames <- as.vector(centroidFileNames)
R> centroidFileNames <- centroidFileNames[order(nchar(centroidFileNames),
+    centroidFileNames)]
R> allCentroids <- lapply(centroidFileNames, FUN = function(ff) {
+      if (exists("myMeans")) rm(myMeans)
+      load(ff)
+      dd <- t(sapply(myMeans, function(elm) elm$centroid))
+      dd <- scale(dd, center = datMeans, scale = datSds)
+      rownames(dd) <- paste("Clust", 1:nrow(dd))
+      colnames(dd) <- paste("Dim", 1:ncol(dd))
+      return(dd)
+    })
R> thisNumIter <- length(allCentroids)
R> thisNumClust <- nrow(allCentroids[[1]])
R> myColors <- rainbow(thisNumClust)
R> myColorsAlpha <- rainbow(thisNumClust, alpha = 0.5)
R> datColors <- t(apply(subsamp, 1, function(rr) {
+      which.min(as.matrix(dist(rbind(rr,
+        allCentroids[[thisNumIter]])))[-1, 1])
+    }))
R> plot(svdDat$u, xlab = "Principal Component 1",
+    ylab = "Principal Component 2", col = myColorsAlpha[datColors])
R> for (i in 1:thisNumIter) {
+    transCentroids <- allCentroids[[i]] %*% transMat
+    points(transCentroids, pch = 19, col = myColors, cex = 2.5)
+    points(transCentroids, pch = 1, col = "black", cex = 2.5, lwd = 2)
+    text(transCentroids, labels = i-1, cex = 0.8)
+  }
R> legend(x = "topright", title = "Cluster #", pch = 19, col = myColors,
+    legend = 1:thisNumClust)
```

Affiliation:

Alexander H. Foss, Marianthi Markatou
Department of Biostatistics
University at Buffalo
706 Kimball Tower, United States of America
E-mail: ahfoss@buffalo.edu, markatou@buffalo.edu

Journal of Statistical Software http://www.jstatsoft.org/
published by the Foundation for Open Access Statistics http://www.foastat.org/
February 2018, Volume 83, Issue 13 *Submitted:* 2016-08-28
doi:10.18637/jss.v083.i13 *Accepted:* 2017-05-05