Journal of Statistical Software

August 2016, Volume 72, Issue 5. doi: 10.18637/jss.v072.i05

RSKC: An R Package for a Robust and Sparse K-Means Clustering Algorithm

Yumi Kondo
The University of British Columbia

Matias Salibian-Barrera
The University of British Columbia

Ruben Zamar
The University of British Columbia

Abstract

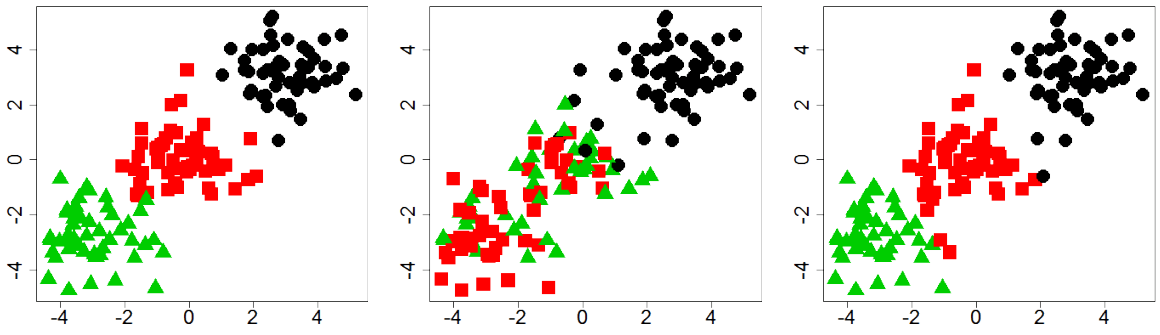
Witten and Tibshirani (2010) proposed an algorithim to simultaneously find clusters and select clustering variables, called sparse K-means (SK-means). SK-means is particularly useful when the dataset has a large fraction of noise variables (that is, variables without useful information to separate the clusters). SK-means works very well on clean and complete data but cannot handle outliers nor missing data. To remedy these problems we introduce a new robust and sparse K-means clustering algorithm implemented in the R package **RSKC**. We demonstrate the use of our package on four datasets. We also conduct a Monte Carlo study to compare the performances of RSK-means and SK-means regarding the selection of important variables and identification of clusters. Our simulation study shows that RSK-means performs well on clean data and better than SK-means and other competitors on outlier-contaminated data.

Keywords: K-means, robust clustering, sparse clustering, trimmed K-means.

1. Introduction

K-means is a very popular clustering method introduced by Steinhaus (1956) and popularized by MacQueen (1967). K-means is conceptually simple, optimizes a natural objective function, and is widely implemented in statistical packages. Datasets may contain two types of variables: *clustering variables* and *noise variables*. Clustering variables change their behavior from cluster to cluster while noise variables behave similarly across clusters. It is easy to construct examples where K-means breaks down in the presence of a large fraction of noise variables. Furthermore, it may be of interest to simultaneously find the clusters and a small number of variables which are sufficient to uncover the cluster structure.

Witten and Tibshirani (2010) proposed an algorithm called *sparse K-means (SK-means)*



(a) True cluster labels (b) K-means (c) SK-means

Figure 1: Scatter plots of the data with respect to their clustering features.

which simultaneously finds the clusters and the *clustering variables*. To illustrate the performance of SK-means when there is a large number of noise features we generate $n = 300$ observations with $p = 1000$ features. The observations follow a multivariate normal distribution with covariance matrix equal to the identity and centers at $(\mu, \mu, 0, \ldots, 0)^\top$ with $\mu = -3$, 0 and 3 for each cluster, respectively. Only the first two features are *clustering features*. Figure 1 contains the scatter plot of the data with respect to their clustering features. Colors and shapes indicate the true cluster labels. Panel (a) shows the true clusters and panels (b) and (c) show the partition obtained by K- and SK-means. K-means is unable to uncover the cluster structure while SK-means produces a good partition.

SK-means cleverly exploits the fact that commonly used dissimilarity measures (e.g., squared Euclidean distance) can be additively decomposed into p terms, each depending on a single variable. Given a cluster partition $\mathcal{C} = (C_1, C_2, \ldots, C_K)$, the between-cluster dissimilarity measure $B(\mathcal{C})$ satisfies

$$B(\mathcal{C}) = \sum_{j=1}^{p} B_j(\mathcal{C}) \,,$$

where $B_j(\mathcal{C})$ solely depends on the jth variable. The basic idea is to assign weights to each variable and base the clustering algorithm on the corresponding weighted dissimilarity measure. Given p non-negative weights $\mathbf{w} = (w_1, w_2, ..., w_p)^\top$ consider the *weighted between-cluster* dissimilarity measure

$$B(\mathbf{w}, \mathcal{C}) = \sum_{j=1}^{p} w_j B_j(\mathcal{C}) = \mathbf{w}^\top \mathbf{B}(\mathcal{C}) \,, \tag{1}$$

where

$$\mathbf{B}(\mathcal{C}) = \begin{pmatrix} B_1(\mathcal{C}) \\ B_2(\mathcal{C}) \\ \vdots \\ B_p(\mathcal{C}) \end{pmatrix}.$$

SK-means searches for the pair $(\mathbf{w}^*, \mathcal{C}^*)$ that maximizes (1) subject to the constraints

$$\sum_{i=1}^{p} w_j^2 \leq 1 \quad \text{and} \quad \sum_{i=1}^{p} w_j \leq l \,,$$

for a given value of l between 1 and $\sqrt{p}$. In other words, SK-means performs a regularized (LASSO-type) version of K-means.

Sections 2 and 4 show that a very small fraction of outliers – e.g., 1% of outlying observations one out of several hundred features – can drastically upset the performance of SK-means. To remedy this problem we propose a robustified alternative called robust and sparse K-means (RSK-means). Our simulation study and real data examples show that RSK-means works well with clean and contaminated data.

The rest of the paper is organized as follows. Section 2 describes RSK-means. Section 3 illustrates the implementation of our method in the **RSKC** package using four real datasets. Section 4 reports the results of our simulation study and Section 5 provides concluding remarks.

2. Robust and Sparse K-Means

The K-means clustering algorithm finds K clusters $C_1, \ldots, C_K$ that minimize the within-clusters sum of squares

$$\min_{C_1,\ldots,C_K} \left\{ \sum_{k=1}^{K} \frac{1}{n_k} \sum_{i,i' \in C_k} d_{i,i'} \right\}, \tag{2}$$

where $C_1, \ldots, C_K$ are the disjoint sets of cluster indices, n_k is the number of observations in the kth cluster, and

$$d_{i,i'} = \sum_{j=1}^{p} d_{i,i',j}$$

is the (additive) dissimilarity measure between the i and i' observations. When our observations are vectors $\mathbf{x}_1, \ldots, \mathbf{x}_n$ in $\mathbb{R}^p$ and $d_{i,i'}$ is the squared Euclidean distance between $\mathbf{x}_i$ and $\mathbf{x}_{i'}$ we have $d_{i,i',j} = (x_{i,j} - x_{i',j})^2$, $j = 1, \ldots, p$ and

$$\sum_{k=1}^{K} \frac{1}{n_k} \sum_{i,i' \in C_k} \|\mathbf{x}_i - \mathbf{x}_{i'}\|^2 = \sum_{k=1}^{K} \sum_{j \in C_k} \|\mathbf{x}_j - \boldsymbol{\mu}_k\|^2 ,$$

where $\boldsymbol{\mu}_k$ is the sample mean of the observations in the k-th cluster. A popular algorithm (see Lloyd 1982) to find local solutions to Equation 2 is as follows. First randomly select K initial "centers" $\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_K$ and iterate the following two steps until convergence:

(a) Given cluster centers $\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_K$, assign each point to the cluster with the closest center.

(b) Given a cluster assignment, update the cluster centers to be the sample mean of the observations in each cluster.

Although this algorithm decreases the objective function at each iteration it may be trapped in a local minima. Hence, one tries different starting points and the best solution is returned.

Cuesta-Albertos, Gordaliza, and Matran (1997) proposed a modification of this algorithm in order to obtain outlier-robust clusters. The main idea is to replace step (b) by

(b') Given a cluster assignment, trim $\alpha 100\%$ of the observations with the largest distance to their cluster centers, and update the cluster centers to the sample mean of the remaining observations in each cluster.

Since the total sum of squares

$$\frac{1}{n}\sum_{i=1}^{n}\sum_{i'=1}^{n} d_{i,i'}$$

does not depend on the cluster assignments, minimizing the within-cluster sum of squares is equivalent to maximizing the between-cluster sum of squares

$$\frac{1}{n}\sum_{i=1}^{n}\sum_{i'=1}^{n} d_{i,i'} \; - \; \sum_{k=1}^{K}\frac{1}{n_k}\sum_{i,i'\in C_k} d_{i,i'} \; = \; \sum_{j=1}^{p}\left\{\frac{1}{n}\sum_{i=1}^{n}\sum_{i'=1}^{n} d_{i,i',j} \; - \; \sum_{k=1}^{K}\frac{1}{n_k}\sum_{i,i'\in C_k} d_{i,i',j}\right\}.$$

Witten and Tibshirani (2010) introduced non-negative weights w_j, $j = 1, \ldots, p$ for each feature and solved

$$\max_{C_1,\ldots,C_K,\mathbf{w}} \; \sum_{j=1}^{p} w_j \left\{\frac{1}{n}\sum_{i=1}^{n}\sum_{i'=1}^{n} d_{i,i',j} \; - \; \sum_{k=1}^{K}\frac{1}{n_k}\sum_{i,i'\in C_k} d_{i,i',j}\right\}, \tag{3}$$

subject to $\|\mathbf{w}\|_2 \leq 1$, $\|\mathbf{w}\|_1 \leq l$ and $w_j \geq 0$, $j = 1, \ldots, p$, where $l > 1$ determines the degree of sparcity (in terms of non-zero weights) of the solution. The optimization problem in Equation 3 can be solved by iterating the following steps:

(a) Given weights $\mathbf{w}$ and cluster centers $\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_K$ solve

$$\min_{C_1,\ldots,C_K} \; \sum_{k=1}^{K}\sum_{i\in C_k}\sum_{j=1}^{p} w_j \left(x_{i,j} - \mu_{k,j}\right)^2,$$

which is obtained by assigning each point to the cluster with closest center using weighted Euclidean squared distances.

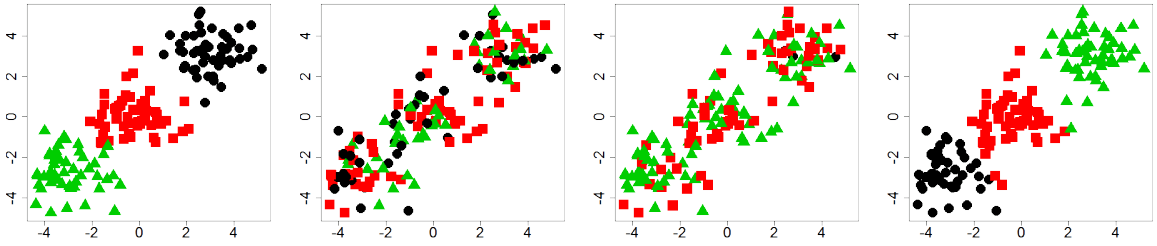
(b) Given weights $\mathbf{w}$ and cluster assignments $C_1, \ldots, C_K$, update the cluster centers to be the weighted sample mean of the observations in each cluster.

(c) Given cluster assignments $C_1, \ldots, C_K$ and centers $\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_K$ solve

$$\max_{\|\mathbf{w}\|_2\leq 1, \|\mathbf{w}\|_1\leq l} \; \sum_{j=1}^{p} w_j \, B_j(C_1,\ldots,C_K),$$

where $B_j(C_1,\ldots,C_K) = \sum_{i=1}^{n}\sum_{i'=1}^{n} d_{i,i',j} \; - \; \sum_{k=1}^{K}\frac{1}{n_k}\sum_{i,i'\in C_k} d_{i,i',j}$. There is a closed form expression for the vector of weights that solves this optimization problem (see Witten and Tibshirani 2010).

A naive first approach to robustify this algorithm is to use trimmed K-means with weighted features, and then optimize the weights using the trimmed sample. In other words, to replace step (b) above with (b') where $B_j(C_1,\ldots,C_K)$, $j = 1, \ldots, p$ are calculated without the observations flagged as outliers. A problem with this approach is that if an observation is outlying in a feature that recieved a small weight in steps (a) and (b'), then this observation might not be trimmed. In this case, the variable where the outlier is more evident will receive a very high weight in step (c) (because this feature will be associated with a very large B_j). This may in turn cause the weighted K-means steps above to form a cluster containing this single point, which will then not be downweighted (since its distance to the cluster center will be zero). This phenomenom is illustrated in the following example.



(a) True cluster labels (b) Naive robustification (c) SK-means (d) RSK-means

Figure 2: Scatter plots of the synthetic simulated data example, with respect to their clustering features. The dataset includes 3 outliers.

Simulated data example: We generate $n = 300$ 5-dimensional observations arranged in three clusters of equal size. The observations have multivariate normal distributions with covariance matrix equal to the identity and centers at $(\mu, \mu, 0, 0, 0)$ with $\mu = -3, 0$ and 3 for each cluster, respectively. Only the first 2 features contain information on the clusters. Figure 2 contains the scatterplot of these 2 clustering features. Colors and shapes indicate the true cluster labels.

To illustrate the problem mentioned above, we replace the 4th and 5th entries of the first 3 observations with large outliers. The naive robustification described above returns a disappointing partition because it fails to correctly identify the clustering features, placing all the weight on the noise ones. The result is shown in Figure 2 (b). As expected, SK-means also fails in this case assigning all the weights to the noise features and forming one small cluster with the 3 outliers (see Figure 2 (c)). Finally, Figure 2 (d) shows the partition found by our RSK-means algorithm which is described below.

The key step in our proposal is to use two sets of trimmed observations which we call the weighted and unweighted trimmed sets. By doing this, zero weights will not necessarily mask outliers in the noise features. Our algorithm can be described as follows

(a) Perform trimmed K-means on the weighted dataset:

(i) Given weights $\mathbf{w}$ and cluster centers $\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_K$ solve

$$\min_{C_1,\ldots,C_K} \sum_{k=1}^{K} \sum_{i \in C_k} \sum_{j=1}^{p} w_j \left(x_{i,j} - \mu_{k,j}\right)^2 ,$$

which is obtained by assigning each point to the cluster with closest center using weighted Euclidean squared distances.

(ii) Given weights $\mathbf{w}$ and cluster assignments, trim the $\alpha 100\%$ observations with largest distance to their cluster centers, and update the cluster centers to the sample mean of the remaining observations in each cluster.

(iii) Iterate the two steps above until convergence.

(b) Let O_W be the subscripts of the $\alpha 100\%$ cases labelled as outliers in the final step of the weighted trimmed K-means procedure above.

(c) Using the partition returned by trimmed K-means, calculate the unweighted cluster centers $\tilde{\boldsymbol{\mu}}_k$, $k = 1, \ldots, K$. For each observation $\mathbf{x}_i$, let $\tilde{d}_i$ be the unweighted distance to its cluster center, i.e., $\tilde{d}_i = \|\mathbf{x}_i - \tilde{\boldsymbol{\mu}}_k\|^2$ where $i \in C_k$. Let O_E be the subscripts of the $\alpha 100\%$ largest distances $\tilde{d}_i$.

(d) Form the set of trimmed points $O = O_W \bigcup O_E$.

(e) Given cluster assignments C_1, ..., C_K, centers $\boldsymbol{\mu}_1$, ..., $\boldsymbol{\mu}_K$ and trimmed points O, find a new set of weights $\mathbf{w}$ by solving

$$\max_{\|\mathbf{w}\|_2 \leq 1, \|\mathbf{w}\|_1 \leq l} \sum_{j=1}^{p} w_j \, B_j(C_1, \ldots, C_K, O) \,,$$

where $B_j(C_1, \ldots, C_K, O)$, $1 \leq j \leq p$, are calculated without the observations in O.

This algorithm is called RSK-means and it is implemented in the R package **RSKC** (Kondo 2016).

RSK-means requires the selection of three parameters: the L_1 bound, l, the trimming proportion, α, and the number of clusters, K. In our experience the choice $\alpha = 0.10$ works well for most applications. The choice of l determines the degree of sparcity and hence l can be chosen to achieve a desired number of selected features. In practice, one can also consider several combinations of values for these two parameters and compare the results. The problem of selecting K is outside of the scope of this paper, and has been discussed extensively in the literature for standard K-means clustering; see Milligan and Cooper (1985), Kaufman and Rousseeuw (1990), Sugar and James (2003), and Tibshirani and Walther (2005). To select the number of clusters K we recommend using the Clest algorithm of Dudoit and Fridlyand (2002) which is the default method implemented in the **RSKC** package.

Missing values are a challenging difficulty for clustering algorithms. All commonly used implementations of K-means (including SK-means and trimmed K-means) do not work if there are missing values in the data. However, even in moderately high-dimensional datasets it is not uncommon to find that a large proportion of observations has a missing value in at least one variable. Hence, using only complete cases may result in a significant loss of information. Unfortunately, standard imputation methods will typically not work well in a clustering application, since missing values may depend on the cluster to which the observation belongs, and these are of course unknown. The **RSKC** package implements the following adjustment to deal with potential missing values. Consider the i-th observation, $\mathbf{x}_i = (x_{i,1}, \ldots, x_{i,p})^\top$ and the center of the k-th cluster, $\boldsymbol{\mu}_k = (\mu_{k,1}, \ldots, \mu_{k,p})^\top$. Let $\mathcal{M}_{i,k} \subset \{1, 2, \ldots, p\}$ be the set of coordinates where either $\mathbf{x}_i$ or $\boldsymbol{\mu}_k$ have a missing value. Our proposed algorithm scales the (weighted) distances according to the number of missing features. More specifically, the adjusted distance between $\mathbf{x}_i$ and $\boldsymbol{\mu}_k$ is

$$S_{i,k} \sum_{j \notin \mathcal{M}_{i,k}} w_j (x_{i,j} - \mu_{k,j})^2$$

where $S_{i,k} = \sum_{j=1}^{p} w_j / \sum_{j \notin \mathcal{M}_{i,k}} w_j$, and w_j, $1 \leq j \leq p$ are the sparsity weights.

3. Using the RSKC Package

In this section we illustrate the functionality available in the **RSKC** package (version 2.4.2) through the analysis of four real datasets. The main function in the package is called `RSKC` and it implements the RSK-means algorithm described in Section 2. The function `RSKC` has five main arguments:

- `d`: An N by p data matrix, where each row corresponds to a data point $\mathbf{x}_i$, $1 \leq i \leq N$.
- `ncl`: The number of clusters to be identified.
- `alpha`: The proportion of observations to be trimmed, $0 \leq$ `alpha` < 0.50.
- `L1`: The upper bound for the L_1 constraint for the vector of weights $\mathbf{w}$, $L_1 > 1$.
- `nstart`: The number of initial partitions used in step (a).

In addition the function `RSKC` runs K-means when `alpha` $= 0$ and `L1` $=$ `NULL`; Sparse K-means when `alpha` $= 0$ and `L1` $\neq$ `NULL`; and trimmed K-means when `alpha` > 0 and `L1` $=$ `NULL`.

Several other R packages implement K-means type algorithms. For example, the function `kmeans` in **stats** (R Core Team 2016), the trimmed K-means algorithm in **trimcluster** (Hennig 2012), and the SK-means algorithm in **sparcl** (Witten and Tibshirani 2013). None of these can handle missing values.

3.1. Optical recognition of handwritten digits

The dataset was downloaded from the UCI Machine Learning Repository (Bache and Lichman 2013) and consists of $n = 1797$ digitized images of the numbers 0 to 9 handwritten by 13 subjects. The number of observations for each digit varies between 174 and 183. The raw observations are 32 x 32 bitmaps, which were divided into 64 non-overlapping blocks of 4 x 4 bits. For each block we observe the number of "black" pixels. Hence, each image is represented by $p = 64$ variables recording the count of pixels in each block, taking integer values between 0 and 16. This dataset is included in the **RSKC** package in the object `optd`. Its row names identify the true digit in the image and the column names identify the position of each block in the original bitmap. The following code loads the data into R:

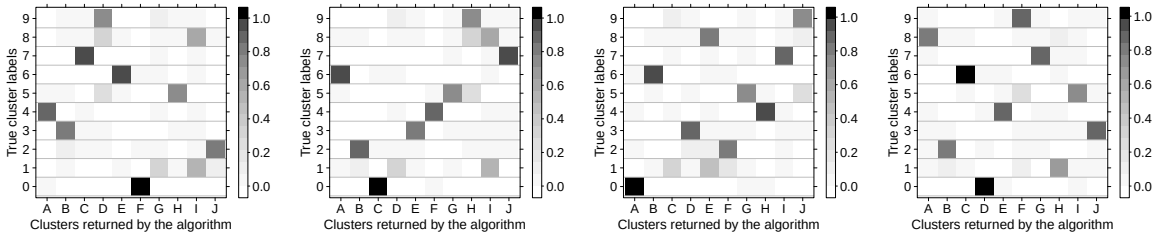
```
R> library("RSKC")
R> data("optd")
R> truedigit <- rownames(optd)
```

We run K-means, trimmed K-means, SK-means and RSK-means on the data ignoring the known labels (*confirmatory* cluster analysis). Since the digits tend to appear in the center of the images, the left and right ends are often left blank. We select an L_1 bound that returns around 16 weights equal to zero, that corresponds to the 16 blocks on both sides of each picture. The trimming proportion α is set to 0.10, which works well in most cases. The observations trimmed by `RSKC` are not necessarily removed from the cluster partition as they can be assigned to the closest cluster. We use the classification error rate (CER, Chipman and Tibshirani 2005) to measure agreement between the true labels and the cluster assignments returned by each algorithm. Given two partitions of a dataset, the CER is the proportion of

pairs of observations that are together in one partition and apart in the other. Note that CER is one minus the Rand index (Rand 1971). We also compute a measure of the probability of the correct classification of each partition. Given the clusters identified by an algorithm, we compute the proportion of cases from each original class that are assigned to each cluster. The largest of these proportions is the *sensitivity* of that class. This number represents the proportion of cases of each class that remained "together" in the partition returned by the clustering algorithm. We also report the label of the cluster where this majority of points was assigned. Given two vectors of labels `label1` and `label2` the sensitivities of each class in `label2` with respect to the estimated partition in `label2` can be computed using `Sensitivity(label1, b)`.

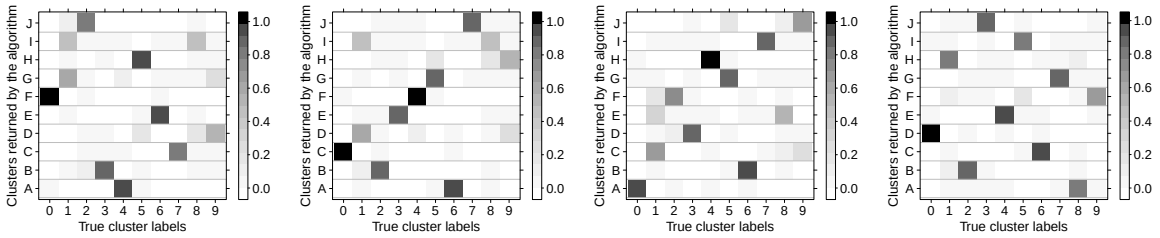
The following code runs K-means, trimmed K-means, SK-means and RSK-means and computes the CER and the sensitivity of the true classes (digits), for each of the returned partitions. The column labels represent the true classes.

```
R> re <- list(K = list(), TRIM = list(), SK = list(), RSK = list());
R> options(digits = 3)
R> alphas <- c(0, 0.1, 0, 0.1)
R> set.seed(1)
R> for (imethod in 1:4) {
+    cat("\n", names(re)[imethod])
+    if (imethod %in% c(1, 2)) L1 <- NULL else L1 <- 5.7
+    r <- RSKC(optd, ncl = 10, alpha = alphas[imethod], L1 = L1,
+      nstart = 1000)
+    r$labels <- LETTERS[r$labels]
+    re[[imethod]] <- r
+    cat(": CER", CER(r$labels, truedigit), "\n")
+    print(Sensitivity(label1 = r$labels, label2 = truedigit)$prob)
+  }
 K: CER 0.0613
                        0  1  2  3  4  5  6  7  8  9
Sensitivity. (%)      99 54 83 84 90 75 98 98 57 78
Class label by label1.  F  I  J  B  A  H  E  C  I  D

 TRIM: CER 0.0617
                       0  1  2  3  4  5  6  7  8  9
Sensitivity. (%)      99 55 86 84 90 75 98 94 57 78
Class label by label1.  C  I  B  E  F  G  A  J  I  H

 SK: CER 0.0588
                       0  1  2  3  4  5  6  7  8  9
Sensitivity. (%)      99 45 80 88 92 77 98 91 79 77
Class label by label1.  A  E  F  D  H  G  B  I  E  J

 RSK: CER 0.0518
                       0  1  2  3  4  5  6  7  8  9
Sensitivity. (%)      98 66 84 89 89 77 98 88 79 85
Class label by label1.  D  H  B  J  E  I  C  G  A  F
```



(a) K-means (b) trimmed K-means (c) SK-means (d) RSK-means

Figure 3: Each panel depicts how well the corresponding cluster partition preserves the integrity of the true classes. In each row, darker shades of grey correspond to larger conditional cluster relative frequencies. Clearly RSK-means does the best job.



(a) K-means (b) trimmed K-means (c) SK-means (d) RSK-means

Figure 4: Each panel depicts the *true-class purity* of each cluster partition, that is, the conditional distribution of the true classes within each cluster. Darker grey shades correspond to larger true class conditional relative frequencies and so, ideally, there should be only one dark cell per row. Again, RSK-means does the best job.

RSK-means has the lowest CER and relatively large sensitivities for digits "1" (66%), "8" (79%) and "9" (85%). Ideally, all the cases in the true classes would remain together and each cluster would consist of cases coming from a single class.

Figure 3 displays the conditional relative frequencies of estimated clusters (columns) given each true class (rows). Darker shades correspond to higher conditional relative frequencies. Note that the largest relative frequency for the kth row is the reported sensitivity of kth true class. Figure 3 shows that K-means, trimmed K-means and SK-means split the digits "1", "8" and "9" into 2 or 3 different clusters, whereas in the RSK-partition the observations corresponding to each digit are mostly assigned to a single cluster.

The rows in Figure 4 depict conditional relative frequencies of the true classes given the estimated clusters. Note that clusters "D", "G" and "I" from K-means are relatively weak (no true class dominates). Similarly, clusters "D", "H" and "I", from trimmed K-means, clusters "C", "E", "F" and "J' from SK-means are rather weak. The weakest cluster in the partition from RSK-means is cluster "F", although it is fairly dominated by 9's.

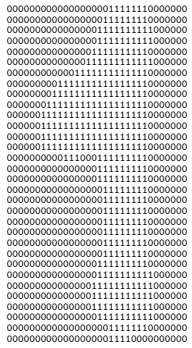
The sensitivity of the true digit 1 from SK-means is in particular low (45%) and Figure 3 shows that the true 1's are separated into three distinct clusters; "C", "E" and "F". These subjects tend to write 1 in three ways; straight vertical line with hat (namely Arial font style), straight line with hat and vertical line at bottom (namely Times New Roman – TNR – font style) and thick straight line style. SK-means classifies the Arial font observations

into a cluster containing observation with true digits 9, the TNR font observations into a cluster containing observations with true digits 2, and the straight line style observations into a cluster containing observations with true digits 8.

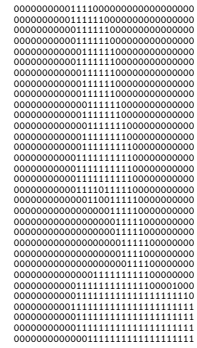
To observe the original 32 x 32 bitmap of any observation in data `optd`, the function `showbitmap` is useful. The following commands return the bitmaps of true digits 1 observations, which were classified into clusters labeled as "C", "E" or "F" by SK-means.

```
R> c9 <- which(re$SK$labels == "C" & truedigit == 1)
R> showbitmap(c9)
R> c2 <- which(re$SK$labels == "F" & truedigit == 1)
R> showbitmap(c2)
R> c8 <- which(re$SK$labels == "E" & truedigit == 1)
R> showbitmap(c8)
```

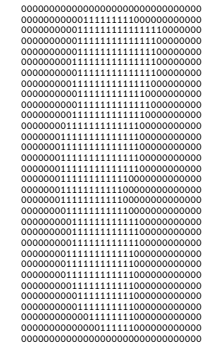
We present the partial outputs on the R console for each font style observations; Arial, TNR and straight line.



obs. 388 (Arial) classified with true-digit-9 obs.



obs. 973 (TNR) classified with true-digit-2 obs.



obs. 973 (straight) classified with true-digit-8 obs.

About 40% of the observations trimmed by RSK-means are 1's, many of these in turn the "Arial font" type. These 1's were later mistakenly assigned to a cluster dominated by 9's. Had they not been trimmed, these 1's would have affected the center of the cluster that contains most of the 9's. In other words, trimming these miss-classified observations results in a better separation between the clusters containing true 1's and true 9's.

It is interesting to note that, of those observations that were trimmed by RSK-means, about 40% of them are true 1's, and many of these are in turn of the "Arial font" type, which were later assigned to a cluster dominated by 9's. Had they not been trimmed, these 1's might have affected the centre of the cluster that contains most of the 9's. In other words, trimming these miss-classified observations results in a better separation between the clusters containing true 1's and true 9's.

The choice of L_1 bound $L_1 = 5.7$ returns 19 weights equal to zero. These are 19 blocks that were not used in the construction of the clusters by RSK-means and SK-means. The following script displays the names of the variables which received zero weights:

```
R> names(which(re$RSK$weights == 0))
```

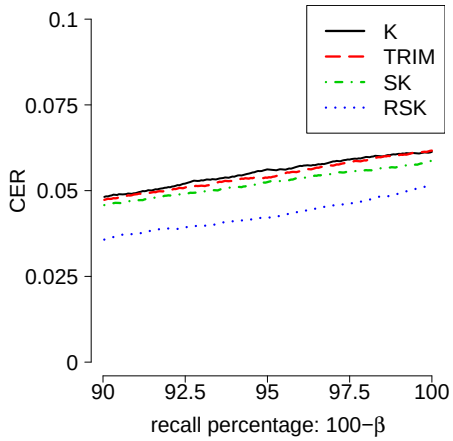


Figure 5: Recall-performance curves for the different cluster partitions. We plot the percentages of remained cases versus the corresponding CER obtained using only the assigned cases.

```
 [1] "1:1" "1:2" "1:8" "2:1" "2:8" "3:1" "3:8" "4:1" "4:8" "5:1" "5:8" "6:1"
[13] "6:8" "7:1" "7:2" "7:8" "8:1" "8:2" "8:8"
```

These labels have the format `row:column` and indicate the relative position of the block in the figure. We see that the whole first and eighth columns (left and right side of the picture) were ignored, plus 3 blocks in the 2nd column (in the top and last two rows). As expected, the RSK-means did not use the blocks on the sides of the figures.

It is a good practice to flag potential outliers identified by each method and assess the method performances after these potential outliers have been removed. A good method should achieve a good performance (very small CER, say) for a large *recall percentage*. For the considered algorithms, we computed the weighted Euclidean distances from each case to its cluster center. For non-sparse methods, all the feature weights are set to 1. For robust methods, cluster centers are computed without trimmed cases in weighted distances. For each clustering method, we sorted the weighted distances in increasing order and then computed CER without $\beta\%$ of the cases with the largest weighted distances (removed as potential outliers). Figure 5 shows the plot of the percentage of remained cases (that is, $100 - \beta\%$) versus CER without the $\beta\%$ most extreme cases. Note that in the case of the robust procedures, the top 10% of the removed cases are those originaly trimmed by the robust algorithms in order to decide the cluster centers and the cluster partitions. For all the methods, we observe an overall increasing trend of CER as the recall percentage increases, indicating that most of the removed cases are in fact outliers which are indeed misclassified. RSK-means returns the lowest CER for all the considered recall percentages.

In summary, RSK-means identifies the true given handwritten digits better than the other considered algorithms. It is also interesting to note that it does so by ignoring the left and right sides of the pictures, and that many trimmed "outliers" appear to be 1's, which look very similar to 9's.

Finally, to illustrate the use of the function `RSKC` on the missing values, we randomly replace 10% of entries in the datamatrix of `optd` by `NA`. Then RSK-means is fitted to this dataset with missing values as

```
R> Nentries <- length(optd)
R> set.seed(1)
R> optd[sample(1:Nentries, round(0.1 * Nentries), replace = FALSE)] <- NA
R> reNA <- RSKC(optd, ncl = 10, alpha = 0.1, L1 = 5.7, nstart = 1000)
```

The resulting partitions are nearly identical as indicated by the CER between the results RSK-means on the handwirtten-digits dataset with and without missing values:

```
R> CER(reNA$label, re$RSK$label)
```

```
[1] 0.02
```

The resulting partitions are reasonably similar.

3.2. Digits from a collection of Dutch utility maps

The dataset, available from the UCI Machine Learning Repository, was extracted from a collection of Dutch utility maps and consists of digitized handwritten numerals between 0 and 9. The object `DutchUtility` available in the **RSKC** package contains this dataset. There are 200 instances of each digit, for a total of $n = 2000$ images. The raw observations consist of 32 x 45 bitmaps, which are divided into a 16 x 15 grid of non overlapping blocks, each of size 2 x 3 bits. The number of "black" pixels in each block is recorded as a feature taking integer values between 0 and 6, hence each image is represented by $p = 240$ features.

As an previous example, we use K-means, trimmed K-means, SK-means and RSK-means to identify $K = 10$ clusters. The choice $L_1 = 12.4$ results in approximately 30 zeroes for the feature weights which we would expect will ignore 15 blocks on the left side and 15 blocks on right side of the images as in previous example. Again, the trimming proportion is set at 0.10. The CER and the sensitivity of the true classes (digits), for each method are computed as in previous example. The output is as follows.

```
 K: CER 0.0635
                         0  1  2  3  4  5  6  7  8  9
Sensitivity. (%)        96 84 96 92 53 94 70 70 84 90
Class label by label1.   B  J  G  I  C  H  F  A  B  D

 TRIM: CER 0.0526
                         0  1  2  3  4  5  6  7  8  9
Sensitivity. (%)        95 86 98 91 60 91 53 74 89 90
Class label by label1.   A  J  C  G  B  E  H  F  D  I

 SK: CER 0.0755
                         0  1  2  3  4  5  6  7  8  9
Sensitivity. (%)        96 81 51 92 57 91 62 69 86 90
Class label by label1.   F  G  H  B  C  E  I  J  F  A

 RSK: CER 0.0568
                         0  1  2  3  4  5  6  7  8  9
```

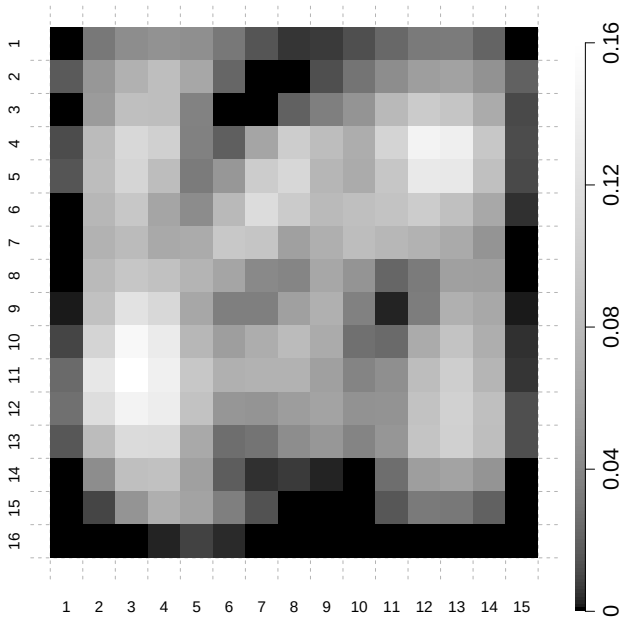


Figure 6: The weight allocations over the 16 x 15 reduced scale bitmap.

```
Sensitivity. (%)       93 83 91 92 60 91 60 71 89 90
Class label by label1.  F  J  E  H  D  G  B  I  C  A
```

K-means return class "B" which contains 96% of true 0's and 84% of 8's. Similarly, SK-means returns class "F" which contains 96% of true 0's and 86% of 8's. On the other hand, trimmed K-means and RSK-means create 10 clusters, each of which uniquely corresponds to a majority of observations from a single true digit group. Although the CER is almost the same between the trimmed K-means and RSK-means, the sensitivity of digit "2" of trimmed K-means is substantially larger than RSK-means, and the sensitivity of digit "6" of RSK-means is substantially larger than trimmed K-means. This could be due to the weights allocation. Figure 6 shows the weight allocations over the 16 x 15 reduced scale bitmap. RSK-means returns zero weights to 12 (out of 15) bottom pixels. As the horizontal line of the digit "2" is often written in the bottom pixels, treating these features as noise resulted in relatively poor performance of RSK-means. Figure 7 left panel shows 16 x 15 reduced scale bitmap of an observation with true digit "2" which was missclassified by RSK-means while classified correctly by trimmed K-means. We note that the original 32 x 45 bitmmaps was lost. The **RSKC** package contains the function `showDigit` which, given an observation index, returns the reduced scale bitmap and Figure 7 can be generated with

```
R> showDigit(467)
```

On the contrary, RSK-means outperforms trimmed K-means in terms of the sensitivity of digit "6" while trimmed K-means tends to misclassify observations with true digits "6" and "4". Observations with true digit "4" or "6" both equally (and indistinguishably) have high pixel in the center top reverse triangle region. The better distinction of observations with true digit "4" and "6" could be because RSK-means weights less this region (see Figure 12). Unlike

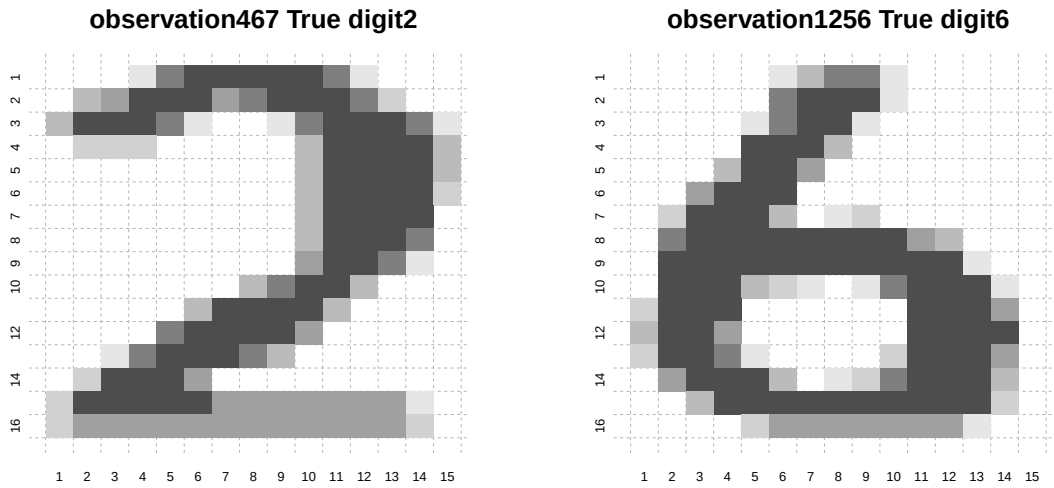


Figure 7: The 16 x 15 reduced scale bitmaps of an observation with true digit 2 which is missclassified by RSK-means and correctly classified by trimmed K-means (Left) and an observation with true digit 6 which is misclassified by trimmed K-means and correctly classified by RSK-means.

the handwritten digit, the digits from the collections of Dutch utility maps are not always centerd. In the absence of obvious noise features, the sparcity component of RSK-means did not contribute much to its good performance on this dataset. On the other hand its trimming component played an important role.

3.3. DBWorld e-mail

The next dataset contains $n = 64$ bodies of e-mails in binary "bag-of-words" representation. Filannino (2011) manually collected the dataset from the DBWorld mailing list between October 19 and 29 in 2011. The dataset is available from the UCI Machine Learning Repository and included in our package as object `DBWorld`. The DBWorld mailing list announces conferences, jobs, books, software and grants. Of the 64 e-mails in this dataset, 29 correspond to conference announcements. Filannino (2011) applied supervised learning algorithms to classify e-mails in two classes: "conference announcement" and "everything else".

A "bag-of-words" representation of a document consists of a vector of p zeroes and ones, where p is the number of words extracted from the entire corpus (the size of the dictionary). Typically these collections do not contain many common or highly unusual words. For example, the dictionary in this example does not contain words such as "the", "is" or "which", so-called stop words, and words that have less than 3 characters or more than 30 characters. The entry of the vector is 1 if the corresponding word belongs to the e-mail and 0 otherwise. The number of unique words in this dataset is $p = 4702$. The following code loads the data into R

```
R> library("RSKC")
R> data("DBWorld")
R> true <- rownames(DBWorld)
```

To compare the performance of the different K-means variants in identifying the conference

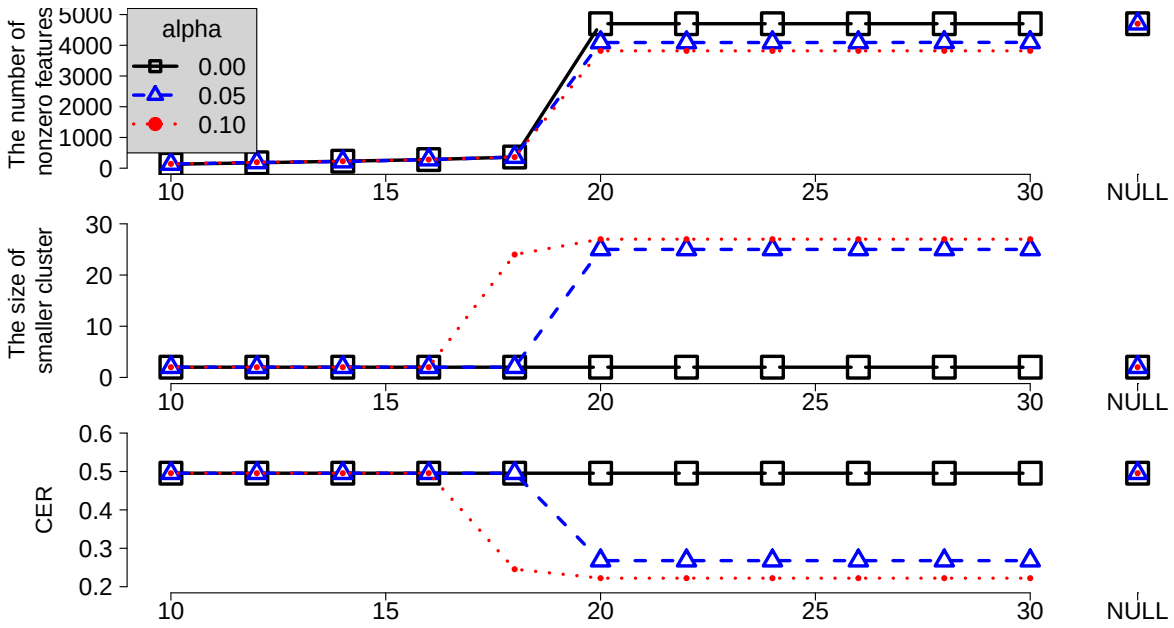


Figure 8: The numbers of nonzero features, the size of the smallest cluster and the CER performance for the considered algorithm for `DBWorld` data. The `NULL` L_1 parameter corresponds to the non-sparse methods.

announcements from the other documents, we run K-means, trimmed K-means, SK-means and RSK-means on the data ignoring the known labels. Since there is no information about the number of clustering features or the number of outliers that may be present in the data, we explore the results obtained with different values of the sparsity and trimming parameters. Specifically, we use $L_1 = 10, 12, \ldots, 30$ and $\alpha = 0.00, 0.05, 0.1$, always with $K = 2$. The following code runs the 4 algorithms for all combinations of L_1 and α and computes the CER performances. This code also returns the size of the smallest cluster and the number of nonzero weights for each procedure as well as the plots in Figure 8.

```
R> alphas <- c(0, 0.05, 0.1)
R> l.a <- length(alphas)
R> minL1 <- 10
R> maxL1 <- 30
R> L1NULL <- maxL1 + 2.5
R> L1s <- c(seq(minL1, maxL1, 2), L1NULL)
R> l.L <- length(L1s)
R> nonZero <- clustSize <- reCER <- matrix(NA, l.a, l.L)
R> temp <- rep(list(NULL), l.L)
R> names(temp) <- paste("L1", L1s, sep = "")
R> re.rskc <- rep(list(temp), l.a)
R> names(re.rskc) <- paste("alpha", alphas, sep = "")
R> set.seed(1)
R> for (ialpha in 1:l.a) {
+    for (iL in 1:l.L) {
+      cat("\nAlpha", alphas[ialpha], "L1", L1s[iL])
```

```
+      if (L1s[iL] == L1NULL ) L1 <- NULL else L1 <- L1s[iL]
+      re.rskc[[ialpha]][[iL]] <- rskc <- RSKC(DBWorld, L1 = L1, ncl = 2,
+        alpha = alphas[ialpha], nstart = 1000, silent = TRUE)
+      reCER[ialpha, iL] <- CER(rskc$label, true)
+      clustSize[ialpha, iL] <- min(sum(rskc$label == 1),
+        sum(rskc$label == 2))
+      nonZero[ialpha, iL] <- sum(rskc$weight > 0)
+    }
+  }
R> ylabs <- list(feat = "The number of \nnonzero features",
+    cluster = "The size of \nsmaller cluster", CER = "CER")
R> pchs <- c(0, 2, 19)
R> ltys <- 1:3
R> cex.axis <- 2
R> cexs <- c(4, 2.5, 0.5)
R> cols <- c("black", "blue", "red")
R> cex.ylab <- 1.25
R> lwd <- 3
R> par(mfrow = c(3, 1), oma = c(2, 0, 0, 0), mar = c(3, 9, 0, 0))
R> ylims <- list(feat = c(0, 5000), cluster = c(0, 30), CER = c(0.2, 0.6))
R> resp <- list(nonZero, clustSize, reCER)
R> for (iplot in 1:3) {
+    ys <- resp[[iplot]]
+    ylim <- ylims[[iplot]]
+    ylab <- ylabs[[iplot]]
+    plot(1, 1, xlab = "", ylab = "", ylim = ylim,
+      xlim = c(L1s[1], L1s[l.L] + 3), axes = FALSE)
+    axis(1, L1NULL, "NULL", cex.axis = cex.axis)
+    axis(1, pretty(L1s[-l.L]), cex.axis = cex.axis)
+    axis(2, pretty(ylim, n = 4), las = 2, cex.axis = cex.axis)
+    for (ialpha in 1:l.a) {
+      points(x = L1s[-l.L], y = ys[ialpha, -l.L], type = "b", lwd = lwd,
+        lty = ltys[ialpha], col = cols[ialpha], pch = pchs[ialpha],
+        cex = cexs[ialpha])
+      points(x = L1s[l.L], y = ys[ialpha, l.L], type = "b", lwd = lwd,
+        col = cols[ialpha], pch = pchs[ialpha], cex = cexs[ialpha])
+    }
+    mtext(side = 2, ylab, cex = cex.ylab, line = 5)
+    if (iplot == 1) legend("topleft", cex = 2, title = "alpha",
+      bg = "lightgray", legend = sprintf("%1.2f", alphas), lty = ltys,
+      col = cols, lwd = lwd, pch = pchs)
+  }
R> mtext(side = 1, "L1-tuning parameter", cex = cex.ylab, line = 3,
+    outer = TRUE)
```

Figure 8 contains three diagnostic plots of the partitions obtained with different sparsity and trimming levels. The horizontal axis is the value of L_1. We note that the robust fits (i.e., when

$\alpha > 0$) find two rather different partitions as we vary the level of sparsity. When $L_1 \leq 16$, one of the two clusters is formed by only two observations. This is the only partition found by the non-robust fit ($\alpha = 0$) for any level of L_1. Interestingly, the 2-document cluster contains e-mails numbers 26 and 50, which are identical conference announcements in the bag of word representation. Clearly, this partition is not satisfactory. To see this, type

```
R> mean(DBWorld[50,] == DBWorld[26,])
```

```
[1] 1
```

When we increase the amount of sparsity (e.g., $L_1 > 18$) both robust fits identify roughly the same two clusters with a much lower CER (bottom panel of Figure 8). The table of classification errors with respect to the true labels for the clusters found by RSK-means with $\alpha = 0.10$ and $L_1 = 20$ can be computed with the following commands:

```
R> ialpha <- which(alphas == 0.1)
R> iL <- which(L1s == 20)
R> rskc.label <- re.rskc[[ialpha]][[iL]]$label
R> table(rskc.label, true)
```

```
           true
rskc.label conference others
         1          5     32
         2         24      3
```

It is important to note that for this partition RSK-means trims e-mails 26 and 50, while the non-sparse trimmed K-means fits (with either $\alpha = 0.05$ or 0.10) do not. In other words, these potentially atypical documents are trimmed only by the robust and sparse methods – robustness alone does not seem to be sufficient. Finally, this partition agrees reasonably well with the true classes (see the CER plot in the bottom panel of Figure 8).

In order to investigate why the variables chosen by the sparse and robust method provide a better partition, we will show that the selected features seem to have a higher degree of association with the true classes than the other variables. To measure the association of words and true classes we use the chi-squared test for the corresponding 2-by-2 contigency table ("presence"/"absence" of the word and "conference announcement"/"other"). Of the 4702 words in the dictionary, 167 (3.6%) of them result in a p value of 0.05 or less. Although the sparse and robust RSK-means with $\alpha = 0.1$ and $L_1 = 20$ returns nonzero weights for 3819 words (Figure 8), most of the weight (95%) is assigned to only 371 variables. Of these, 74 (20%) are among those with a small chi-squared p value. In other words, many of the highly weighted words from RSK-means are indeed associated with the e-mail classes.

In summary, RSK-means with a relatively large sparsity parameter (L_1) performed the best in identifying conference announcement e-mails. This method assigns most weight to features that are highly associated with the true underlying classes. Furthemore, RSK-means trims the two e-mails that, if left unchecked, tend to form their own non-informative 2-point cluster.

3.4. Biometric identification of subjects with miniature inertial sensors

In this example we analyze data collected from inertial sensors placed on the body of 8 individuals. The dataset is available on line from the UCI Machine Learning Repository. The

original objective of the experiment Altun, Barshan, and Tuncel (2010) conducted was to assess the performance of supervised learning methods to identify human activities. We will focus on one of these activities (sitting) with the goal of identifying the different subjects. As recent mobile devices contain accelerometers, biometric subject identification based on inertial sensors has a great potential to serve as an unobtrusive authentication procedure for mobile phone users (Derawi, Nickel, Bours, and Busch 2010).

The observations come from an accelerometer, a gyroscope and a magnetometer, with a 25-Hz sampling frequency. Measurements are taken for 5 minutes, and these are divided into 5-second segments, resulting in 60 observations per subject ($n = 480$). Each 5-second time window produces 125 measurements, which are then converted into 96 featuresmaximum, minimum, mean, skewness, kurtosis, the largest 30 peaks of the discrete Fourier transform of the signal, the 30 associated frequencies, and their auto-covariance function with lags 0 to 30. Each individual wears 5 sets of sensors, each sensor contains three trackers (accelerometer, gyroscope and magnetometer), and each tracker records data on 3 dimensions. This results in 45 ($5 \times 3 \times 3$) sets of signals per subject, for a total of 4320 features (45×96). These features coincide with those in Altun *et al.* (2010), except that they only considered the top 5 Fourier frequencies, and auto-correlations of lags up to 10.

The following code loads the corresponding data matrix `sitting`.

```
R> library("RSKC")
R> sitting0 <- read.csv("sitting.csv")
R> true <- sitting0[,1]
R> sitting <- as.matrix(sitting0[,-1])
R> featActivity <- read.csv("featActivity.csv")
```

Each row of the matrix `featActivity` identifies the source of each feature (location of the sensor on the subject's body, the type of tracker, direction, and the name of the summary statistics). Because the units of the available features are different, we normalize them to the interval [0,1] as follows: each column $\boldsymbol{x}_j$ of `sitting` is transformed into $(\boldsymbol{x}_j - \min(\boldsymbol{x}_j))/(\max(\boldsymbol{x}_j) - \min(\boldsymbol{x}_j))$

```
R> dat.norm0 <- apply(sitting, 2, function(x) (x - min(x))/(max(x) - min(x)))
```

We now remove features that remain constant across our observations, which results in a total of $p = 4301$ features.

```
R> complete.columns <- complete.cases(t(dat.norm0))
R> dat.norm <- dat.norm0[, complete.columns]
R> feat <- featActivity[complete.columns, ]
```

We use K-means, trimmed K-means and their sparse versions to simultaneously reduce the number of features and identify clusters of observations coming from the same individual in the study. Altun *et al.* (2010) reduce the number of features via the principal component analysis and use the 30 principal components corresponding to the largest 30 eigenvalues, so the ideal sparsity could be 30 nonzero-weighted features. However, for demonstration purpose, we analyze the dataset assuming that there is no information regarding the number of clustering features but it is pre-known that the sample size for each subject is the same.

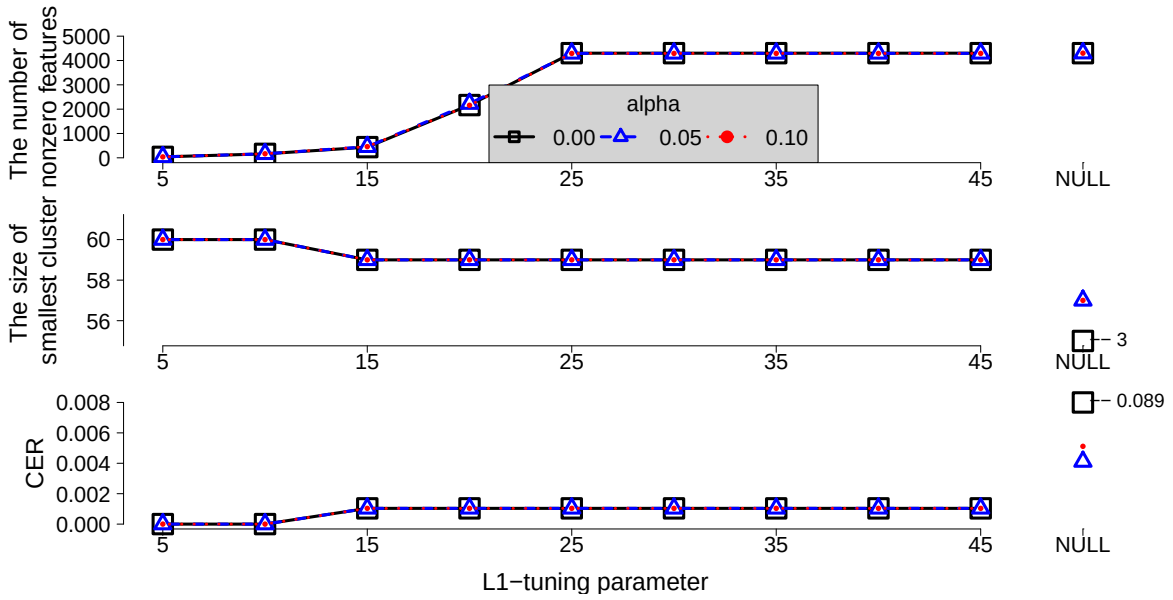


Figure 9: The clustering results for the sitting subject dataset. The rightmost observations (with label `NULL` on the x-axis) corresponds to the non-sparse methods. CER is computed between the estimated partition and the true cluster labels. The size of the smallest cluster and CER from K-means are beyond the ranges of y-axes.

For the sparse methods we set the penalty L_1 to 5, 10, 15, ..., 45, whereas for the robust methods we consider $\alpha = 0.05$ and 0.1.

Figure 9 shows the results of our analysis for different levels of sparsity, in terms of the number of non-zero weights, the size of the smallest cluster and the CER. Since we know that there are 60 observations per subject, we expect a good partition to have all clusters with approximately 60 points in them. This corresponds to a large minimum cluster size. Note that RSK-means with $L_1 = 5$ or 10 and $\alpha = 0.05$ or 0.1, and SK-means with $L_1 = 5$ or 10 have the largest possible minimum cluster size. However, K-means returns one very small cluster of size 3. Interestingly, these 3 cases are indeed trimmed in either weighted or non-weighted distances by both robust clustering methods with all considered α and L_1. Finally, note from the last panel that all the partitions with the "right" smallest cluster size (60) are perfect solutions (CER = 0).

The ability of a clustering method to filter noise features seems to be an advantage in this example, where sparse methods outperform non-sparse ones. To investigate which features are selected by the sparse variants we analyze the feature weights. SK-means with $L_1 = 5$ and RSK-means with $\alpha = 0.1$ and $L_1 = 5$ only assign non-zero weights to 39 and 38 features, respectively. Furthermore, the weight allocations of both methods are very similar, and hence here we only look at the weights returned by RSK-means. To obtain the results of RSK-means with $\alpha = 0.1$ and $L_1 = 5$, run the codes

```
R> set.seed(1)
R> reRSK <- RSKC(dat.norm, L1 = 5, nstart = 300, ncl = 8, alpha = 0.1)
```

We first compute the proportion of weights associated to the measurements taken by each of

the 5 sensors on the individual

```
R> myRound <- function(x) {
+    name.x <- names(x)
+    or <- order(x, decreasing = TRUE)
+    out <- paste(sprintf("%1.1f", x), "%", sep = "")
+    out[x == 0] <- "ZERO"
+    names(out) <- name.x
+    return(out[or])
+  }
R> w <- reRSK$weight
R> sw <- sum(w)/100
R> myRound(tapply(w, feat$body, function(x) sum(x)/sw))

right.arm  left.arm right.leg  left.leg     torso
  "30.8%"   "27.8%"   "23.9%"    "9.6%"    "7.8%"
```

The largest weights are assigned to the features based on the signals from the right arm (30.8%) while the features based on torso signals received the smallest weights (7.8%). We now look at the proportion of weights allocated to features originated on each type of sensor

```
R> myRound(tapply(w, feat$tracker, function(x) sum(x)/sw))

magnetometer accelrometer    gyroscope
     "92.3%"       "7.7%"       "ZERO"
```

`ZERO` means exact zero. As one might expect, features obtained from the gyroscope do not seem to be useful for clustering sitting individuals, while those taken from the magnetometer receive the largest proportion of weights. The statistics derived from the signals that received highest 5 weights can be found as follows

```
R> myRound(tapply(w, feat$statistic, function(x) sum(x)/sw))[1:5]

  peak1    mean     max     min    ACF0
"30.4%" "29.2%" "21.3%" "19.1%"  "ZERO"
```

The largest weight (30.4%) is assigned to features corresponding to the first Fourier peak, followed by the mean, maximum and minimum. The rest of statistics do receive zero weights.

To explore the precision of the cluster partitions obtained when outliers are removed from the data, we construct a curve similar to a recall rate. For each clustering method (K-means, trimmed K-means, SK-means with $L_1 = 5$ and RSK-means with $L_1 = 5$ and $\alpha = 0.1$) we sort the weighted distances of each point to their cluster centers and then compute the CER obtained by removing β% of cases with the largest distances, which may be considered potential outliers. The curves are displayed in Figure 10. Note that the curves for the sparse methods (RKS-means and SK-means) are constant at CER = 0, while those of the non-sparse methods have an increasing trend. This indicates that the cases misclassified by K-means and

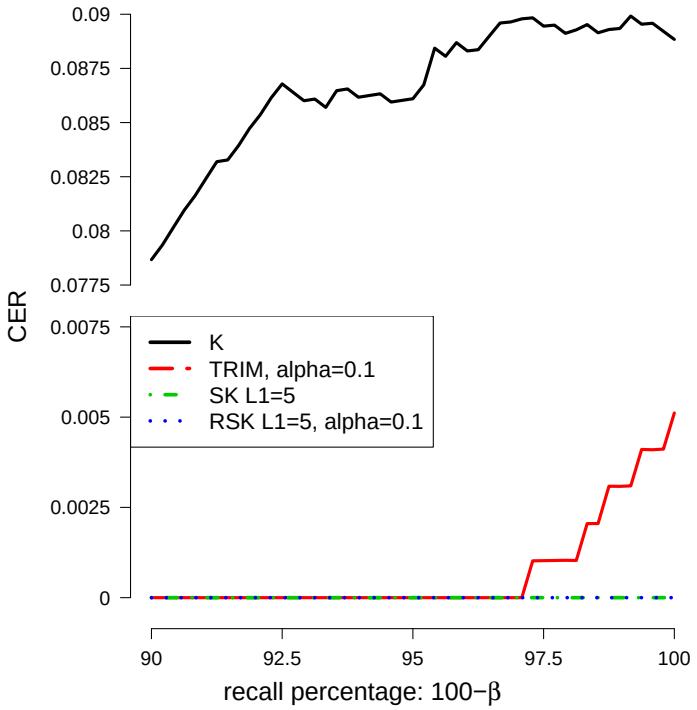


Figure 10: Recall-performance curves for the different cluster partitions on the `sitting` dataset. Each curve corresponds to each clustering method and displays the resulting CER when one removes $\beta\%$ of cases with largest distances to their cluster centers.

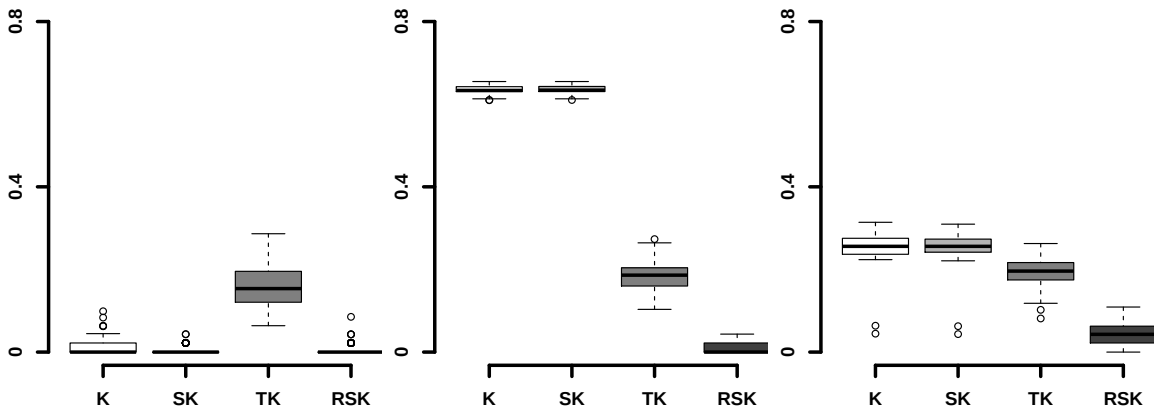
trimmed K-means are indeed possible outliers (those with largest weighted distances to their cluster centers).

In summary, we see that in this example sparse methods outperform their non-sparse counterparts. Non-sparse methods tend to misclassify the cases with large distances to their cluster centers. Both sparse methods achieve a notable dimension reduction (RSK-means uses 38 features, whilie SK-means uses 39) while still correctly identifying all clusters. The results of this analysis suggest that signals from gyroscopes may not be useful in identifying sitting individuals, and also that few and relatively simple summary statistics are needed for this purpose.

4. Simulation Results

In this section we report the results of a simulation study to investigate the properties of the proposed RSK-means algorithm and compare it with K-means, trimmed K-means, and SK-means. Our simulated datasets contain $n = 60$ observations generated from multivariate normal distributions with covariance matrix equal to the identity. We form 3 clusters of equal size by setting the mean vector to be $\boldsymbol{\mu} = \mu \times \mathbf{b}$, where $\mathbf{b} \in \mathbb{R}^{500}$ has its first 50 entries equal to 1 followed by 450 zeroes, and $\mu = -1$, 0, and 1, respectively. Note that the clusters are determined by the first 50 features only (which we call *clustering features*), the remaining 450 being *noise features*.

We will assess the performance of the different cluster procedures regarding two outcomes:



(a) Clean data (b) Model 1 (c) Model 2

Figure 11: Boxplots of CERs calculated between the true partition and the partitions from four algorithms. They correspond to 1000 simulated datasets of size $n = 60$ and $p = 500$ features (with 50 true clustering features). "K", "SK", "TK" and "RSK" denote K-means, sparse K-means, trimmed K-means and robust sparse K-means, respectively.

the identification of the true clusters and the identification of the true clustering features. To measure the degree of cluster agreement we use the CER. To investigate the correct identification of clustering features we adapt the average precision measure, which we compute as follows. First, sort the features in decreasing order according the their weights and count the number of true clustering features appearing among the 50 top-ranked features.

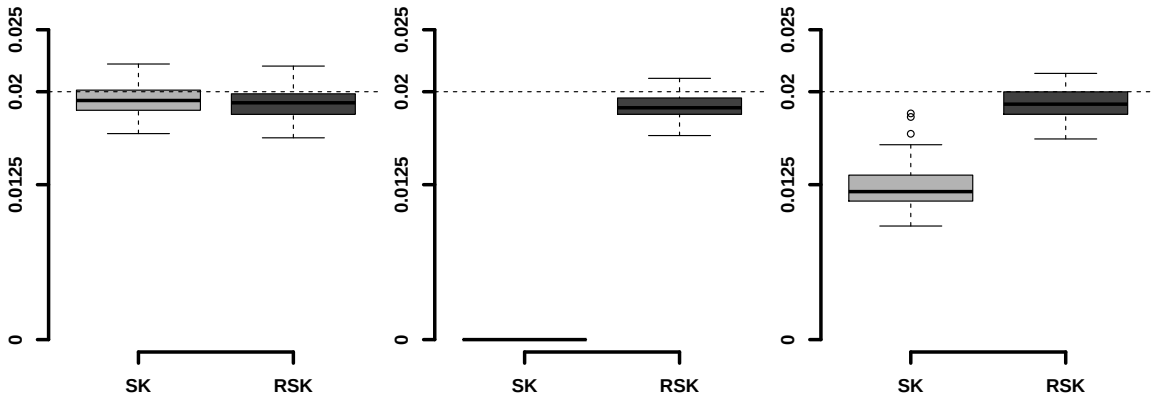
We consider the following two contamination configurations

- Model 1: For each dataset we randomly select 45 noise features, and 10% of the observations are contaminated by replacing these variables with a random number uniformly distributed over the set $[-13, -7] \cup [7, 13]$.
- Model 2: For each dataset we randomly select 5 clustering features, and 10% of the observations are contaminated by replacing these variables with a random number uniformly distributed over the set $[-13, -7] \cup [7, 13]$.

These configurations can be thought of as producing "scattered" outliers located outside the range of the data. Additional experiments where outliers were introduced by replacing a single feature with a large value are discussed in Kondo, Salibian-Barrera, and Zamar (2012). The general conclusions of all our studies agree with those reported here.

The L_1 bound for each algorithm was selected in such a way that approximately 50 features would receive positive weights when used on clean datasets. More specifically, we generated 50 datasets without outliers and, for each algorithm, considered the following 11 L_1 bounds: 5.7, 5.8, ..., 6.7. The one resulting in the number of non-zero weights closest to 50 was recorded. In our simulation study we used the corresponding average of selected L_1 bounds for each algorithm. For SK-means and RSK-means this procedure yielded an L_1 bound of 6.264 and 6.234, respectively. The proportion α of trimmed observations in the trimmed K-means and RSK-means was set equal to 0.1, the true proportion of outliers.

We generated 1000 datasets from each model. Figure 11 shows the boxplots of CERs between



(a) Clean data (b) Model 1 (c) Model 2

Figure 12: The boxplots of median proportions of weights on the 50 clustering features over 1000 simulations. The dotted line represents the ideal amount of weights on each clustering feature. SK = SK-means RSK = RSK-means.

		Non-zero weights	Average precision
No outliers	RSK-means	50.0 (1.80)	48.7 (0.49)
	SK-means	49.4 (0.95)	48.9 (0.24)
Model 1	RSK-means	50.6 (2.14)	48.5 (1.52)
	SK-means	79.3 (17.17)	0.00 (0.07)
Model 2	RSK-means	49.8 (5.82)	48.8 (0.43)
	SK-means	299.3 (93.99)	47.0 (1.15)

Table 1: The first column contains the average number of non-zero weights over the 1000 datasets (SD in parentheses). The second column shows the average number of true clustering features among the 50 features with largest weights (SD in parentheses).

the true partition and the partitions returned by the algorithms. When the data do not contain any outliers we see that the performance of K-means, SK-means, and RSK-means are comparable. The results of Models 1 and 2 show that the outliers affect the performance of K-means and SK-means, and also that the presence of 450 noise features upsets the performance of the trimmed K-means algorithm. However, RSK-means retains small values of CER for both types of contamination.

To compare the performance of the different algorithms with regards to the features selected by them we consider the median weight assigned to the true clustering features. The results are shown in Figure 12. We can see that when there are no outliers in the data both the SK-means and RSK-means algorithms assign very similar weights to the correct clustering features. The presence of outliers, however, results in the SK-means algorithm to assign much smaller weights to the clustering features.

Table 1 contains the average number of non-zero weights returned by each algorithm, and average number of true clustering features among the 50 features receiving highest weights. When the data are clean, both SK-means and RSK-means return approximately 50 clustering features, and they are among the ones with highest weights. The presence of outliers (with either contamination Model 1 or 2) results in SK-means selecting noise features (Model 1),

as the average precision is zero, or selecting about 300 features (Model 1), while RSK-means remains unaffected.

5. Conclusion

We propose a robust algorithm to simultaneously identify clusters and features using K-means and illustrate our method and R package **RSKC** with the analysis of four real datasets. The main idea behind our proposal is to adapt the SK-means algorithm of Witten and Tibshirani (2010) by trimming a given fraction of observations that are farthest away from their cluster centers, using the approach of the trimmed K-means algorithm of Cuesta-Albertos *et al.* (1997). Sparcity is obtained by assigning weights to the features and bounding their L_1 norm. Only those features for which the optimal weights are positive are used to determine the clusters. Because possible outliers may contain atypical entries in features that are being downweighted, our algorithm also considers the distances of each point to their cluster centers using all features. Our simulations and examples show that RSK-means works as well as SK-means when there are no oultiers in the data, and much better than SK-means in the presence of outliers. Handdigit data example (the frist real data example) and DBWorld dataset (the third real data example) show that when noise features and outliers are present, RSK-means can recover the true partition outperforming all the considered competing algorithms. Dataset of digits from Dutch utility map (the second real data example) shows that when sparsity is not clearly present, RSK-means still does a good job and can deal with a relatively large fraction of outliers. Dataset containing signals from the body-worn miniture inertial sensors (the fourth real data example) shows that when outliers are not clearly present, RSK-means outperforms K-means and trimmed K-means, and performs as well as SK-means by identifying sensible clustering features. Finally, the RSK-means algorithm is implemented in the R package **RSKC** available from the Comprehensive R Archive Network at `https://CRAN.R-project.org/package=RSKC`, including the two digit datasets and DBWorld dataset.

Acknowledgments

We would like to thank Michele Filannino for providing supports to analyze the DBWorld dataset.

References

Altun K, Barshan B, Tuncel O (2010). "Comparative Study on Classifying Human Activities with Miniature Inertial and Magnetic Sensors." *Pattern Recognition*, **43**, 3605–3620. doi:10.1016/j.patcog.2010.04.019.

Bache K, Lichman M (2013). "UCI Machine Learning Repository." URL `http://archive.ics.uci.edu/ml/`.

Chipman H, Tibshirani R (2005). "Hybrid Hierarchical Clustering with Applications to Microarray Data." *Biostatistics*, **7**, 286–301. doi:10.1093/biostatistics/kxj007.

Cuesta-Albertos JA, Gordaliza A, Matran C (1997). "Trimmed K-Means: An Attempt to Robustify Quantizers." *The Annals of Statistics*, **25**, 553–576. doi:10.1214/aos/1031833664.

Derawi M, Nickel C, Bours P, Busch C (2010). "Unobtrusive User-Authentication on Mobile Phones Using Biometric Gait." In *Proceeding of the 6th IEEE International Conference on Intelligent Information Hiding and Multimedia Signal Processing*, pp. 306–311. doi:10.1109/iihmsp.2010.83.

Dudoit S, Fridlyand J (2002). "A Prediction-Based Resampling Method for Estimating the Number of Clusters in a Dataset." *Genome Biology*, **3**, 1–21. doi:10.1186/gb-2002-3-7-research0036.

Filannino M (2011). "DBWorld E-Mail Classification Using a Very Small Corpus." *Technical report*, University of Manchester. Project of Machine Learning course.

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

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

Kondo Y (2016). ***RSKC**: Robust Sparse K-Means*. R package version 2.4.2, URL https://CRAN.R-project.org/package=RSKC.

Kondo Y, Salibian-Barrera M, Zamar RH (2012). "A Robust and Sparse K-Means Clustering Algorithm." arXiv:1201.6082 [stat.ML], URL http://arxiv.org/abs/1201.6082.

Lloyd SP (1982). "Least Squares Quantization in PCM." *IEEE Transactions on Information Theory*, **28**, 129–136. doi:10.1109/tit.1982.1056489.

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

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.

Rand WM (1971). "Objective Criteria for the Evaluation of Clustering Methods." *Journal of the American Statistical Association*, **66**, 846–850. doi:10.2307/2284239.

R Core Team (2016). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. URL https://www.R-project.org/.

Steinhaus H (1956). "Sur la Division des Corps Matériels en Parties." *Bulletin de l'Académie Polonaise des Sciences*, **4**, 801–804.

Sugar CA, James GM (2003). "Finding the Number of Clusters in a Dataset: An Information-Theoretic Approach." *Journal of the American Statistical Association*, **98**, 750–763. doi:10.1198/016214503000000666.

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

Witten DM, Tibshirani R (2010). "A Framework for Feature Selection in Clustering." *Journal of the American Statistical Association*, **105**, 713–726. doi:10.1198/jasa.2010.tm09415.

Witten DM, Tibshirani R (2013). ***sparcl**: Perform Sparse Hierarchical Clustering and Sparse K-Means Clustering*. R package version 1.0.3, URL https://CRAN.R-project.org/package=sparcl.

Affiliation:

Yumi Kondo
Department of Statistics, University of British Columbia
3182 Earth Sciences Building
2207 Main Mall
Vancouver, BC, Canada V6T 1Z4
E-mail: y.kondo@stat.ubc.ca
URL: http://stat.ubc.ca/~y.kondo/

Journal of Statistical Software http://www.jstatsoft.org/
published by the Foundation for Open Access Statistics http://www.foastat.org/
August 2016, Volume 72, Issue 5 *Submitted:* 2013-04-04
doi:10.18637/jss.v072.i05 *Accepted:* 2015-06-25