Journal of Statistical Software

May 2012, Volume 47, Issue 14. *http://www.jstatsoft.org/*

bayesclust: An R Package for Testing and Searching for Significant Clusters

Vikneswaran Gopal
IBM Research
Collaboratory Singapore

Claudio Fuentes
Oregon State University

George Casella
University of Florida

Abstract

The detection and determination of clusters has been of special interest among researchers from different fields for a long time. In particular, assessing whether the clusters are significant is a question that has been asked by a number of experimenters. In Fuentes and Casella (2009), the authors put forth a new methodology for analyzing clusters. It tests the hypothesis $H_0 : \kappa = 1$ versus $H_1 : \kappa = k$ in a Bayesian setting, where κ denotes the number of clusters in a population. The **bayesclust** package implements this approach in R. Here we give an overview of the algorithm and a detailed description of the functions available in the package. The routines in **bayesclust** allow the user to test for the existence of clusters, and then pick out optimal partitionings of the data. We demonstrate the testing procedure with simulated datasets.

Keywords: clustering, hierarchical, Bayes, R.

1. Introduction

1.1. About this document

Clustering of data is required in many different disciplines of science, including machine learning, image analysis and genetics. The great demand for this methodology has led to several varied approaches to cluster analysis. However, the majority of these methods are directed towards dividing the data into a pre-specified number of groups; rarely do they accommodate explicitly testing for the existence of clusters.

In Fuentes and Casella (2009), the authors put forward a novel approach to analyzing clusters, which permits us to test the hypothesis

$$H_0 : \kappa = 1 \qquad \text{vs.} \qquad H_1 : \kappa = k$$

where κ is a parameter denoting the true number of clusters in the population from which the data arose. If the null hypothesis is indeed rejected, we can then proceed to partition the data.

This article is an introduction to the R (R Development Core Team 2012) package **bayesclust** – an implementation of the methodology proposed in Fuentes and Casella (2009). The package comprises a suite of functions for testing and searching for significant clusters in multivariate data. It is available from the Comprehensive R Archive Network at `http://CRAN.R-project.org/package=bayesclust`. The aim here is to provide a detailed user guide, depicting the entire testing protocol from start to finish. We will use simple examples and scripts that demonstrate how the functionality of the package can be easily extended.

1.2. The clustering problem

The fundamental aim in clustering is to distribute a set of n distinguishable objects into groups such that the objects within each group are similar to one another, while the groups themselves are different. Implicit in this objective, though, is the assessment of whether or not it is meaningful to partition the observations into different groups, and if so, how many. Partitioning of the data should only be carried out if the assessment deems it is reasonable to do so.

The relatively few methods for testing for significant clusters include Hartigan's Rule in Hartigan (1975) and more recently, work by Tibshirani, Walther, and Hastie (2001) and Sugar and James (2003). These methods are distance-based in their assessment of whether clusters are far enough apart.

We propose embedding the model in a Bayesian framework, and including the true number of clusters as a parameter κ. Then applying a Bayesian model selection methodology, we can derive an explicit hypothesis test for the existence of clusters, or equivalently, that $\kappa \neq 1$. As our procedure is not distance-based, we are able to avoid the use of a metric to determine the clusters. Moreover, evidence for clusters will be determined according to the probability structure used to model the data, and not the "proximity" of the observations.

1.3. Outline of this article

In Section 2, we outline the theoretical framework and distributional assumptions behind our approach. This portion is essentially a distilled version of Fuentes and Casella (2009). The reader is referred to that paper for full details and proofs. As the methods in Sections 2.4, 2.5 and 2.6 are all computationally intensive, we present typical run-times for these routines in Appendix A. Section 3 is concerned with demonstrating the main features of the package. It walks the reader through the entire process of testing a dataset for clusters, running diagnostics, and then searching for the optimal partitioning of the data into clusters. In Section 4, we extend the methodology to testing multiple hypotheses on the same dataset, and highlight some salient features of the frequentist calibration process introduced in Section 2.5. Finally in Section 5, we compare our procedure to the one in **mclust** (Fraley and Raftery 2006).

2. Bayes clustering methodology

2.1. Notations and terminologies

We will denote the observed data by $\boldsymbol{Y} = (Y_1, \ldots, Y_n)$ where $Y_i \in \mathbb{R}^p$ for $i = 1, \ldots, n$. Let κ denote the (unknown) true number of clusters from which the data were drawn. Hence κ takes integer values greater than 0, with $\kappa = 1$ corresponding to there being exactly one cluster. Our primary aim is to test if $\kappa > 1$, which would signify the existence of at least 2 clusters.

With $\kappa = k$ clusters in the dataset, let us denote the number of observations in the j-th cluster by n_j, for $j = 1, \ldots, k$. It is evident that $\sum_{j=1}^{k} n_j = n$, and in accordance with our set-up in the preceding paragraph, that $n_j > 0$ for $i = 1, \ldots, k$.

In addition, we parametrize the *exact* partitioning of $\boldsymbol{Y}$ into $\kappa = k$ clusters by ω_k. In this paper we shall use the terms partitioning, clustering and clusters interchangeably to refer to the groupings that ω_k gives rise to. In order to visualize ω_k, we can view it as a vector of length n, with each individual element drawn from the set $\{1, 2, \ldots, k\}$ such that there are n_1 occurrences of 1, n_2 occurrences of 2, and so on. In fact this is exactly how ω_k is represented in the R code. Given that there are k clusters and n observations, the total number of possible partitionings is equal to $\mathcal{S}_{n,k}$, the Stirling number of the second kind, as shown in Gould (1960). Even when n is quite small, say 50 or so, $\mathcal{S}_{n,k}$ will still be unmanageably large. The secondary goal of our methodology is to pick out the optimal ω_k as evidenced by the data, in light of this hurdle.

For a given partition $\omega_k \in \mathcal{S}_{n,k}$, we will denote by $Y_1^{(j)}, \ldots, Y_{n_j}^{(j)}$ the n_j data points that are allocated to cluster j. Finally, to describe the individual elements of the vector $Y_l^{(j)}$, we shall use the notation $Y_l^{(j)} = (y_{l1}^{(j)}, \ldots, y_{lp}^{(j)})'$, where $l = 1, \ldots, n_j$ and $j = 1, \ldots, k$.

2.2. Bayesian hypothesis testing

This section provides a brief overview of carrying out hypothesis testing in a Bayesian setting, focusing on the special case of model selection. A comprehensive overview of this topic can be found in Pericchi (2005).

Our aim is to test the following hypothesis:

$$H_0 : \kappa = 1 \qquad \text{vs.} \qquad H_1 : \kappa = k \tag{1}$$

Thus each hypothesis is identified with one of the two models we are trying to distinguish between. In general, if $\boldsymbol{Y}$ is conditionally distributed as $f(\cdot|\boldsymbol{\theta})$ and the prior on $\boldsymbol{\theta}$ is denoted by π, then the Bayes factor associated with the above hypothesis test is

$$BF_{10} = \frac{m(\boldsymbol{Y}|\kappa = k)}{m(\boldsymbol{Y}|\kappa = 1)} \tag{2}$$

where $m(\boldsymbol{Y}|\kappa = k) = \int f(\boldsymbol{Y}|\boldsymbol{\theta})\pi(\boldsymbol{\theta}|\kappa = k)\mathrm{d}\boldsymbol{\theta}$. Intuitively, the Bayes factor gives an indication of the evidence in favor of or against H_1 over H_0. In addition to the data, BF_{10} depends on the priors associated with each hypothesis, or in our case, with the two models under consideration. If we denote the distribution of the data given that there are κ clusters by

$m(\boldsymbol{Y}|\kappa)$, we can express the posterior probability of the null hypothesis as

$$\begin{aligned} P(H_0|\boldsymbol{Y}) = P(\kappa=1|\boldsymbol{Y}) &= \frac{m(\boldsymbol{Y}|\kappa=1)P(\kappa=1)}{m(\boldsymbol{Y}|\kappa=1)P(\kappa=1)+m(\boldsymbol{Y}|\kappa=k)P(\kappa=k)} \\ &= \frac{P(\kappa=1)}{P(\kappa=1)+P(\kappa=k)BF_{10}} \end{aligned}$$

With the assumption that either model is equally likely before observing the data, the posterior probability of the null hypothesis reduces to

$$P(H_0|\boldsymbol{Y}) = \frac{1}{1+BF_{10}} \tag{3}$$

This will be our measure of evidence for or against H_0. Small values would correspond to evidence against $\kappa = 1$.

Finally, we rewrite BF_{10} in terms of ω_k:

$$BF_{10} = \frac{1}{m(\boldsymbol{Y}|\kappa=1)} \sum_{\omega_k \in \mathcal{S}_{n,k}} m(\boldsymbol{Y}|\omega_k)\pi(\omega_k) \tag{4}$$

where $\pi(\omega_k)$ denotes the prior probability of partition ω_k.

2.3. Distributional assumptions

Suppose that $\kappa = k$. Then for any partition $\omega_k \in \mathcal{S}_{n,k}$, we assume that all the observations in cluster j follow a multivariate normal distribution.

$$Y_l^{(j)} \sim N(\boldsymbol{\mu}_j, \Sigma_j)$$

for $l = 1, \ldots, n_j$ and $j = 1, \ldots, k$. For the prior on $\boldsymbol{\mu}_j$, we assume that

$$\boldsymbol{\mu}_j|\Sigma_j \sim N(\boldsymbol{\mu}_0^{(j)}, \tau^2\Sigma_j) \quad \text{for } j = 1, \ldots, k \tag{5}$$

With regards to Σ_j, we assume that $\Sigma_j = \text{diag}(\sigma_{1j}^2, \ldots, \sigma_{pj}^2)$, with

$$\sigma_{rj}^2 \sim IG(a,b) = \frac{1}{\Gamma(a)b^a}\frac{1}{(\sigma_{rj}^2)^{a+1}}e^{-1/b\sigma_{rj}^2} \tag{6}$$

for $r = 1, \ldots, p$ and $j = 1, \ldots, k$. This assumption on the covariance matrices, which implies independence within the clusters, is admittedly not the most general. However, as we shall see in Section 2.5, this assumption allows us to obtain the distribution of $P(H_0|\boldsymbol{Y})$ in the multivariate case, as an immediate extension of the univariate case.

If we now set $\boldsymbol{\mu}_0^{(j)}$ to be equal to the sample means $\bar{y}^{(j)}$ we can arrive at the following expression for the ratio term in Equation 4.

$$\begin{aligned} \frac{m(\boldsymbol{Y}|\omega_k)}{m(\boldsymbol{Y}|\kappa=1)} = \left(\frac{2}{b}\right)^{pa(k-1)} \frac{(n\tau^2+1)^{p/2}}{\Gamma(a)^{p(k-1)}\Gamma(\frac{n}{2}+a)^p} \prod_{j=1}^{k} \frac{\Gamma(\frac{n_j}{2}+a)^p}{(n_j\tau^2+1)^{p/2}} \\ \times \prod_{r=1}^{p}\left[\frac{(ns_r^2+\frac{2}{b})^{n/2+a}}{\prod_{j=1}^{k}(n_j s_{rj}^2+\frac{2}{b})^{n_j/2+a}}\right] \end{aligned} \tag{7}$$

where $s_r^2 = (1/n)\sum_{i=1}^{n}(y_{ir} - \bar{y}_r)^2$, $s_{rj}^2 = (1/n_j)\sum_{i=1}^{n_j}(y_{ir}^{(j)} - \bar{y}_r^{(j)})^2$ and $\bar{y}_r^{(j)} = (1/n_j)\sum_{i=0}^{n_j} y_{ir}^{(j)}$ The derivation of the above expression is not trivial and has been omitted here.

What remains is to decide on a prior distribution on the space of possible partitions. Instead of assuming that all priors in $\mathcal{S}_{n,k}$ are equally likely, we will assume the following probability mass function

$$g(\omega_k) = \frac{k!}{\binom{n-1}{k-1}\binom{n}{n_1 \dots n_k}} \tag{8}$$

As explained in Section 2.3 of Fuentes and Casella (2009), one of the benefits of this density is that it is easy to sample from. More importantly, however, the sampling scheme that was used to draw from this distribution could be easily modified to incorporate a minimum cluster size. This was an important consideration in the application for which this methodology was constructed.

2.4. Estimating the Bayes factor

Our strategy to compute the posterior probability of the null hypothesis using Equation 3 requires us to first calculate the Bayes factor using Equation 4. Since this is a sum over all possible partitionings in $\mathcal{S}_{n,k}$, it is practically impossible to compute. We shall resort to Markov chain Monte Carlo techniques in order to approximate the Bayes factor in the following manner

$$BF_{10} = \sum_{\omega_k \in \mathcal{S}_{n,k}} \left[\frac{m(\boldsymbol{Y}|\omega_k)}{m(\boldsymbol{Y}|\kappa = 1)}\right] \pi(\omega_k) \tag{9}$$

$$\approx \frac{1}{M}\sum_{i=1}^{M} \left[\frac{m(\boldsymbol{Y}|\omega_k^{(i)})}{m(\boldsymbol{Y}|\kappa = 1)}\right] \tag{10}$$

where for $i = 1, \dots, M$, the $\omega_k^{(i)}$ are draws from the g distribution. We shall utilize a Metropolis-Hastings algorithm with stationary distribution g to draw the partitions, in which case the ergodic average above would converge to the desired integral BF_{10}, as M increases without bound.

The purpose of using a Metropolis-Hastings algorithm is to ensure that the sampling will remain in areas of high probability, thus achieving a more accurate calculation while maintaining the correct stationary distribution. The candidate distribution for our Metropolis-Hastings step will be a mixture of 2 component distributions:

- *Independent draw:* At iteration t, draw candidate ω_k' from g.
- *Random walk:* At iteration t, obtain candidate ω_k' by choosing one observation at random from $\omega_k^{(t)}$, and moving it to one of the other $k-1$ clusters with equal probability.

The density of this mixture distribution we denote by

$$h(\omega_k'|\omega_k^{(t)}) = \frac{a}{n(k-1)} + (1-a)g(\omega_k') \tag{11}$$

where $a \in (0,1)$ determines the probability of drawing a partition from the random walk component.

The final Metropolis-Hastings algorithm is as follows: At iteration t

1. With probability a, draw candidate ω_k' from the random walk centered at $\omega_k^{(t)}$, and with probability $1 - a$ draw candidate ω_k' independently from g.

2. Compute the Metropolis-Hastings ratio

$$MH = \frac{g(\omega_k')}{h(\omega_k'|\omega_k^{(t)})} \times \frac{h(\omega_k^{(t)}|\omega_k')}{g(\omega_k^{(t)})}$$

3. With probability $\min(1, MH)$ set $\omega_k^{(t+1)} = \omega_k'$, otherwise set $\omega_k^{(t+1)} = \omega_k^{(t)}$

There are a number of texts that explain why the stationary distribution of the above Markov chain will be g, and that justify convergence of the mean to the desired integral. For instance, the reader can consult Chapter 7 of Robert and Casella (2004).

2.5. Frequentist calibration

Although we are now at the stage where, for a given dataset, we can compute the posterior probability of there being only one cluster in the data, we are still unable to assess exactly how strong the evidence is for or against the null hypothesis. All we know is that lower values for the computed $P(H_0|\boldsymbol{Y})$ correspond to evidence for more than one cluster.

Our solution will be to derive the frequentist null distribution of $P(H_0|\boldsymbol{Y})$, that is, the distribution of $P(H_0|\boldsymbol{Y})$ as a function of the data when the null hypothesis is true. In essence, we will treat the computed value of (3) as a statistic. Knowing its distribution under H_0 will permit us to obtain a p value for it, or to obtain critical values which we can check against for fixed significance levels of the hypothesis test.

First, define $\mathcal{P}_{n,k}$ be the number of ways that we can partition an integer n into k integers $n_1, n_2, \ldots, n_k$ such that $n = n_1 + n_2 + \ldots + n_k$. Also, let ξ be an element in $\mathcal{P}_{n,k}$, and consider first the case $p = 1$. Then Lemma 1 from Fuentes and Casella (2009) gives us a method of obtaining samples from the distribution of $BF_{10}(\boldsymbol{Y})$ when testing the hypothesis (1) and when H_0 is true.

Lemma 1 (Fuentes and Casella 2009)**.** *Let $v_1, \ldots, v_{n-1}$ be iid χ_1^2 random variables and let $\boldsymbol{V} = (v_1, \ldots, v_{n-1})$. Then, if the null hypothesis holds,*

$$BF_{10}(\boldsymbol{Y}) \overset{\mathcal{D}}{=} \sum_{\xi \in \mathcal{P}_{n,k}} \phi(\xi) T(\boldsymbol{V}|\xi)$$

where

$$T(\boldsymbol{V}|\xi) \overset{\mathcal{D}}{=} \frac{1}{(\sigma^2)^{(k-1)a}} \frac{\left(\sum_{i=1}^{n-1} v_i + 2/b\sigma^2\right)^{\frac{n}{2}+a}}{\prod_{j=1}^{k} \left(\sum_{i=n_{j-1}}^{n_j - 1} v_i + 2/b\sigma^2\right)^{\frac{n_j}{2}+a}}$$

and $\phi(\xi)$ is an appropriate normalizing constant for every $\xi \in \mathcal{P}_{n,k}$.

The details of the proof are in Sections 4 and Appendix B.2 of the quoted paper. It can be seen that to obtain one draw from this distribution, a vector of χ_1^2 random variables has to be drawn, and the above sum computed. Due to the number of elements in $\mathcal{P}_{n,k}$, this cannot be computed exactly and will be estimated by Monte Carlo techniques. Although the result above corresponds to the univariate case, it can immediately be extended to the multivariate situation, because of the assumed form of the covariance matrices Σ_j in Section 2.3. This result is also detailed in the original paper.

At this point, we have outlined the theory that will allow an experimenter to conduct the hypothesis test in Equation 1, and decide whether or not to reject the null. For a given dataset, the experimenter would have to first estimate $P(H_0|\boldsymbol{Y})$ by approximating BF_{10} using the theory laid out in Section 2.4. The next step would be to calibrate this posterior probability by applying the theory in this section. This would involve generating the distribution of the computed statistic under H_0, by sampling from it. The experimenter would then obtain a cutoff point from this distribution, which would serve as a critical value for a pre-specified α-level of significance.

2.6. Searching for the optimal cluster

In this section we consider the problem of picking out the optimal partitioning of the data once we have concluded that there are indeed significant clusters. This immediately raises the question of what is meant by optimality. We define the objective function to be $m(\boldsymbol{Y}|\omega_k)$, considered as a function of ω_k. With this definition, the optimal ω_k is the one that maximizes the objective function for a given set of observations. Coupled with the assumption that $\pi(\omega_k) \propto 1$, maximizing this objective function is equivalent to maximizing the posterior probability of ω_k.

However, as the partition space is extremely large, it is not possible to examine every single partition. Instead we shall have to resort to a stochastic search through $\mathcal{S}_{n,k}$. The proposed Metropolis search algorithm will again utilise the mixture distribution in Equation 11 as the proposal distribution.

The Metropolis algorithm to find the optimal cluster is as follows: At iteration t

1. With probability a, draw candidate ω_k' from the random walk centered at $\omega_k^{(t)}$, and with probability $1-a$ draw candidate ω_k' independently from g.

2. Compute the Metropolis-Hastings ratio

$$MH = \frac{m(\boldsymbol{Y}|\omega_k')}{h(\omega_k'|\omega_k^{(t)})} \times \frac{h(\omega_k^{(t)}|\omega_k')}{m(\boldsymbol{Y}|\omega_k^{(t)})}$$

3. With probability $\min(1, MH)$ set $\omega_k^{(t+1)} = \omega_k'$, otherwise set $\omega_k^{(t+1)} = \omega_k^{(t)}$

As the algorithm cycles through points in the partition space, it will store the best partitions found.

2.7. Minimum cluster size

From the scatter plots, it might appear that one observation is extremely different from the rest. However, we might want to avoid placing that observation in a cluster of its own. In fact, in our application, the investigator only wanted to consider clusters to be significant if they were of a certain minimum size. Our methodology allows for this modification, and this feature is available in **bayesclust**. The minimal changes needed are concerned with the prior and the candidate distribution in the Metropolis algorithms.

Suppose that a minimum cluster size of m is imposed. Then the new modified prior will look like this:

$$g_m(\omega_k) = \frac{k!}{\binom{n-mk+k-1}{k-1}\binom{n}{n_1 n_2 \ldots n_k}}, \tag{12}$$

and the new candidate distribution will look like this:

$$h(\omega'_k|\omega_k^{(t)}) = \frac{a}{n_m(k-1)} + (1-a)g(\omega'_k), \tag{13}$$

where n_m is the total number of observations from clusters that have strictly more than m observations.

It should be pointed out that imposing a minimum cluster size does prevent outliers from being isolated, as it forces every single point to belong to a cluster. If this is not palatable, then the user has the option of removing minimum cluster size restriction. For more on this aspect of the methodology, please see the Rejoinder section of Fuentes and Casella (2009).

3. Usage

In this portion of the article we provide an example of how to use **bayesclust** to test and search for clusters in multivariate data, using the methods described in the previous sections. We demonstrate a workflow that an experimenter would go through from start to finish. Figure 1 summarises this workflow, and the manner in which the functions in **bayesclust** are related. The "No" branch in Figure 1 and the use of the `combine()` function are described in more detail in Section 4.1.

Suppose that an experimenter wishes to test the hypothesis given in Equation 1 for a specific k. Then a summary of the workflow is as follows:

1. Run `cluster.test()` on dataset to obtain an estimate of $P(H_0|\boldsymbol{Y})$ for the given hypothesis test. We shall refer to this estimate as the *empirical posterior probability of the null hypothesis.*

2. Run `nulldensity()` to generate the distribution of $P(H_0|\boldsymbol{Y})$ under the null hypothesis, for the particular hypothesis test.

3. Assess the significance of the estimate of $P(H_0|\boldsymbol{Y})$, by obtaining its p value. The relevant function at this stage is `emp2pval()`, which stands for "empirical posterior probability of null hypothesis to p value".

4. If the k-clustering is deemed significant, then proceed to partition the observations into an optimal k-clustering using `cluster.optimal()`.

In the remainder of this section, we devote one subsection to each of the 4 steps outlined above.

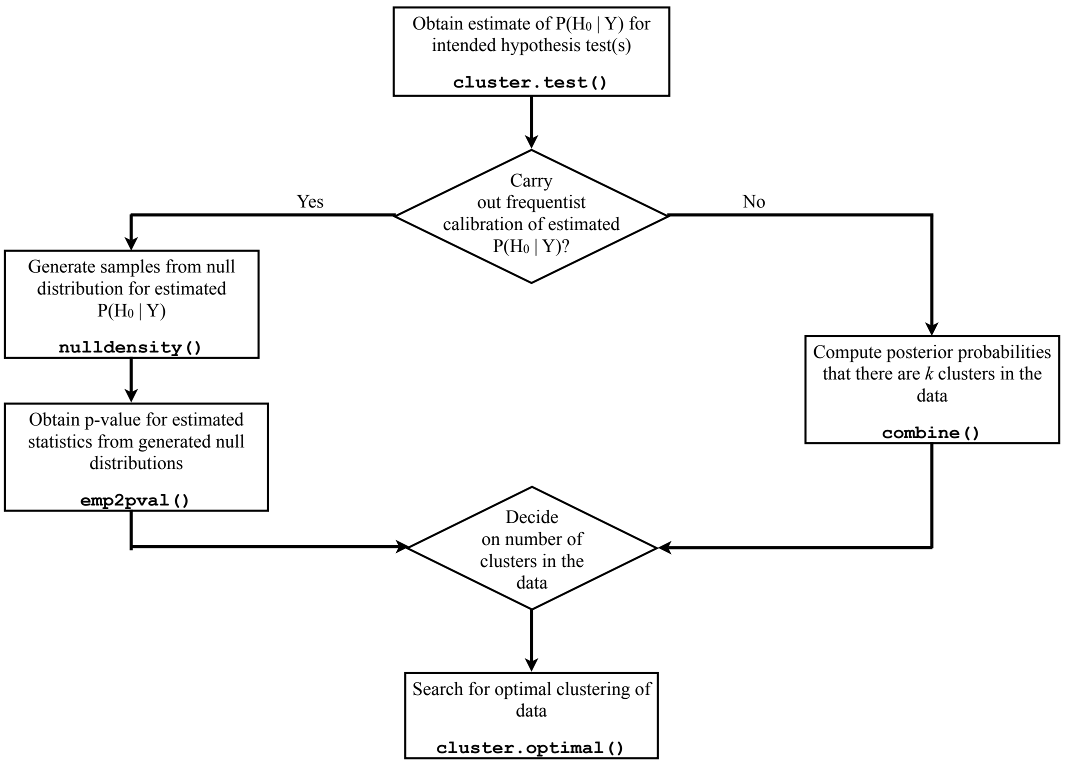
Obtain estimate of P(H0 | Y) for intended hypothesis test(s)
cluster.test()
Carry out frequentist calibration of estimated P(H0 | Y)?
Yes
No
Generate samples from null distribution for estimated P(H0 | Y)
nulldensity()
Obtain p-value for estimated statistics from generated null distributions
emp2pval()
Compute posterior probabilities that there are k clusters in the data
combine()
Decide on number of clusters in the data
Search for optimal clustering of data
cluster.optimal()

Figure 1: Flowchart demonstrating usage of functions in **bayesclust**. The relevant functions in each stage of the analysis are in bold font.

3.1. Carrying out the cluster test

Dataset 1 consists of 75 observations, simulated from bivariate normal distributions with three different means. This dataset is available as part of the **bayesclust** package. Figure 2 contains a scatter plot of this dataset, which will be used to demonstrate the functions in **bayesclust**. As we know the true value of κ to be 3, we expect the test procedure to reject

$$H_0 : \kappa = 1 \qquad \text{vs.} \qquad H_1 : \kappa = 3 \tag{14}$$

The R code below will compute $P(H_0|\boldsymbol{Y})$ for the hypothesis given in Equation 14, just as described in Sections 2.2 and 2.4.

```
R> library("bayesclust")
R> data("egDataset1")
R> clusterTestK3 <- cluster.test(egDataset1, nsim = 500000, p = 2, k = 3,
+    mcs = 0.1, replications = 4)
```

The first argument to `cluster.test()` is the data matrix. Now we put the rest of the function arguments into the context of the previous section: `nsim` sets the value of M in Equation 10, `aR` corresponds to a in the mixture distribution of Equation 11 and `mcs` sets the minimum cluster size discussed in Section 2.7. For this test, we have set it to be 10% of the total sample size. Hence it would take a value of 8 here. (To remove the minimum cluster size restriction

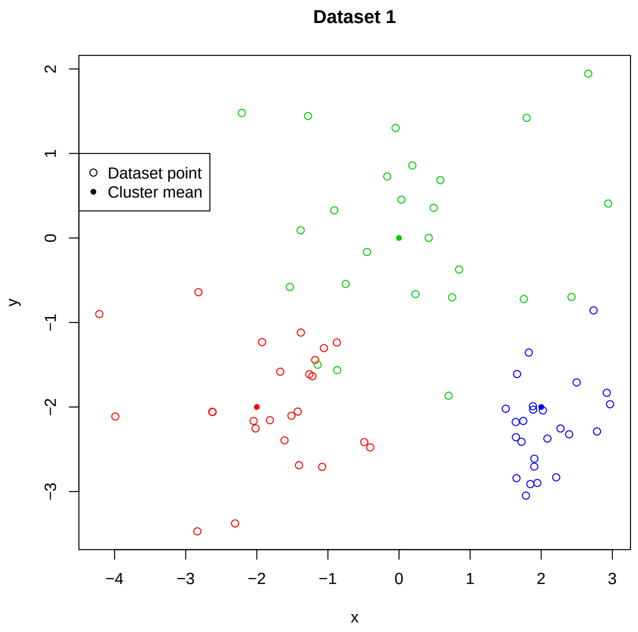

Figure 2: Unshaded circles represent the 75 points from dataset 1. There are 25 observations from each of the 3 different sampling distributions, which are differentiated by color, and whose means are indicated by smaller, shaded circles. For this example dataset, we have $n = 75$, $\kappa = 3$, and $p = 2$.

entirely, the user can specify a value of `mcs` that is less than $1/n$.) The unspecified arguments for `a,b` and `tau2` correspond to the hyperparameters in Equations 5 and 6. Since we are sampling from the space of partitions when running our Markov chain, the usual convergence diagnostics are not are not applicable. Hence we recommend running replications of the chain in order to monitor convergence of the estimate of $P(H_0|\boldsymbol{Y})$. In the above example code, the final argument is an instruction to carry out 4 replicate chains.

When only 1 replication is requested, `cluster.test()` will return an S3 object of class "`cluster.test`". It is a list with 3 components. The first component, `param`, serves as a record of the arguments that were used when `cluster.test()` was run. The second and third components in the list are for plotting the running estimates of $P(H_0|\boldsymbol{Y})$. If more than one replication has been carried out, then `cluster.test()` returns an object of class "`cluster.test.reps`", which is a list of objects of class "`cluster.test`".

The `plot()` method can be called on objects of class "`cluster.test.reps`" or "`cluster.test`". From our experience, it is recommended that when running `cluster.test`, `nsim` be at least 500,000. With so many points, it is not necessary to plot or store every single one. Thus, only every 500th value of the Markov chain is stored in the `cluster.test` object. An example of the convergence plot is shown in Figure 3.

```
R> plot(clusterTestK3)
```

Extracting the (final) estimated posterior probability can be done either manually or using the `summary()` function available in **bayesclust**.

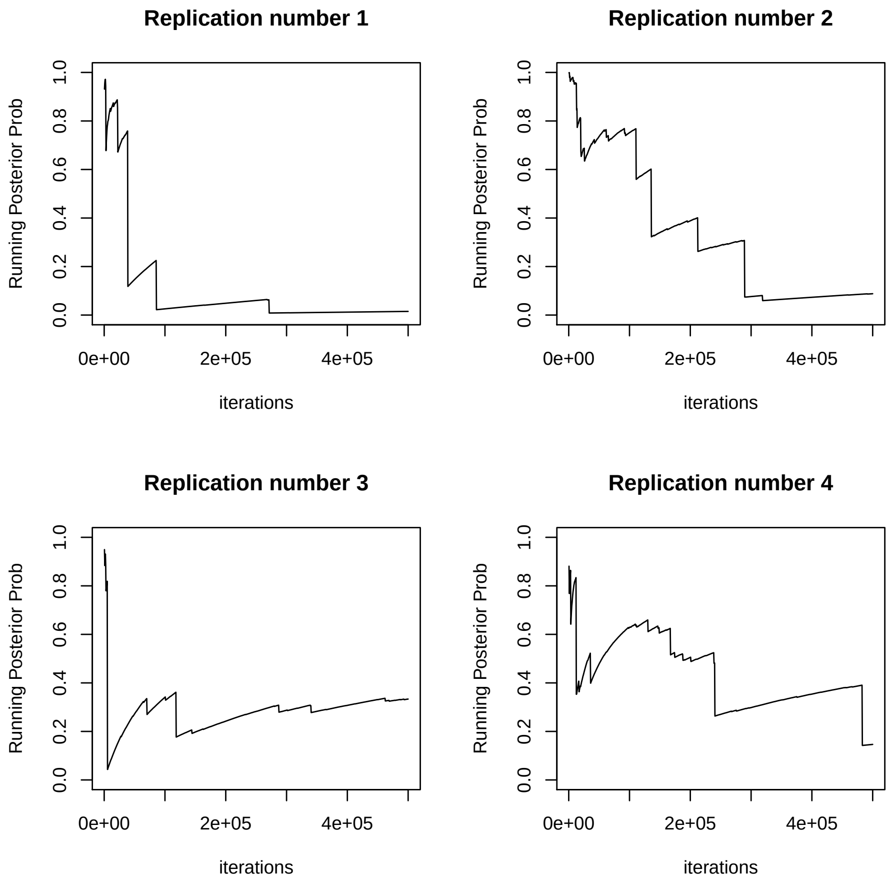

Figure 3: Running estimates of posterior probability for dataset 1.

```
R> summary(clusterTestK3)

Cluster test conducted on data object data1, with 5e+05 iterations.
Num. observations          : 75
Min cluster size           : 8
p                          : 2
H1                         : k = 3
****************************************
Final Empirical Posterior Probabilities:
****************************************
     Post.Probs
rep1     0.0151
rep2     0.0880
rep3     0.3333
rep4     0.1465

Please run emp2pval to obtain the corresponding P-values for
the mean of the above statistics
```

3.2. Distribution of estimates under the null hypothesis

This section covers generation of random variables from the distribution of $P(H_0|\boldsymbol{Y})$, when the null hypothesis is true. It is part of the calibration procedure detailed in Section 2.5. In order to save time, this portion of the testing procedure should be carried out at the same time as Section 3.1, since the output from `cluster.test()` is not needed here. As such, we have kept `nulldensity()` independent of `cluster.test()`, even though our methodology requires both steps.

The key piece of information necessary to start `nulldensity()` running is the particular simple hypothesis being tested. The fact that $BF_{10}(\boldsymbol{Y})$, and consequently $P(H_0|\boldsymbol{Y})$, depends on this can be seen from Equations 4 and 7. Now we continue with our example dataset, running `nulldensity()` for the hypothesis in Equation 14.

```
R> nulldensK3 <- nulldensity(n = 75, nsim = 8000, k = 3, mcs = 0.1, p = 2,
+    prop = 0.25)
R> hist(nulldensK3, main = "Null Density Histogram", xlab = "samples")
```

A histogram of the null distribution can be seen in Figure 4. The object returned by `nulldensity()` has class "`nulldensity`". The `prop` argument refers to the proportion of the partition space $\mathcal{P}_{n,k}$ that is sampled from *at each iteration* of the algorithm, since it is not feasible to compute the entire sum indicated in Lemma 1 at every step. It is recommended that `prop` be at least 0.25. The `nsim` argument indicates how many samples are to be drawn from the distribution. The recommended number of draws here is 8,000 - 10,000.

It has to be stressed that parameters used to generate the null distribution, that is, n, p, κ, τ^2, a, b and the minimum cluster size, should exactly match those used when running `cluster.test()`. Only in this situation will the test valid.

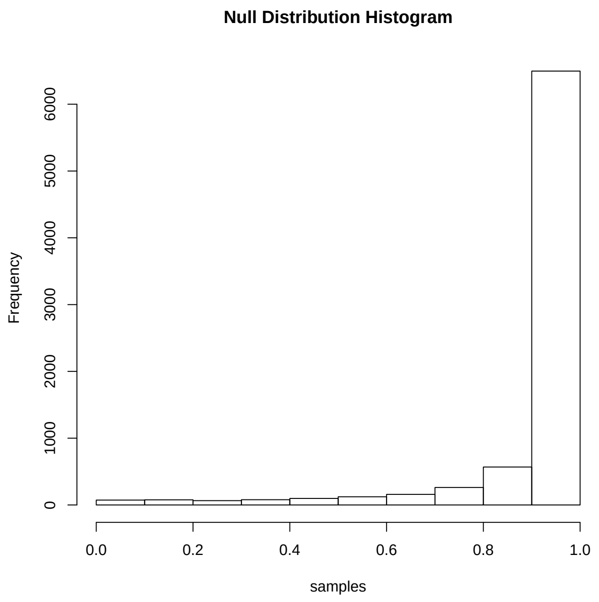

Figure 4: Null distribution of $P(H_0|\boldsymbol{Y})$ when testing $\kappa = 3$. For this example, $n = 75$, $\kappa = 3$, $p = 2$.

3.3. Obtaining frequentist p value

In order to calibrate the test statistic from Section 3.1, the funtion `emp2pval()` with the objects from the preceding two steps has to be called.

```
R> emp2pval(clusterTestK3, nulldensK3)

      mean.emp.prob  pvalue
data1     0.1457166 0.01375
```

3.4. Searching for optimal clusters

The function to search for the optimal partitioning of the data is `cluster.optimal()`. By default, the routine will store the best 4 partitionings encountered. However, the argument `keep` can be used to specify how many to store instead. Here, we consider partitioning dataset 1 into 3 clusters. The R code below will run the search function and then plot the top 4 clusterings found.

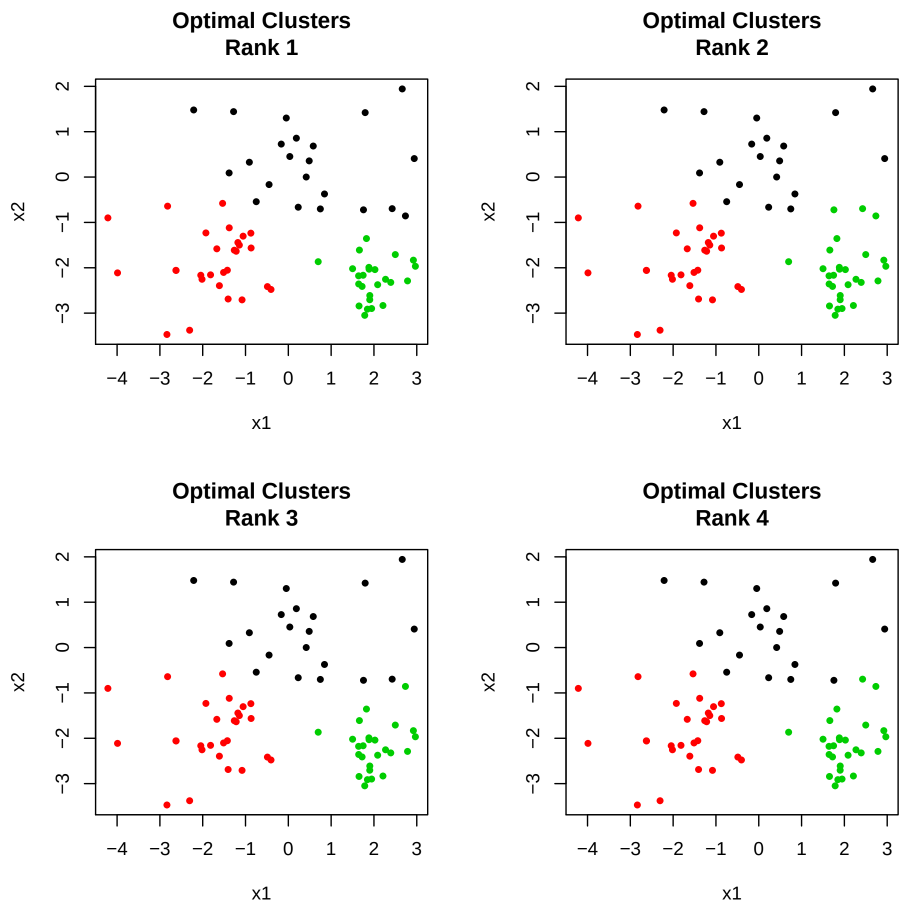

Figure 5: Optimal partitioning of dataset 1 into 3 clusters.

```
R> clusterOptK3 <- cluster.optimal(egDataset1, nsim = 100000, p = 2,
+    k = 3, mcs = 0.1)
R> plot(clusterOptK3)
```

The top 4 clusterings found by `cluster.optimal` can be seen in Figure 5. They are ranked according to their corresponding $\log(m(\boldsymbol{Y}|\omega_k))$ values. The cluster with the largest $\log(m(\boldsymbol{Y}|\omega_k))$ value is ranked the highest. These values can be manually extracted from the output object, which has class "`cluster.optimal`". Examining them allows one to assess the relative quality of the partitionings found.

4. Discussion

4.1. Conducting multiple tests on a single dataset

The situation might arise where an experimenter has to conducts several tests on the same dataset. For example, for dataset 1, we might test the following hypotheses, as well as (14).

$$H_0 : \kappa = 1 \quad \text{vs.} \quad H_1 : \kappa = 2 \tag{15}$$

$$H_0 : \kappa = 1 \quad \text{vs.} \quad H_1 : \kappa = 4 \tag{16}$$

How then should we combine the results? Here we describe two possible options for the experimenter.

The first option is to obtain p values for all 3 tests, and then test for their significance while controlling the false discovery rate (FDR). The 3 tests are not independent since they involve the same dataset, so one possible controlling procedure is the one outlined in Theorem 1.3 of Benjamini and Yekutieli (2001). Note also that, as emphasized in Section 3.2, a different null distribution has to be generated for each test that is conducted on the dataset.

The second option was suggested in Bayarri (2009). If the experimenter knows in advance the maximum number of clusters present in the data, then this alternative procedure is applicable (see the "Rejoinder" section in Fuentes and Casella 2009). An attractive feature of this method is that it side-steps having to generate the null distribution of $P(H_0|\boldsymbol{Y})$ for any of the 3 hypothesis tests. Using this method returns us posterior probabilities that there are k clusters in the data. Here is a brief explanation of how and why it works. Refer to our example dataset, and the 3 hypothesis tests conducted on it for $\kappa = 2, 3$ and 4. If we assume that all 4 hypotheses are equally likely a priori, and we denote the Bayes factor for testing $\kappa = k$ by $BF_{10}(k)$, then the following formula yields posterior probabilities for the 4 different models under consideration.

$$P(\kappa = k|\boldsymbol{Y}) = \frac{BF_{10}(k)}{\sum_{i=1}^{4} BF_{10}(i)} \tag{17}$$

where $BF_{10}(1) = 1$. Since the Bayes factors that are required for the above computation can be derived from the $P(H_0|\boldsymbol{Y})$ estimated in each test, the resulting probability distribution on the set $\{1, 2, 3, 4\}$ can be used to pick the k that has the highest posterior probability. The function `combine()` in **bayesclust** implements this procedure. It takes any number of objects of class "`cluster.test`" or "`cluster.test.reps`" as its arguments. As an example, consider once more the example dataset introduced in the previous section.

```
R> clusterTestK2 <- cluster.test(egDataset1, nsim = 500000, p = 2, k = 2,
+    mcs = 0.1, replications = 4)
R> clusterTestK4 <- cluster.test(egDataset1, nsim = 500000, p = 2, k = 4,
+    mcs = 0.1, replications = 4)
R> combine(clusterTestK2, clusterTestK3, clusterTestK4)

    Postr.Prob
K=1 0.02342574
K=2 0.34990387
K=3 0.48843776
K=4 0.13823263
```

The dataset can then be partitioned into 3 clusters using `cluster.optimal()`, as per Section 3.4.

4.2. Using the default null distributions

Sometimes, the experimenter might not want to wait for the null distribution to be generated. In this situation, **bayesclust** provides pre-computed cut-off points to use.

```
R> data("cutoffs")
R> tail(cutoffs)

     n mcs p k cutoff1pct cutoff5pct
49 100 0.2 2 2  0.1803965  0.6043680
50 100 0.2 2 3  0.4229948  0.8828393
51 100 0.2 2 4  0.7322049  0.9714527
52 100 0.2 3 2  0.3271753  0.8140608
53 100 0.2 3 3  0.8228163  0.9850078
54 100 0.2 3 4  0.9707377  0.9987355
```

The experimenter can then look for the parameters most similar to those under which his own test was run, and repeat the procedure outlined earlier in this section to obtain the cut-off point.

Notice that the cutoff for a test with significance level $\alpha = 0.05$ is 0.999 when $n = 100$, $p = 3$, $\kappa = 4$ and the minimum cluster size is set to 20. This underlines the importance of the calibration step. Without it, we would probably accept H_0 if the computed value of $P(H_0|\boldsymbol{Y})$ turned out to be 0.8. However, based on the null distribution cutoff, we should in fact reject H_0 at $\alpha = 0.05$ significance level.

4.3. Variation of the null distribution

The null distribution of $P(H_0|\boldsymbol{Y})$ varies with the parameters under which the test is conducted: n, p, κ and the minimum cluster size. Even for a given dataset and a fixed minimum cluster size, the distribution of the statistic under the null hypothesis will not be the same when different values of κ are tested. The reason for this is that the expression for the Bayes factor in Equation 2 depends explicitly on the value of κ being tested.

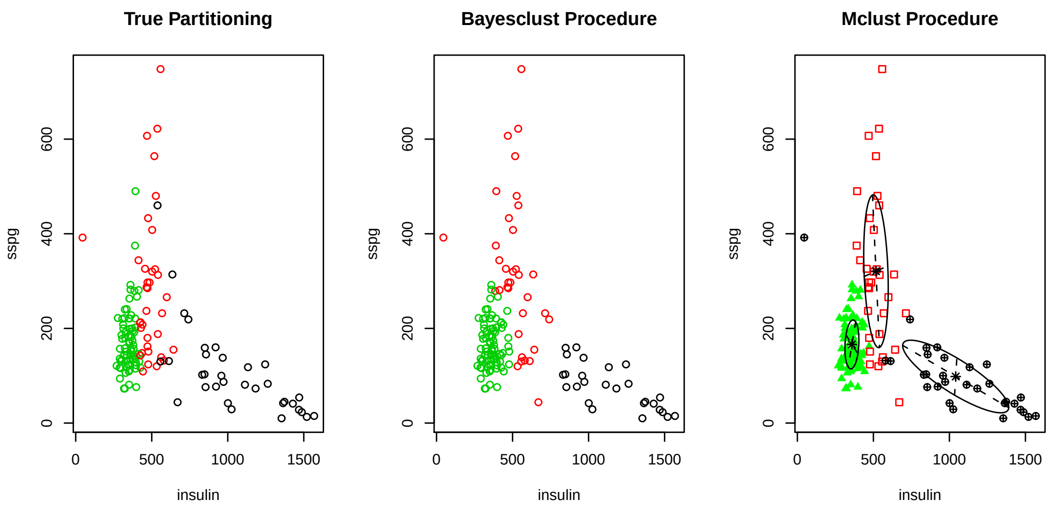

Figure 6: The plot on the left presents the true partitioning of the dataset. The plot in the middle is the optimal 3-clustering found by **bayesclust**. The plot on the right displays the best partitioning of the data found by **mclust**.

In all cases, though, the effect on the null distribution is similar. As n, p, κ and the minimum cluster size increase, the shape of the distribution becomes more skewed to the left. As far as possible, this should be taken into account when the default null cutoffs are used in decision-making. For example, suppose that `cluster.test()` is run with `n = 57`, `mcs = 0.1`, `p = 1`, and `k = 2`. Then the closest set of parameters from the table of default values would be `n = 50`, `mcs = 0.1`, `p = 1`, and `k = 2`. However, the experimenter should be aware that there could be an error in the conclusion, and that the error would be in the conservative direction. Plots that demonstrate some of the variations are given in Appendix B.

5. Comparison with mclust

In this section, we compare the **bayesclust** procedure with classification by **mclust**. For the methods and routines implemented in **mclust**, the reader is referred to Fraley and Raftery (2007). We shall apply both routines to the `diabetes` dataset that can be obtained from the **mclust**, and compare their outputs. This dataset consists of data from 3 clusters. We shall compare the two procedures by assessing the number of misclassifications returned by each.

The first step we have to do is test for the presence of 3 clusters using **bayesclust**.

```
R> library("bayesclust")
R> library("mclust")
R> data("diabetes")
R> X <- as.matrix(diabetes[, 2:4])
R> clusterTestK3 <- cluster.test(X, nsim = 500000, p = 3, k = 3,
+    mcs = 0.1, replications = 4)
R> nulldensK3 <- nulldensity(n = 145, nsim = 8000, k = 3, mcs = 0.1,
+    p = 3, prop = 0.25)
```

```
R> emp2pval(clusterTestK3, nulldensK3)

      mean.emp.prob   pvalue
data1  2.761797e-15 0.000125
```

As we can see, the obtained p value is significant at $\alpha = 0.05$ level, giving strong evidence for the presence of 3 significant clusters. Hence we can proceed to partition the data into 3 clusters.

```
R> clusterOptK3 <- cluster.optimal(X, nsim = 200000, p = 3, k = 3,
+    mcs = 0.1)
```

Plots of the clusterings found by the two algorithms can be found in Figure 6. Counting the wrongly classified steps, we can see that the **bayesclust** procedure makes 23 misclassifications, whereas **mclust** makes only 17. Both procedures agree that a 3-clustering is significant, but the **mclust** procedure seems to work better in this case. However, it is worth pointing out that by construction, **bayesclust** provides the user with a method of explicitly testing the significance of the 3-clustering, whereas **mclust** does not.

For the above comparison, we have used a relatively small dataset. However, we have applied the package to larger datasets from genetics in order to test its limitations. In De Souto, Costa, De Araujo, Ludermir, and Schliep (2008), the authors cluster 35 different datasets. The maximum number of observations they have is $n = 250$, and the maximal dimension is $p = 4500$. In such a situation, our cluster test and search would take approximately 1 minute to carry out 100 iterations of the Markov chain on a 2.2 GHz processor. Generating from the null distribution is typically slower though, and would take about 45 seconds for 2 simulations. Overall, when n is larger than, say 150, the experimenter might prefer not to calibrate the test statistic and to use the alternative, fully Bayesian approach (see Figure 1 and Equation 17). For more timings, please refer to Section A.

6. Conclusions

We have presented a detailed user guide for the R package **bayesclust**, which allows for testing and searching for significant clusters in multivariate data. We have demonstrated that our methodology provides a ready alternative to other available methods, such as **mclust**.

Acknowledgments

This research was partially supported by NSF-DBI grant 0606607 and NIH grant R01GM081704. We would also like to thank the anonymous referees and Assistant Editor, whose comments and suggestions greatly improved this paper and package.

References

Bayarri MJ (2009). "Discussion of 'Testing for the Existence of Clusters'." *SORT: Statistics and Operations Research Transactions*, **33**(2), 149–152.

Benjamini Y, Yekutieli D (2001). "The Control of the False Discovery Rate in Multiple Testing under Dependency." *The Annals of Statistics*, **29**(4), 1165–1188.

De Souto MCP, Costa IG, De Araujo DSA, Ludermir TB, Schliep A (2008). "Clustering Cancer Gene Expression Data: A Comparative Study." *BMC Bioinformatics*, **9**(1), 497.

Fraley C, Raftery AE (2006). ***mclust** Version 3 for R: Normal Mixture Modeling and Model-Based Clustering.* Revised in 2012.

Fraley C, Raftery AE (2007). "Model-Based Methods of Classification: Using the **mclust** Software in Chemometrics." *Journal of Statistical Software*, **18**(6), 1–13.

Fuentes C, Casella G (2009). "Testing for the Existence of Clusters." *SORT: Statistics and Operations Research Transactions*, **33**(2), 115–146.

Gould HW (1960). "Stirling Number Representation Problems." *Proceedings of the American Mathematical Society*, **11**(3), 447–451.

Hartigan JA (1975). *Clustering Algorithms.* John Wiley & Sons.

Pericchi LR (2005). "Model Selection and Hypothesis Testing Based on Objective Probabilities and Bayes Factors." *Bayesian Thinking: Modeling and Computation.*

R Development Core Team (2012). *R: A Language and Environment for Statistical Computing.* R Foundation for Statistical Computing, Vienna, Austria. ISBN 3-900051-07-0, URL `http://www.R-project.org/`.

Robert CP, Casella G (2004). *Monte Carlo Statistical Methods.* Springer-Verlag.

Sugar C, James G (2003). "Finding the Number of Clusters in a Data Set: An Information Theoretic Approach." *Journal of the American Statistical Association*, **98**, 750–763.

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.

A. Timings

The functions `cluster.test()`, `nulldensity()` and `cluster.optimal()` are computationally intensive. Portions of these functions have been coded in C in order to speed up the routines. We recommend that the experimenter time the functions for a small number of simulations in order to estimate the total running time. Tables 1–3 provide some timings on a 2.2 GHz AMD Dual Core Opteron processor with 8 GB RAM.

`n`	`p`	`k`	`mcs`	`nsim`	Elapsed time (sec)
100	10	2	0.1	500000	1346
100	10	4	0.1	500000	1408
100	10	6	0.1	500000	1434
100	20	2	0.1	500000	1501
100	20	4	0.1	500000	1244
100	20	6	0.1	500000	1143

Table 1: Timings for the `cluster.test()` function.

`n`	`p`	`k`	`mcs`	`nsim`	Elapsed time (sec)
100	10	2	0.1	500000	877
100	10	4	0.1	500000	887
100	10	6	0.1	500000	886
100	20	2	0.1	500000	936
100	20	4	0.1	500000	921
100	20	6	0.1	500000	958

Table 2: Timings for the `cluster.optimal()` function.

`n`	`p`	`k`	`mcs`	`nsim`	`prop`	Elapsed time (sec)
50	10	2	0.1	8000	0.25	150
50	10	4	0.1	8000	0.25	5715
50	10	6	0.1	8000	0.25	34266
50	20	2	0.1	8000	0.25	294
50	20	4	0.1	8000	0.25	11573

Table 3: Timings for the `cluster.optimal()` function.

B. Null distribution plots

This section contains plots of how the distribution of $P(H_0|\boldsymbol{Y})$ varies with n and k. It is meant to demonstrate the critical nature of the calibration step.

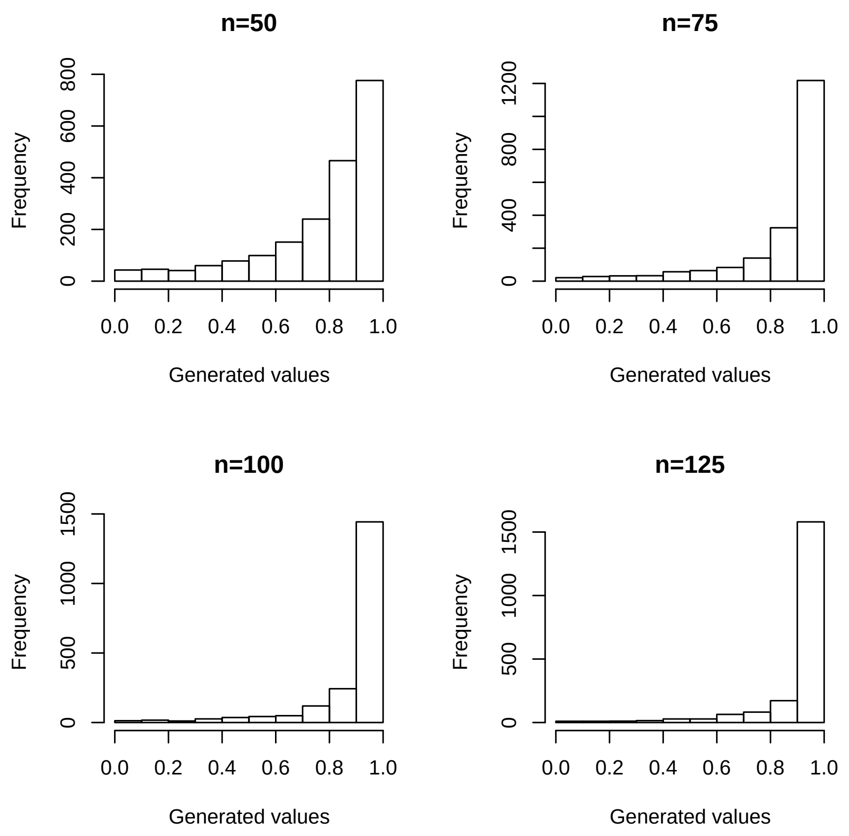

Figure 7: Null distribution under varying n, with $k = 2$ and $mcs = 10\%$.

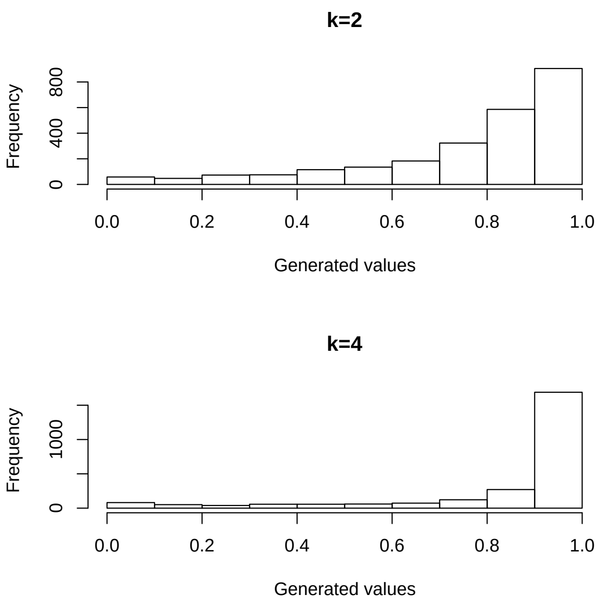

Figure 8: Null distribution under varying k, with $n = 50$ and $mcs = 10\%$.

Affiliation:

Vikneswaran Gopal
IBM Research Collaboratory Singapore
7 Changi Business Park I
The IBM Place
Singapore 468048, Singapore
E-mail: viknes@sg.ibm.com

Claudio Fuentes
Department of Statistics
Oregon State University
44 Kidder Hall
Corvallis, Oregon 97331, United States of America
E-mail: fuentesc@stat.oregonstate.edu

George Casella
Department of Statistics
Institute of Food and Agriculture Sciences
PO Box 110339
University of Florida
Gainesville, Florida 32611-0339, United States of America
E-mail: casella@ufl.edu

Journal of Statistical Software http://www.jstatsoft.org/
published by the American Statistical Association http://www.amstat.org/

Volume 47, Issue 14 *Submitted:* 2009-03-21
May 2012 *Accepted:* 2012-04-13