

Contents

1	Introduction	1
2	Working with a Single Dataset	2
2.1	Carrying out the Cluster Test	2
2.1.1	Without Replications	2
2.1.2	With Replications	5
2.2	The Null Distribution	6
2.2.1	Generating the Null Distribution	6
2.2.2	Extracting Sample Quantiles or Cut-offs	6
2.3	Converting to a Frequentist p -value	8
2.3.1	Without Replications	8
2.3.2	With Replications	8
2.4	Using Cut-off Points to Make a Decision	8
2.4.1	Using the “correct” Null Distribution	8
2.4.2	Using the Default Cut-off Points	9
3	Working with Multiple Datasets/Tests	9
3.1	Without replications	9
3.2	With Replications	10
4	Searching For Optimal Clusters	10

1 Introduction

This user guide provides examples on how to use the R-package `bayesclust` in order to test for the significance of clusters in multivariate data. It demonstrates the workflow that the experimenter would have to go through from start to finish and provides some code that could be useful for the user. The user can also look at the examples in the online `help` for the package.

The overall aim of this suite of functions is to test the following hypothesis:

$$H_0 : \text{No clusters} \quad \text{vs.} \quad H_1 : k \text{ clusters}$$

where $k \in \{2, 3, 4\}$. Although the problem is set-up as one of model selection in a Bayesian framework, the methodology proposed here involves calibrating the computed Empirical Posterior Probability (EPP) to obtain a frequentist p -value, which can then be used to make a decision. Alternatively, the experimenter can use the calibration procedure to simply obtain critical values at any desired α level, and then check the EPP against this critical value to assess the significance of H_1 . The secondary aim of this package is to allow the experimenter to search for an optimal partitioning of the observations into 2,3 or 4 clusters.

A quick summary of the workflow is:

1. Run `cluster.test` on dataset
2. Generate distribution of EPP under the null hypothesis
3. Find p -value of EPP, or compare EPP to critical value obtained from null distribution in Step 3. Reject or accept H_0 .
4. Partition observations into optimal clustering

2 Working with a Single Dataset

The first step is to load the library.

```
> library(bayesclust, version = "1.0")
```

For the purpose of this section, we shall use a randomly generated dataset, with the number of observations $n = 20$ and $p = 2$. Note that the dataset are of class `matrix`, which means that if the dataset has to be read in from a text or csv file, it will have to be converted using `as.matrix` before proceeding. The plot of `data.ex1` can be seen in Figure 1.

```
> data.ex1 <- matrix(rnorm(40), nrow = 20)
> plot(data.ex1, main = "Example Data 1", pch = 20)
```

2.1 Carrying out the Cluster Test

2.1.1 Without Replications

The first line in the code will carry out the test, running the Metropolis-Hastings chain for 2500 iterations, using a random-walk parameter of 60%. For full details on what these parameters mean, please refer to the paper. The second line of the code will plot the running posterior probabilities computed using the output from the Markov Chain. The plot can be seen in Figure 2. This plot is meant to serve as a diagnostic to check convergence of the chain. 2500 iterations are certainly not enough; a minimum of 500,000 are recommended.

```
> test.ex1 <- cluster.test(data.ex1, nsim = 2500, p = 2, aR = 0.6,
+ k = 2)
> plot(test.ex1, main = "Convergence Plot for test.ex1")
```

The `summary` function can be used to obtain a summary of the test that has been run.

```
> summary(test.ex1)
```

Example Data 1

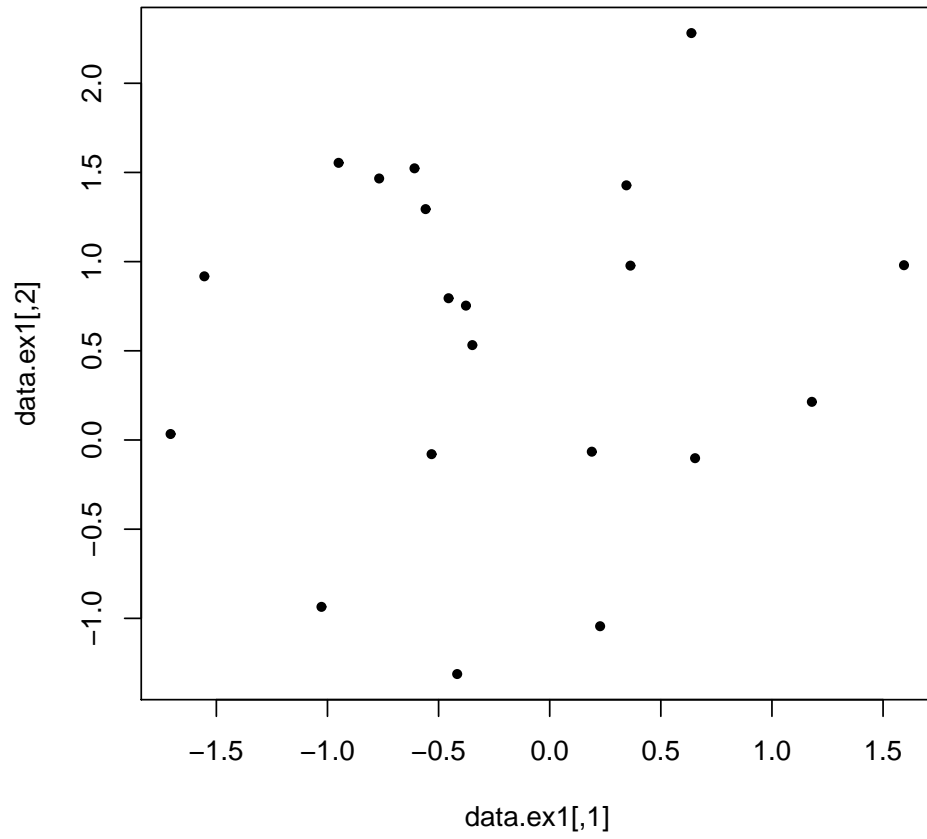


Figure 1: Dataset 1

Convergence Plot for test.ex1

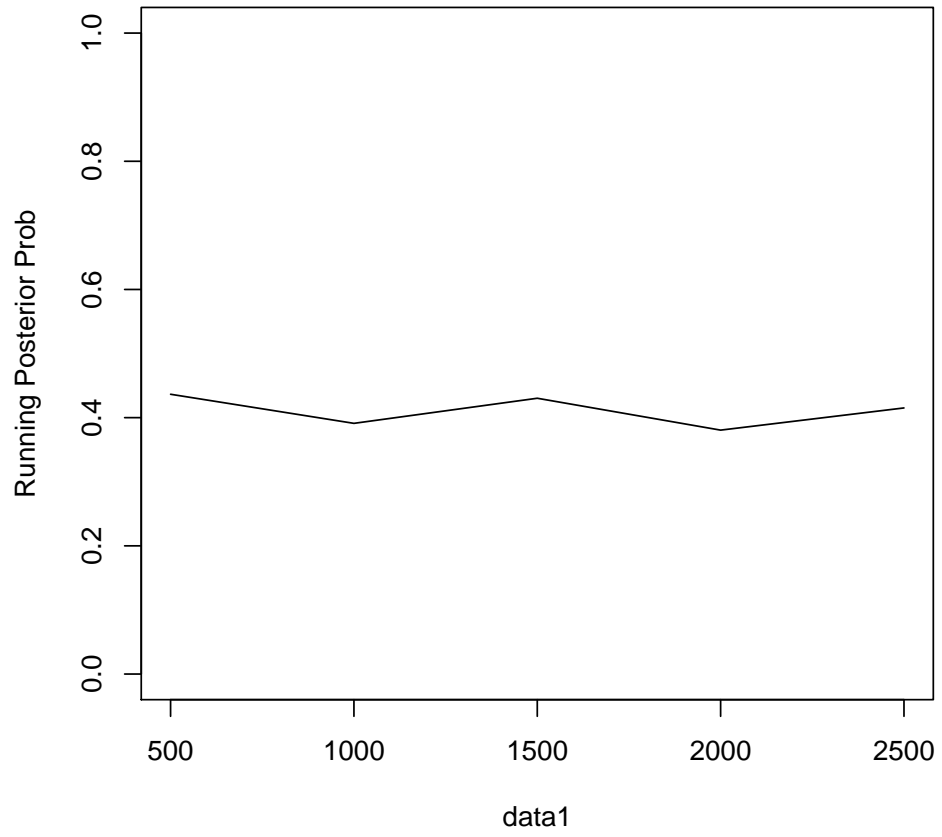


Figure 2: Convergence Plot

Cluster test conducted on data object data1, with 2500 iterations.

```
Num. observations      : 20
Min cluster size      : 4
p                     : 2
H0                    : k = 2
```

```
*****
Final Empirical Posterior Probability:
*****
```

```
      Post.Probs
data1      0.5189
```

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

The object returned, of class “cluster.test”, is a list object. Its components can be viewed with

```
> str(test.ex1)
```

```
List of 3
```

```
$ param      :List of 8
..$ n        : int 20
..$ k        : num 2
..$ min.clust.size: num 4
..$ a        : num 2.01
..$ b        : num 0.99
..$ tau2     : num 1
..$ p        : num 2
..$ data     : num [1:20, 1:2] 1.261 0.973 0.216 0.691 0.472 ...
$ iterations: num [1:5] 500 1000 1500 2000 2500
$ data1     : num [1:5] 0.674 0.594 0.545 0.557 0.519
- attr(*, "class")= chr "cluster.test"
```

Thus if one wishes merely to extract the (final) computed posterior probability, then the command would be

```
> test.ex1$data1[length(test.ex1$data1)]

[1] 0.5189336
```

2.1.2 With Replications

It is highly recommended that replications be carried out for any dataset. This will allow the experimenter to monitor the convergence of the Markov Chain used in computing the EPP, and to check on the variability of the EPP. Here is an example of code that will carry

out 4 replications of a cluster test. It will consolidate the final EPPs in an appropriately named `.csv` file and save the convergence plots to a `.pdf` file. When doing replications, it is important to store the mean EPP, so that it can be compared against the null distribution later.

```
> p <- 2
> RW <- 0.6
> NSIM <- 2500
> K <- 2
> fname.pdf <- paste(paste("convergence_rep1to4", RW, NSIM, K,
+ sep = "_"), ".pdf", sep = "")
> fname.csv <- paste(paste("finalprobs", RW, NSIM, K, sep = "_"),
+ ".csv", sep = "")
> pdf(file = fname.pdf)
> par(mfrow = c(2, 2))
> output.probs <- vector("numeric", 4)
> for (m in 1:4) {
+ tmp <- cluster.test(data.ex1, nsim = NSIM, k = K, aR = RW,
+ p = p)
+ output.probs[m] <- tmp$data1[length(tmp$data1)]
+ plot(tmp)
+ }
> dev.off()
> write.csv(cbind(REP = 1:4, NSIM, RW, K, output.probs), fname.csv,
+ quote = FALSE, row.names = FALSE)
> mean.output.probs <- mean(output.probs)
```

2.2 The Null Distribution

2.2.1 Generating the Null Distribution

This step can in fact be carried out at the same time as section 2.1. It involves generation of the null distribution with the same parameters as the cluster test in section 2.1. The requisite function here is `nulldensity`. Here is the code to generate the object of class “`nulldensity`”, and to then plot its histogram.

```
> test.nulld <- nulldensity(nsim = 1000, n = 20, k = 2, p = 2)
> hist(test.nulld, main = "Histogram of null distribution")
```

The object returned is of class “`nulldensity`”. Again, it is a list object.

2.2.2 Extracting Sample Quantiles or Cut-offs

Sample quantiles from the simulated null distribution can be used as critical values in the hypothesis test. For example, to obtain the cut-off point for an $\alpha = 0.05$ level test, we can do

Histogram of null distribution

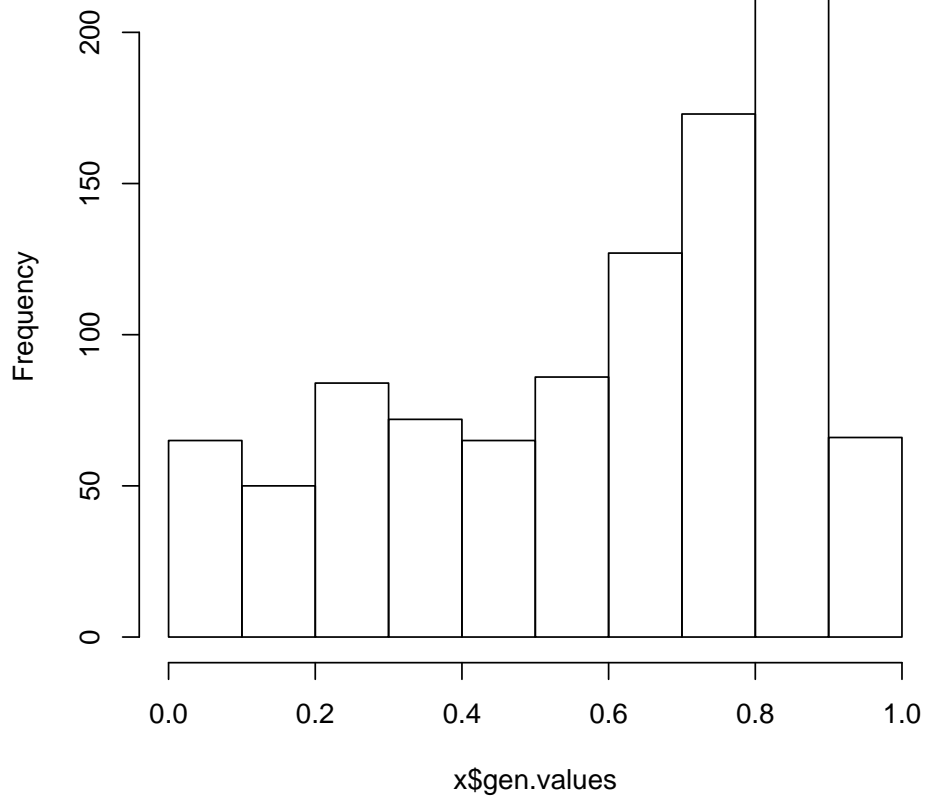


Figure 3: Null Distribution

```
> quantile(test.nulld$gen, 0.05)
```

```
      5%  
0.09050967
```

2.3 Converting to a Frequentist p -value

2.3.1 Without Replications

If no replications were carried out, this task can be achieved using the `emp2pval` function, which ties together objects of class “cluster.test” and “nulldensity” for this exact purpose.

```
> ex1.pval <- emp2pval(test.ex1, test.nulld)  
> ex1.pval$pvals
```

```
      emp.prob pvalue  
data1 0.5189336 0.335
```

2.3.2 With Replications

If replications had been carried out for a particular test, as per section 2.1.2, then it is the mean EPP that needs to be converted to a p -value. Unfortunately, the package does not have this capability (yet), but it can be carried out with these 3 lines of code:

```
> pvalue <- findInterval(mean.output.probs, test.nulld$gen) + 1  
> pvalue <- pvalue/length(test.nulld$gen.values)  
> pvalue <- min(1, pvalue)  
> pvalue
```

```
[1] 0.312
```

Bear in mind that the null distribution needs to be generated to a suitably fine resolution in order for this conversion to be precise.

2.4 Using Cut-off Points to Make a Decision

2.4.1 Using the “correct” Null Distribution

When the null distribution is actually generated for a particular dataset/test, then the sample quantile can be extracted for, say, the $\alpha = 0.05$ level readily (please see section 2.2.2). Let this be $T_{0.05}$. Then if the computed $EPP < T_{0.05}$, the null hypothesis should be rejected at the $\alpha = 0.05$ significance level.

2.4.2 Using the Default Cut-off Points

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

```
> data(cutoffs)
> head(cutoffs)

  n mcs p k cutoff1pct cutoff5pct
1 50 0.1 1 2 0.05767249 0.2237771
2 50 0.1 1 3 0.03765728 0.1866796
3 50 0.1 1 4 0.02362772 0.1527216
4 50 0.1 2 2 0.04247978 0.2220476
5 50 0.1 2 3 0.02626706 0.2103927
6 50 0.1 2 4 0.02161469 0.2019330
```

The experimenter can then look for parameters n, mcs, p, k close to those in his test and use the provided value as his T_α .

3 Working with Multiple Datasets/Tests

When working with multiple datasets or when conducting multiple hypothesis tests, the experimenter would want to control the FDR. The experimenter will need the p -value for each test conducted. These can be obtained through one of the methods described above.

The package provides a function to carry out the FDR procedure when no replications are done. In the situation when replications *are* carried out for each test, this document provides an easy solution. The next few lines of code generate 3 more datasets and run `cluster.test` on them.

```
> data.ex2 <- matrix(rnorm(40), nrow = 20)
> test.ex2 <- cluster.test(data.ex2, nsim = 2500, p = 2, aR = 0.6,
+   k = 2)
> data.ex3 <- matrix(rnorm(40), nrow = 20)
> test.ex3 <- cluster.test(data.ex3, nsim = 2500, p = 2, aR = 0.6,
+   k = 2)
> data.ex4 <- matrix(rnorm(40), nrow = 20)
> test.ex4 <- cluster.test(data.ex4, nsim = 2500, p = 2, aR = 0.6,
+   k = 2)
```

3.1 Without replications

In this case the experimenter would have a series of “emp2pval” objects, and needs to feed them as input to `fdr.test`.

```
> ex2.pval <- emp2pval(test.ex2, test.null.d)
> ex3.pval <- emp2pval(test.ex3, test.null.d)
> ex4.pval <- emp2pval(test.ex4, test.null.d)
> fdr.test(ls(pattern = "pval$"), 0.3)
```

None of the hypothesis are significant when controlling FDR at level $\alpha = 0.3$

3.2 With Replications

Assume that the user has a string of p -values, denoted `pval.string`, one for each test conducted, and that he wishes to control the FDR at 30%.

```
> pval.string <- runif(10)
> fdr.seq <- 1/10 * 0.3
> min(sort(pval.string) >= fdr.seq)
```

```
[1] 0
```

The returned number will inform the experimenter of how many of the smallest p -values to consider to be significant.

4 Searching For Optimal Clusters

Thankfully this section of the package is not at all complicated. There are only 2 functions to be aware of: `cluster.optimal` and `plot`

```
> opt.ex1 <- cluster.optimal(data.ex1, nsim = 2500)
> plot(opt.ex1)
```

To retrieve the 2nd best cluster found, the following command will suffice

```
> opt.ex1$data1$clusters[2, ]

[1] 2 2 1 1 2 1 2 1 2 2 2 1 1 2 2 2 1 1 2 2
```

Optimal Cluster

