Package ‘bayesclust’

May 15, 2012

Type Package

Title Tests/Searches for significant clusters in genetic data.

Version 3.1

Date 2012-05-15

Author Gopal, V. and Fuentes, C. and Casella, G.

Maintainer Gopal, V. <viknesh@stat.ufl.edu>

Description Bayes Clustering Package

License GPL-2

LazyLoad yes

Repository CRAN

Date/Publication 2012-05-15 14:21:23

R topics documented:

bayesclust-package 2
cluster.optimal 3
cluster.test 5
combine 7
cutoffs 8
egDataset1 9
emp2pval 10
hist.nulldensity 11
nulldensity 12
plot.cluster.optimal 14
plot.cluster.test 15
summary.cluster.test 16

Index **18**

`bayesclust-package` *Testing and Searching for Clusters in A Hierarchical Bayes Model*

Description

This package contains a suite of functions that allow the user to carry out the following hypothesis test on genetic data:

$$H_0 : \text{No clusters}$$
$$H_1 : \text{2, 3 or 4 clusters}$$

Details

The hypothesis test is formulated as a model selection problem, where the aim is to identify the model with the highest posterior probability. A Hierarchical Bayes model is assumed for the data. Note that firstly, the null hypothesis is equivalent to saying that the population consists of just one cluster. Secondly, since the functions here only allow the alternative hypothesis to be either 2, 3 or 4 at any one time, the package allows the user to test multiple hypotheses while controlling the False Discovery Rate (FDR).

This is a brief of summary of the test procedure:

1. For a given dataset, compute the empirical posterior probability (EPP) of the null hypothesis using `cluster.test`. EPP will serve as the test statistic in this hypothesis test.
2. Monitor the convergence of EPP by running `plot` on the object returned in Step 1.
3. Generate the distribution of EPP under the null hypothesis using `nulldensity`. This can be done concurrent to Steps 1 and 2. Be sure to use the same parameters for Steps 1 and 3 though.
4. Estimate the p-value of the EPP for this dataset using `emp2pval`. This function takes the objects returned in Steps 1 and 3 as input.
5. Run `cluster.optimal` on significant datasets to pick out optimal clusters.
6. Run `plot` on the object returned in Step 6 to view the optimal clustering/partition of the data.

For full details on the distributional assumptions, please refer to the papers listed in the references section. For further details on the individual functions, please refer to their respective help pages and the examples.

Author(s)

George Casella <casella@stat.ufl.edu> and Claudio Fuentes <cfuentes@stat.ufl.edu> and Vik Gopal <viknesh@stat.ufl.edu>

Maintainer: Vik Gopal <viknesh@stat.ufl.edu>

References

Fuentes, C. and Casella, G. (2009) Testing for the Existence of Clusters. *SORT* **33**(2)

Gopal, V. and Fuentes, C. and Casella, G. (2012) bayesclust: An R package for Testing and Searching for Significant Clusters. *Journal of Statistical Software* **47**(14), 1-21. http://www.jstatsoft.org/v47/i14/

See Also

cluster.test, cluster.optimal, emp2pval, nulldensity

---

cluster.optimal *Search for Optimal Clustering of Dataset*

Description

cluster.optimal will search for the optimal k-clustering of the dataset.

Usage

```
cluster.optimal(data, nsim = 100000, aR = 0.4, p = 2, k = 2,
a = 2.01, b = 0.990099, tau2 = 1, mcs = 0.1,
file = "", label = "data", keep = 4)
```

Arguments

data	data should be a matrix, even when the response is univariate. The number of rows should equal the number of observations, and the number of columns should equal to p.
nsim	The algorithm is based on a stochastic search through the partition space, and nsim corresponds to the number of points in the partition space to inspect. It is recommended that nsim be at least 500,000.
aR	The Metropolis search algorithm samples points from the space of partitions according to a mixture of g and a random walk. aR, which must be a value between 0 and 1, specifies the percentage of time that the random walk is chosen. Please see the references below for further details on the density g.
p	The observations are assumed to come from a multivariate normal distribution, of length p.
k	k must take an integer value strictly greater than 1. It instructs the algorithm to search for the optimal partition of the data into k clusters.
a	a is a hyperparameter for the prior on σ^2. Further details can be found in the references below.
b	Like a, b is also a hyperparameter for the prior on σ^2. Further details can be found in the references below.
tau2	tau2 is a hyperparameter for the prior on the mean μ for each cluster.

`keep` This argument instructs the algorithm to store the top `keep` number of clusters that it finds during it's run. By default, the best 4 clusters that are found will be kept.

`mcs` `mcs` stands for Minimum Cluster Size. It should be a value between 0 and 1. It instructs the algorithm to only consider clusters of a certain minimum size.

`file` This argument is a character string. If specified, the output object will be saved to this (binary) file. It can be loaded, inspected and altered later in subsequent R sessions using `load`. If left unspecified, the object will not be saved to a file and could be lost on quitting the R session.

`label` `label` serves to name the dataset in any given hypothesis test.

Details

A Metropolis search algorithm is run to maximise the marginal of Y, that is, $m(Y|\omega_k)$ where ω_k is a particular partitioning of the data into k clusters. The partitions that yield the highest marginal will be deemed to be optimal.

Since the sample space is so large, the algorithm is started at an intelligent starting point, by running kmeans, but with a crude imposition of the minimum cluster size.

Value

The object returned is a list consisting of 2 components.

`param` The purpose of this component is to store the parameters under which the algorithm was run.

`data` This component will contain a table of the best `keep` clusters and the computed value of the marginal corresponding to those clusters. The latter values are kept as a means of assessing the relative merit of the clusters in the table.

Author(s)

Gopal, V.

References

Fuentes, C. and Casella, G. (2009) Testing for the Existence of Clusters. *SORT* **33**(2)

See Also

`plot.cluster.optimal` to plot the clustered data.

Examples

```
# Generate random 2-variate data
Y <- matrix(rnorm(24), nrow=12)

# Search for optimal partitioning of data into 2 clusters
search1 <- cluster.optimal(Y, nsim=1000, p=2, k=4, keep=5)
```

```
# Plot the best cluster found during search
plot(search1)
```

`cluster.test` *Compute Posterior Probabilities for Dataset*

Description

`cluster.test` computes the empirical posterior probability (EPP) of the null hypothesis in the following test:

$$H_0 : \text{No clusters}$$
$$H_1 : k \text{ clusters}$$

where k takes an integer value strictly greater than 1.

Usage

```
cluster.test(data, nsim = 500000, aR = 0.4, p = 2, k = 2,
a = 2.01, b = 0.990099, tau2 = 1, mcs=0.1, file = "", label = "data",
        replications = 1)
```

Arguments

`data`	`data` should be a matrix, even when the response is univariate. The number of rows should equal the number of observations, and the number of columns should equal to `p`.
`nsim`	As the Bayes Factor (BF) for the hypothesis is computed with the aid of a Metropolis-Hastings (MH) MCMC algorithm, the number of simulations has to be specified. It is recommended that `nsim` be at least 500,000, and that replications be carried out in order to monitor convergence.
`aR`	The candidate distribution in the MH algorithm is a mixture of g and a random walk. `aR`, which must be a value between 0 and 1, specifies the percentage of time that the random walk is chosen. Please see the references below for further details on g.
`p`	The observations are assumed to come from a multivariate normal distribution, of length `p`.
`k`	`k` must take an integer value strictly greater than 1. It specifies the alternative hypothesis in the test.
`a`	`a` is a hyperparameter for the prior on σ^2. Further details can be found in the references below.
`b`	Like `a`, `b` is also a hyperparameter for the prior on σ^2. Further details can be found in the references below.
`tau2`	`tau2` is a hyperparameter for the prior on the mean μ for each cluster.

`mcs` — `mcs` stands for Minimum Cluster Size. It should be a value between 0 and 1. It instructs the test procedure to only consider clusters of a certain minimum size.

`file` — This argument is a character string. If specified, the output object will be saved to this (binary) file. It can be loaded, inspected and alterered later in subsequent R sessions using `load`. If left unspecified, the object will not be saved to a file and could be lost on quitting the R session.

`label` — `label` serves to name the dataset in any given hypothesis test.

`replications` — `replications` will instruct the function to perform multiple replications of the test, allowing the user to monitor convergence of the estimates of the posterior probabilities.

Details

Since the hypothesis test is carried out in a Bayesian framework, the Bayes Factor has to be calculated. As this is an integral over a huge space, the sum is estimated using MCMC. Certain portions of `cluster.test` have been coded in C in order to speed up the simulations.

Value

If `replications` is set to 1, then the output from this function is a list object consisting of three components. It will be assigned S3 class "cluster.test", and can then serve as input to `plot` or `emp2pval`. The components of this list object are:

`param` — This component exists purely for bookkeeping purposes, in the sense that these parameters, under which the test was run, will be checked against the parameters used to generate the null distribution of the test statistic. By default, conversion of the empirical posterior probability to a p-value will only proceed if the parameters match. The user does have the option of ignoring this check though. See `emp2pval` for further details.

`iterations` — This is a vector of indices that will be used to plot the running posterior probabilities when `plot` is called. Since it is superfluous to keep the entire chain, the concept of 'thinning' is applied - only every 500th iteration is stored. Counting begins backwards from the most recent iteration.

`ClusterStat` — This is a 'thinned' vector of running posterior probabilities. The values that are kept correspond exactly to those in the preceding `iterations` component. When `summary.cluster.test` is called, the final posterior probability is extracted and printed in a readable format.

If `replications` is set to an integer greater than 1, then the object returned is of S3 class "cluster.test.reps". This is just a list of objects of class "cluster.test". This "cluster.test.reps" object can also be fed directly into `emp2pval` and `plot`.

Author(s)

Gopal, V.

References

Fuentes, C. and Casella, G. (2009) Testing for the Existence of Clusters. *SORT* **33**(2)

See Also

plot.cluster.test, plot.cluster.test.reps to monitor convergence of computation of posterior probability.

summary.cluster.test, summary.cluster.test.reps to display the computed final posterior probabilities for each dataset run.

Examples

```
# Generate random 2-variate data
Y <- matrix(rnorm(24), nrow=12)

# Search for optimal partitioning of data into 2 clusters
test1 <- cluster.test(Y, nsim=1000, p=2, replications=3)

# Plot the running posterior probabilities to monitor convergence
plot(test1)

# Generate corresponding null density object.
null1 <- nulldensity(nsim=100, n=12, p=2, k=2)

# Convert EPP to p-value
emp2pval(test1, null1)
```

---

combine — *Combine posterior probabilities of null hypothesis under several tests*

Description

In the case where the user knows the maximum number of clusters, this function allows for the tester to combine several simple hypothesis tests on the same data set to gauge the posterior probabilities of k=2, 3, 4 clusters, given the observed data.

Usage

```
combine(...)
```

Arguments

...	The arguments to combine should be of class "cluster.test" or "cluster.test.reps". These should be the results of running a single dataset through the cluster.test function with the exact same parameters, but with different settings for k, i.e. this function is appropriate when testing for differing number of clusters on the same data set.

Details

The objects fed into this function need not all be of the same class. Some could be of class "cluster.test" while others could be of class "cluster.test.reps". The function will take the mean of the replications when computing the Bayes Factor for each test. For details on the calculations, please refer to the references below.

Author(s)

Gopal, V.

References

Fuentes, C. and Casella, G. (2009) Testing for the Existence of Clusters. *SORT* **33**(2)

See Also

`cluster.test` for further information on objects of class "cluster.test" and "cluster.test.reps".

`plot.cluster.test` and `plot.cluster.test.reps` to monitor convergence of computation of posterior probability.

Examples

```
# Generate random 2-variate data
Y <- matrix(rnorm(24), nrow=12)

# Search for optimal partitioning of data into 2 clusters
testK2 <- cluster.test(Y, k=2, p=2, nsim=2000, replications=3)
testK3 <- cluster.test(Y, k=3, p=2, nsim=2000)
testK4 <- cluster.test(Y, k=4, p=2, nsim=2000)

combine(testK2, testK3, testK4)
```

`cutoffs` *Table of Cut-Off Points*

Description

This is a table of pre-computed cut-off points for testing significance of clusters at the $alpha$=0.05 and 0.01 level.

Usage

```
data(cutoffs)
```

Format

A table of cut-off points obtained by generating the null distribution of the posterior probability using the following parameters:

`n` The number of observations in the dataset.

`mcs` `mcs` stands for Minimum Cluster Size.

`p` The length of the vector of each observation. For example, `p=2` corresponds to bivariate data.

`k` The precise (simple) alternative hypothesis being tested.

`cutoff1pct` A numeric vector consisting of the cut-off points for the $alpha$=0.01 level.

`cutoff5pct` A numeric vector consisting of the cut-off points for the $alpha$=0.05 level.

Details

In order to test the significance of the Empirical Posterior Probability (EPP), it is necessary to generate its distribution under H_0, and compare the EPP to the sample quantile for the desired level of significance. However this simulation could take a considerably long time, and hence this table is provided to enable the experimenter to get a crude estimate of critical values at the $alpha$=0.05 and $alpha$=0.01 levels. The parameters under which the table was generated are part of the dataframe, allowing the experimenter to choose the closest set of conditions to his/her particular set-up.

References

Fuentes, C. and Casella, G. (2009) Testing for the Existence of Clusters. *SORT* **33**(2)

Examples

```
data(cutoffs)
## maybe str(cutoffs) ; plot(cutoffs) ...
```

---

egDataset1 *Example dataset used in reference paper.*

Description

Consists of 3 sets of 25 points, each from a multivariate distribution with differing means.

Usage

```
data(egDataset1)
```

References

Fuentes, C. and Casella, G. (2009) Testing for the Existence of Clusters. *SORT* **33**(2)

emp2pval *Convert Empirical Posterior Probability to P-value*

Description

emp2pval converts the Empirical Posterior Probability (EPP) computed by cluster.test into a frequentist p-value, which can then be used to assess the significance of the alternative hypothesis.

Usage

```
## S3 method for class 'cluster.test'
emp2pval(x, y, ignore = FALSE)
## S3 method for class 'cluster.test.reps'
emp2pval(x, y, ignore = FALSE)
```

Arguments

x	x is an object of class "cluster.test" or "cluster.test.reps", which is returned when cluster.test is run on a dataset.
y	y is an object of class "nulldensity", which is returned when nulldensity is run with the necessary parameters provided.
ignore	ignore is a logical variable which specifies whether the parameters in x and y should be matched for consistency.

Details

If ignore is set to FALSE, then the routine will first check to see if the parameters under which the test was run match exactly with the parameters under which the null distribution was generated. If they were, then the EPP's in the "cluster.test" object will be converted to a frequentist p-value by checking the "nulldensity" object to see which empirical quantile they fall in. If ignore is set to TRUE, the same EPP to p-value conversion is carried out, but this time without the preliminary check on the parameters.

If replications had been specified when cluster.test was called, then emp2pval uses the mean of the estimates of the posterior probabilities.

Value

emp2pval returns a list comprising 2 components.

param	This component is a copy of the parameters used when running cluster.test on the dataset. In the case that emp2pval was run with ignore set to TRUE, then this could potentially be different than the parameters under which the nulldensity object was generated.
pvals	A dataframe with the EPP of the dataset and the corresponding frequentist p-values.

Author(s)

Gopal, V.

References

Fuentes, C. and Casella, G. (2009) Testing for the Existence of Clusters. *SORT* **33**(2)

See Also

`cluster.test` for information on objects of class "cluster.test".

`nulldensity` for information on objects of class "nulldensity".

Examples

```
# Generate random 2-variate data
Y <- matrix(rnorm(24), nrow=12)

# Search for optimal partitioning of data into 2 clusters
test1 <- cluster.test(Y, p=2, nsim=2000)

# Generate corresponding null density object.
null1 <- nulldensity(nsim=100, n=12, p=2, k=2)

# Convert EPP to p-value
emp2pval(test1, null1)
```

`hist.nulldensity` *Plot Histogram of Null Density*

Description

`hist.nulldensity` operates on objects of class "nulldensity". It generates a histogram of the values.

Usage

```
## S3 method for class 'nulldensity'
hist(x, ...)
```

Arguments

`x`	An object of class "nulldensity".
`...`	Additional arguments to be passed to the `hist` method.

Author(s)

Gopal, V.

References

Fuentes, C. and Casella, G. (2009) Testing for the Existence of Clusters. *SORT* **33**(2)

See Also

`nulldensity` for further information on objects of class "nulldensity".

`hist` for details on arguments that can be passed through . . .

Examples

```
# Generate null density object.
null1 <- nulldensity(nsim=100, n=12, p=2, k=2)
hist(null1, main="Null density")
```

`nulldensity` *Generate Null Distribution of Empirical Posterior Probability*

Description

In testing the following the hypothesis,

$$H_0 : \text{No clusters}$$
$$H_1 : k \text{ clusters}$$

`nulldensity` generates random variables from the distribution of the Empirical Posterior Probability (EPP) under the null hypothesis.

Usage

```
nulldensity(nsim=8000, n, p = 2, k = 2, a=2.01, b=0.990099,
tau2=1, mcs = 0.1, prop=0.25, file="")
```

Arguments

`nsim`	This denotes the number of random variables to generate from the distriution of EPP under the null. It is recommended to be at least 100,000 when the intention is to carry out multiple testing. Otherwise 8,000-10,000 iterations will suffice.
`n`	`n` is the number of observations in the dataset for testing this hypothesis. See `cluster.test` for more details.
`k`	`k` specifies the alternative hypothesis being tested. It must take an integer value strictly greater than 1.
`mcs`	`mcs` stands for Minimum Cluster Size. It should be a value between 0 and 1. It instructs the test procedure to only consider clusters of a certain minimum size.

a	a is a hyperparameter for the prior on σ^2. Further details can be found in the references below.
b	Like a, b is also a hyperparameter for the prior on σ^2. Further details can be found in the references below.
tau2	tau2 is a hyperparameter for the prior on the mean μ for each cluster.
prop	prop specifies what fraction of the space of partitions under the null hypothesis should be sampled. It is recommended to be at least 0.25.
p	The observations are assumed to come from a multivariate normal distribution, of length p.
file	This argument is a character string. If specified, the output object will be saved to this (binary) file. It can be loaded, inspected and alterered later in subsequent R sessions using load. If left unspecified, the object will not be saved to a file and could be lost on quitting the R session.

Details

The test statistic (EPP) is computed by the function cluster.test. In order to assess the significance of the statistic, it is necessary to obtain the frequentist p-value of the calculated statistic. This package achieves this task by simulating the null distribution of the test statistic with nulldensity and then extracting the sample quantile using emp2pval.

A very small portion of the code has been written in C. The code becomes slower as k gets larger in the alternative hypothesis.

For a particular dataset, this function can be run in parallel with cluster.test.

Value

The object returned is of class "nulldensity". It is a list comprising two components.

param	This component, again, exists purely for bookkeeping purposes. When emp2pval is called, it takes two mandatory arguments - one of class "cluster.test" and the other of class "nulldensity". Both these objects have a parameter component, which should match for the p-value conversion to proceed.
gen.values	This is a vector of length nsim, consisting of the simulations from the null distribution.

Author(s)

Fuentes, C. and Gopal, V.

References

Fuentes, C. and Casella, G. (2009) Testing for the Existence of Clusters. *SORT* **33**(2)

See Also

cluster.test for further information on objects of class "cluster.test".

hist.nulldensity which allows the user to plot a histogram of simulated values in order to view the shape of the null distribution.

Examples

```
# Generate null density object.
null1 <- nulldensity(nsim=100, n=12, p=2, k=2)
hist(null1)
```

`plot.cluster.optimal` *Plot the Optimal Clusters Found in Datasets*

Description

`plot.cluster.optimal` allows the user to plot the optimal clusters found by the search algorithm in `cluster.optimal`.

Usage

```
## S3 method for class 'cluster.optimal'
plot(x, varToPlot=c(1,2), clustToDisp=c(1,2,3,4),
  ...)
```

Arguments

`x`	`x` must be an object of class "cluster.optimal", which is returned when `cluster.optimal` is run on a dataset.
`varToPlot`	`varToPlot` must be vector of length 2, specifying which of the `p` variables should be plotted.
`clustToDisp`	`clustToDisp` must be a vector of length at most 4, indicating how many of the clusters to display.
`...`	Additional arguments to be passed to the inherited `plot`

Details

This function will plot the optimal clusters found, using different colors to represent the different clusters. The function will plot two variables at a time; these can be specified via the `varToPlot` argument. If only the top cluster had been kept, then this will be plotted. If more than one cluster had been retained, then the user can specify which of them should be displayed through the `clustToDisp` argument.

However, for the purpose of keeping the plots readable, only 4 clusterings will be plotted on a given window. For the remaining clusterings, the user can call the function again with a different argument vector for `clustToDisp`.

Author(s)

Gopal, V.

References

Fuentes, C. and Casella, G. (2009) Testing for the Existence of Clusters. *SORT* **33**(2)

See Also

cluster.optimal for further information on objects of class "cluster.optimal".

plot for details on arguments that can be passed through ...

Examples

```
# Generate random 2-variate data
Y <- matrix(rnorm(24), nrow=12)

# Search for optimal partitioning of data into 2 clusters
search1 <- cluster.optimal(Y, p=2, keep=5, nsim=1000)

# Plot the best cluster found during search
plot(search1)
```

---

plot.cluster.test *Plot Running Posterior Probabilities to Monitor Convergence*

Description

This function will plot the running posterior probabilities that were computed by `cluster.test`. These plots can be used to monitor convergence of the computation carried out by `cluster.test`.

Usage

```
## S3 method for class 'cluster.test'
plot(x, ...)
## S3 method for class 'cluster.test.reps'
plot(x, ...)
```

Arguments

`x`	`x` is an object of class "cluster.test", or "cluster.test.reps", which are returned when `cluster.test` is run on a dataset.
`...`	Additional arguments to be passed to the inherited `plot` function.

Details

If only a single replication of the cluster test is carried out, then single plot is displayed on a graphical device. If more than 1 replication had been carried out, then the plots will be arranged in groups of 4, and displayed interactively.

Author(s)

Gopal, V.

References

Fuentes, C. and Casella, G. (2009) Testing for the Existence of Clusters. *SORT* **33**(2)

See Also

`cluster.test` for further information on objects of class "cluster.test" and "cluster.test.reps".

`plot` for details on arguments that can be passed through ...

Examples

```
# Generate random 2-variate data
Y <- matrix(rnorm(24), nrow=12)

# Search for optimal partitioning of data into 2 clusters
test1 <- cluster.test(Y, p=2, nsim=1000, replications=7)

# Plot the running posterior probabilities to monitor convergence
plot(test1)
```

`summary.cluster.test` *Provides Summary of Cluster Test Result*

Description

This function provides a summary of objects of class "cluster.test".

Usage

```
## S3 method for class 'cluster.test'
summary(object, ...)
## S3 method for class 'cluster.test.reps'
summary(object, ...)
```

Arguments

`object`	`object` must be of class "cluster.test" or "cluster.test.reps", which are the classes of objects returned after running `cluster.test`.
`...`	For later expansion.

Author(s)

Gopal, V.

References

Fuentes, C. and Casella, G. (2009) Testing for the Existence of Clusters. *SORT* **33**(2)

See Also

cluster.test for further information on objects of class "cluster.test" and "cluster.test.reps".

plot.cluster.test and plot.cluster.test.reps to monitor convergence of computation of posterior probability.

Examples

```
# Generate random 2-variate data
Y <- matrix(rnorm(24), nrow=12)

# Search for optimal partitioning of data into 2 clusters
test1 <- cluster.test(Y, p=2, nsim=2000, replications=3)

# Summary output
summary(test1)
```

Index

∗Topic **cluster**
- bayesclust-package, 2
- cluster.optimal, 3
- cluster.test, 5
- combine, 7
- cutoffs, 8
- emp2pval, 10
- hist.nulldensity, 11
- nulldensity, 12
- plot.cluster.optimal, 14
- plot.cluster.test, 15
- summary.cluster.test, 16

∗Topic **datasets**
- cutoffs, 8
- egDataset1, 9

∗Topic **htest**
- bayesclust-package, 2
- cluster.optimal, 3
- cluster.test, 5
- combine, 7
- cutoffs, 8
- emp2pval, 10
- hist.nulldensity, 11
- nulldensity, 12
- plot.cluster.optimal, 14
- plot.cluster.test, 15
- summary.cluster.test, 16

∗Topic **models**
- emp2pval, 10
- hist.nulldensity, 11
- nulldensity, 12
- plot.cluster.optimal, 14
- plot.cluster.test, 15

∗Topic **multivariate**
- bayesclust-package, 2
- cluster.optimal, 3
- cluster.test, 5
- combine, 7
- cutoffs, 8
- emp2pval, 10
- hist.nulldensity, 11
- nulldensity, 12
- plot.cluster.optimal, 14
- plot.cluster.test, 15
- summary.cluster.test, 16

∗Topic **package**
- bayesclust-package, 2

bayesclust *(bayesclust-package)*, 2
bayesclust-package, 2

cluster.optimal, 3, *3*, *15*
cluster.test, *3*, 5, *8*, *11–13*, *16, 17*
combine, 7
cutoffs, 8

egDataset1, 9
emp2pval, *3*, 10

hist, *12*
hist.nulldensity, 11, *13*

nulldensity, *3*, *11*, 12, *12*

plot, *15, 16*
plot.cluster.optimal, *4*, 14
plot.cluster.test, *7, 8*, 15, *17*
plot.cluster.test.reps, *7, 8*, *17*
print.emp2pval *(emp2pval)*, 10
print.summary.cluster.test *(summary.cluster.test)*, 16

summary.cluster.test, *7*, 16
summary.cluster.test.reps, *7*