Package 'BetaModels'

September 24, 2026

Version 0.6.3

Date 2026-09-24

Title Bayesian Analysis of Different Rates in Different Groups

Depends R (>= 4.4)

Imports methods, oompaBase, ClassComparison (>= 3.3), grDevices, graphics, stats, splines

Suggests vioplot

Description Test for different proportions (rates) in different groups using a Bayesian model in which all rate parameters follow a beta distribution and are selected from a common hyperdistribution. Includes tools to fit an arbitrary mixture of beta distributions.

License Apache License (== 2.0)

URL http://silicovore.com/OOMPA/index.html

RoxygenNote 7.1.1

NeedsCompilation no

Author Kevin R. Coombes [aut, cre]

Maintainer Kevin R. Coombes <krc@silicovore.com>

Contents

`BetaMixture-class` *Class "BetaMixture"*

Description

Model data on the interval [0,1] as a mixture of an arbitrary number of beta distributions.

Usage

```
BetaMixture(datavec, K = 2, forever = 100, epsilon = 0.001,
            relative = 0.001, debug = FALSE)
## S4 method for signature 'BetaMixture'
summary(object, ...)
## S4 method for signature 'BetaMixture'
hist(x, mixcols = 1:7, ...)
```

Arguments

`datavec`	The observed vector of data points (between 0 and 1).
`K`	The number of mixture components to fit.
`forever`	The maximum number of iterations of the algorithm.
`epsilon`	The minimum change of parameters that must be observed in order to continue iterating.
`relative`	The minimum relative change of parameters that must be observed in order to continue iterating.
`debug`	A local valuecontrolling whethe to print out intermediate steps.
`x`	In the `hist` method, an object of class `BetaMixture`.
`mixcols`	A vector of colors used to show differnt components. Recycled as necessary.
`object`	Object of class `BetaMixture`.
`...`	Extra arguments for generic routines.

Details

Given a data set consisting of values in the interval [0,1], we want to model it as a mixture of K beta distributions:

$$f(x) = \sum_{i=1}^{K} \phi_i B(\nu_i, \omega_i)$$

where the non-negative frequencies or weights of the components sum to 1 (that is, $\sum_{i=1}^{K} \phi_i = 1$).

We fit such a model using en Expectation-Maximation (EM) algorithm. To accomplish this goal, we must introduce a matrix $431Z$ of latent assignments of each of the N observation to the K components. In the current implementation, we intialize teh assignments by dividing the range of the observed data into K subintervals of equal width, and assign the elemnts in the Kth subinterval to component K. (**Warning**: This may change in later versions.) We then alternate between the M-step (finding a maximum likelihood estimate of the parameters of the beta distributions given Z), and the E-step (estimating the weights ϕ by averaging the columns of Z). Iterations continue up to the number specified by `forever` times, or until the log likelihood changes by less than the amount specified by `epsilon`.

Value

The `BetaMixture` constructor returns an object of the indicated class.

The graphical method `hist` and the `summary` method invisibly return the object on which it was invoked.

Creating Objects

Although objects can be created directly using `new`, the most common usage will be to pass a vector of unit interval values to the `BetaMixture` function.

Slots

`datavec`: input data vector of values between 0 and 1.

`phi`: vector of non-negative component frequencies; will sum to 1.

`mle`: $2 \times K$ matrix of parameters of the K beta mixture components.

`Z`: $N \times K$ matrix of the probability for each of the N observations to arise from beta mixture component K. Each row will sum to 1.

`loglike`: The log likelihood of the fitted model.

`converged`: A logical value indicating where the EM algorithm converged.

Methods

summary(object, ...) Prints a summary of the `BetaMixture` object. This includes (1) the frequency/weight of each component, (2) the parameters of each beta distribution, (3) an indicator of convergence, and (4) the log-likelihood of the fimal model.

Author(s)

Kevin R. Coombes <krc@silicovore.com>

Examples

```
showClass("BetaMixture")
set.seed(73892)
datavec <- c(rbeta(130, 1, 4),
             rbeta(170, 7, 4))
model <- BetaMixture(datavec)
summary(model)
hist(model, breaks=35)
```

BetaRates-class *Class "BetaRates"*

Description

Test for different proportions (rates) in different groups using a Bayesian model in which all rate parameters follow a beta distribution and are selected from a common hyperdistribution.

Usage

```
BetaRates(k, n, x=seq(-3,3, length=100), y=x)
## S4 method for signature 'BetaRates'
summary(object, ...)
## S4 method for signature 'BetaRates'
image(x, col=greyscale(128), ...)
## S4 method for signature 'BetaRates'
quantile(x, probs, ...)
samplePosteriorRates(br, nsamp=2000)
expectation(br)
guessCenter(v)
```

Arguments

object	object of class BetaRates
br	object of class BetaRates
x	In the image and quantile methods, an object of class BetaRates. In the BetaRates constructor, a vector of the x-axis grid points at which to compute the posterior probability; see Details.
y	vector of the y-axis grid points at which to compute the posterior probability; see Details.
k	vector of "success" counts
n	vector of all counts
col	vector containing the color map to use for the image
probs	vector of wuantiles to be returned
nsamp	Number of posterior samples to take
v	Vector of observed rates
...	extra arguments for generic routines

Details

TBD

Value

The BetaRates constructor returns an object of the indicated class.

The graphical method image invisibly returns the object on which it was invoked.

The summary method returns a vector with the maximum a posteriori parameters of the beta distribution.

The samplePosteriorRates function returns a list with two components. The first component, xy, is an nsamp-by-2 matrix with x-y values samples from the posterior distribution. The second component, theta, is an nsamp-by-length(k) matrix with posterior samples of the rates associated with each experiment supplied to the constructor.

The guessCenter function returns a list with both x-y and alpha-beta coordinates of the naive (frequentist) estimate of the overall Beta distribution parameters.

The expectation function returns the maximum posterior values from the model. This includes the coordinates (x, y), the transformed coordinates (alpha, beta), the mean (alpha/beta) and the size (alpha + beta).

Creating Objects

Although objects can be created directly using new, the most common usage will be to pass a vector of p-values to the BetaRates function.

Slots

k: vector of "success" counts.

n: vector of all counts.

x: vector of the x-axis grid points at which to compute the posterior probability; see Details.

y: vector of the y-axis grid points at which to compute the posterior probability; see Details.

results: Matrix of posterior probabilities.

logresults: Matrix of log-transformed posterior probabilities.

Methods

summary(object, ...) Prints a summary of the BetaRates object. This includes (1) the maximum a posterior coordinates on x-y-space, (2) the usual alpha-beta parameters for the Beta distribution, and (3) the mean and variance.

image(x, col, ...) Plots an image of the posterior probabilities using the specified color map. The point with the maximum posterior probability is marked in red.

quantile(x, probs, ...) Returns the requested quantiles of the posterior distibution of the binomial rate parameter.

Author(s)

Kevin R. Coombes <krc@silicovore.com>

References

Gelman A, Carlin JB, Stern HS, Rubin DB. Bayesian Data Analysis, second edition. Chapman and Hall/CRC, Boca Raton, 2004. Section 5.3, pages 15-131.

Examples

```
showClass("BetaRates")
event <- c( 37,  4,  6,  1,  2, 10,  1, 13,   7,  1,  10)
total <- c(137, 18, 18, 26, 24, 45, 12, 43, 162, 78, 280)
guessCenter(event/total)
br <- BetaRates(event, total, x=seq(-3, 0, length=100), y=seq(0, 3, length=100))
image(br)
summary(br)
```

---

ebCorrelation-class *Class "ebCorrelation"*

Description

Fit an empirical Bayes model to detemine which values in a large collection of correlation coefficients are significant.

Usage

```
ebCorrelation(ss, nObs, nPoints = 500)
## S4 method for signature 'ebCorrelation'
hist(x,
           xlab='Correlation', ylab='Prob(Different | Y)', main='',
           ...)
## S4 method for signature 'ebCorrelation,missing'
plot(x,
                 prior=1, significance=0.9, ylim=c(-0.5, 1),
                 xlab='Correlation', ylab='Prob(Unusual | Rho)',
                 ...)
## S4 method for signature 'ebCorrelation'
summary(object, prior=1, significance=0.9, ...)
## S4 method for signature 'ebCorrelation'
```

```
cutoffSignificant(object, prior, significance, ...)
## S4 method for signature 'ebCorrelation'
selectSignificant(object, prior, significance, ...)
## S4 method for signature 'ebCorrelation'
countSignificant(object, prior, significance, ...)
```

Arguments

ss	A numerical vector containing correlation coefficinets between -1 and 1.
nObs	A numerical vector of length one, the number of objects used in every computation of correlation coefficients.
nPoints	the number of points at which to estimate the distribution.
object	object of class ebCorrelation
x	object of class ebCorrelation
xlab	Graphical parameter.
ylab	Graphical parameter.
ylim	Graphical parameter.
main	Graphical parameter.
...	Optional extra parameters, either graphical or for the significance functions.
prior	A real number between 0 and 1; the prior probability that a correlation coefficinet is not significant.
significance	A real number between 0 and 1; the posterior probability betyond which a correlation coefficinet will be called significant.

Details

TBD

Value

The ebCorrelation constructor returns an object of the indicated class.

TBD

Creating Objects

Although objects can be created directly using new, the most common usage will be to pass a vector of correlation coefficients to the ebCorrelation function.

Slots

correlation: numeric vector of correlation coefficients.

nObservation: the number of sample observations used to compute correlations.

xvals: vector of the x-axis grid points at which to compute the posterior probability; see Details.

pdf: vector of the ermpirically estimated probability densities at xvals; see Details.

theoretical.pdf: vector of the theoretical probability densities at xvals; see Details.

unravel: Matrix of posterior probabilities.

call: A call object recording how the constructior function was invoked.

Methods

summary(object, ...) Prints a summary of the ebCorrelation object. This includes (1) the maximum a posterior coordinates on x-y-space, (2) the usual alpha-beta parameters for the Beta distribution, and (3) the mean and variance.

image(x, col, ...) Plots an ikmage of the posterior probabilities using te specified color map. The point with the maximum posterior probability is marked in red.

Author(s)

Kevin R. Coombes <krc@silicovore.com>

References

Efron's paper on empirical Bayes for differential exprewssion.

Examples

```
showClass("ebCorrelation")
set.seed(12345)
cc <- c(rbeta(4600, 24, 24), rbeta(400, 8, 8))
rr <- 2*cc-1
fit <- ebCorrelation(rr, 51)
hist(fit)
plot(fit, prior = 0.85)
countSignificant(fit, prior = 0.85, significance = 0.8)
cutoffSignificant(fit, prior = 0.85, significance = 0.8)
summary(fit)
summary(fit, prior = 0.85, significance = 0.8)
```

NegBetaLogLike *Compute the (negative) log-likelihood of a mixture of betas*

Description

Compute the (negative) log-likelihood of a mixture of beta distributions for a given set of observations. We take the negative in order to use the function in a call to `nlm`, which minimizes the objective function, when we actually want to compute the maximum likelihood.

Usage

```
NegBetaLogLike(param, vec, z)
```

Arguments

`param`	A vector of parameters of length 2K
`vec`	A data vector of length N
`z`	A matrix of size NxK of nonegative real numbers, with the sum of each row equal to 1.

Author(s)

Kevin R. Coombes <krc@silicovore.com>

Examples

```
set.seed(73892)
datavec <- c(rbeta(30, 1, 4),
             rbeta(70, 7, 4)) # mixture of two betas

# randomly initialize Z
temp <- sample(2, 100, replace = TRUE)
Z <- matrix(0, nrow = 100, ncol = 2)
for (I in 1:100) Z[I, temp[I]] <- 1

# initialize parameters (2 identical components)
initparam <- rep(1/2, 4)
NegBetaLogLike(initparam, datavec, Z)

# use true parameters
NegBetaLogLike(c(1, 4, 7, 4), datavec, Z)
```

Index