Package 'BayesLCA'

October 12, 2022

Type Package

Title Bayesian Latent Class Analysis

Version 1.9

Date 2020-05-05

Depends e1071, coda

Description Bayesian Latent Class Analysis using several different methods.

License GPL (>= 2)

Imports fields, nlme, MCMCpack

Author Arthur White [aut, cre] (Previous email address: arthur.white@ucdconnect.ie), Thomas Brendan Murphy [aut, ths]

Maintainer Arthur White <arwhite@tcd.ie>

NeedsCompilation no

Repository CRAN

Date/Publication 2020-05-06 17:20:02 UTC

R topics documented:

`BayesLCA-package` *Bayesian Latent Class Analysis*

Description

Bayesian latent class analysis using several different methods.

Details

Package:	BayesLCA
Type:	Package
Version:	1.4
Date:	2015-04-09
License:	GPL (>= 2)
LazyLoad:	yes

Author(s)

Arthur White and Brendan Murphy Maintainer: Arthur White <arthur.white@ucdconnect.ie>

References

Arthur White, Thomas Brendan Murphy (2014). BayesLCA: An R Package for Bayesian Latent Class Analysis." Journal of Statistical Software, 61(13), 1-28. URL: http://www.jstatsoft.org/v61/i13/.

Examples

```
type1 <- c(0.8, 0.8, 0.2, 0.2)
type2 <- c(0.2, 0.2, 0.8, 0.8)
x <- rlca(1000, rbind(type1, type2), c(0.4,0.6))
fit.em <- blca.em(x, 2)
plot(fit.em, which=1)
print(fit.em)
summary(fit.em)
data(Alzheimer)
fit.vb <- blca(Alzheimer, 2, method="vb")
par(mfrow=c(3,3))
plot(fit.vb, which=3:4)
summary(fit.vb)
par(mfrow=c(1,1))
```

Alzheimer *Symptoms of Patients Suffering from Alzheimer's Syndrome*

Description

Presence or absence of 6 symptoms of Alzheimer's disease (AD) in 240 patients diagnosed with early onset AD conducted in the Mercer Institute in St. James's Hospital, Dublin.

Usage

```
data(Alzheimer)
```

Format

A binary matrix, consisting of 240 rows and 6 columns, with each row denoting an individual and each column denoting the presence/absence of one of the 6 symptoms: Hallucination, Activity, Aggression, Agitation, Diurnal and Affective. A 1 denotes the presence of a symptom, a 0 the absence.

Source

Moran M, Walsh C, Lynch A, Coen RF, Coakley D, Lawlor BA (2004) "Syndromes of behavioural and psychological symptoms in mild Alzheimer's disease." *International Journal of Geriatric Psychiatry*, **19(4)**, 359–364. ISSN 1099-1166. doi:10.1002/gps.1091. URL http://dx.doi.org/10.1002/gps.1091.

Walsh C (2006) "Latent Class Analysis Identification of Syndromes in Alzheimer's Disease: A Bayesian Approach." *metodoloyski zvezki - Advances in Methodology and Statistics*, **3(1)**, pp.147 – 162. URL mrvar.fdv.uni-lj.si/pub/mz/mz3.1/walsh.pdf

Examples

```
data(Alzheimer)
fit2 <- blca.em(Alzheimer, 2)
summary(fit2)

fit3<- blca.em(Alzheimer, 3, restarts=20)
summary(fit3)
```

as.mcmc.blca.gibbs *Converts* blca.gibbs *Objects to type* mcmc

Description

Converts blca objects to mcmc objects. This is only to be used with the Gibbs sampling method.

Usage

```
## S3 method for class 'blca.gibbs'
as.mcmc(x, ...)
 blca2mcmc(x)
```

Arguments

x	An object of class blca.gibbs. An error is returned if this is not the case.
...	Additional arguments to be passed to the mcmc function.

Details

Whenever a Gibbs sampler is employed, it is always a good idea to ensure that parameter samples are being obtained correctly - that burn-in has been achieved, and that appropriate mixing is taking place, for example. `as.mcmc.blca.gibbs` converts an object of class `blca` to that of `mcmc` to avail of the diagnostic checks available in other R packages, particularly those in the **coda** package.

Value

An $N \times G * (M + 1)$ matrix of class `mcmc`, where N is the number of data points, M the number of columns and G the number of classes. The first G columns (labelled ClassProb 1 , . . . , ClassProb G) are class membership probability samples, the next G*M columns (labelled ItemProb 1 1 , ItemProb 1 2, . . . , ItemProb G 1, . . . , ItemProb G M) are item response probability samples.

Note

This function replaces the function `mcmc2blca`, which appeared in the original version of the package, and which is retained as an internal function for backwards compatibility reasons.

Author(s)

Arthur White

See Also

`blca.gibbs`, `geweke.diag`, `raftery.diag`

Examples

```
data(Alzheimer)

## Not run:  fit.gibbs <- blca.gibbs(Alzheimer, 2)
## Not run:  raftery.diag(as.mcmc(fit.gibbs))

## Not run:  fit.gibbs <- blca.gibbs(Alzheimer, 2, iter=50000, accept=0.1, burn.in=100)
## Not run:  plot(as.mcmc(fit.gibbs))
```

blca *Bayesian Latent Class Analysis with one of several methods*

Description

Latent class analysis (LCA) attempts to find G hidden classes in binary data X. blca utilises one of: an EM algorithm, a variational Bayes approximation, Gibbs sampling or boot-strapping techniques to find maximum *a posteriori* (MAP), standard error and density estimates of the parameters.

Usage

```
blca(X, G, method = c("em", "gibbs", "boot", "vb"), ...)
```

Arguments

X	The data matrix. This may take one of several forms, see data.blca.
G	The number of classes to run lca for.
method	The method with which to perform lca on the data. Four methods are currently available, "em", "gibbs", "boot" or "vb". Defaults to "em", with a warning.
...	Additional arguments to be passed on, depending on the method. See additional help files for details.

Details

The function calls to one of blca.em, blca.boot, blca.gibbs, blca.vb, depending on the method specified.

Value

A list of class "blca" is returned. All methods return the following items:

classprob	The class probabilities.
itemprob	The item probabilities, conditional on class membership.
Z	Estimate of class membership for each unique datapoint.
prior	A list containing the prior values specified for the model.

See additional help files for details.

Note

Earlier versions of this function erroneously referred to posterior standard deviations as standard errors. This also extended to some of the variable names of the returned function, which are now returned with the corrected suffix blca.em.sd (for standard deviation). For backwards compatability reasons, the earlier suffix .se has been retained as a returned argument.

Author(s)

Arthur White

References

Arthur White, Thomas Brendan Murphy (2014). BayesLCA: An R Package for Bayesian Latent Class Analysis." Journal of Statistical Software, 61(13), 1-28. URL: http://www.jstatsoft.org/v61/i13/.

See Also

blca.em, blca.boot, blca.gibbs, blca.vb

Examples

```
type1 <- c(0.8, 0.8, 0.2, 0.2)
type2 <- c(0.2, 0.2, 0.8, 0.8)
x<- rlca(1000, rbind(type1,type2), c(0.6,0.4))

set.seed(1)
fit <- blca(x, 2) ## EM algorithm used, warning returned
print(fit)   ## No posterior standard deviations returned
summary(fit)

set.seed(1)
fit2 <- blca(x, 2, method="em", sd=TRUE) ##No warning - same fit
print(fit2)   ##Posterior standard deviations returned

set.seed(1)
##Variational Bayes approximation, with priors specified.
fit3 <- blca(x, 2, method="vb", delta=c(5,5), alpha=2, beta=1)
print(fit3) ##Posterior standard deviations returned also.
par(mfrow=c(3,2))
plot(fit3, which=3:4)
par(mfrow=c(1,1))
```

---

blca.boot *Bayesian Latent Class Analysis via an EM Algorithm and Using Empirical Bootstrapping*

Description

Latent class analysis (LCA) attempts to find G hidden classes in binary data X. blca.boot repeatedly samples from X with replacement then utilises an EM algorithm to find maximum posterior (MAP) and standard error estimates of the parameters.

Usage

```
blca.boot(X, G, alpha = 1, beta = 1, delta = rep(1, G),
  start.vals = c("single", "across"), counts.n = NULL,
  fit = NULL, iter = 50, B = 100, relabel = FALSE,
          verbose = TRUE, verbose.update = 10, small = 1e-100)
```

Arguments

X	The data matrix. This may take one of several forms, see data.blca.
G	The number of classes to run lca for.
alpha, beta	The prior values for the data conditional on group membership. These may take several forms: a single value, recycled across all groups and columns, a vector of length G or M (the number of columns in the data), or finally, a $G \times M$ matrix specifying each prior value separately. Defaults to 1, i.e, a uniform prior, for each value.
delta	Prior values for the mixture components in model. Defaults to 1, i.e., a uniform prior. May be single or vector valued (of length G).
start.vals	Denotes how class membership is to be assigned during the initial step of the algorithm. Two character values may be chosen, "single", which randomly assigns data points exclusively to one class, or "across", which assigns class membership via runif. Alternatively, class membership may be pre-specified, either as a vector of class membership, or as a matrix of probabilities. Defaults to "single".
counts.n	If data patterns have already been counted, a data matrix consisting of each unique data pattern can be supplied to the function, in addition to a vector counts.n, which supplies the corresponding number of times each pattern occurs in the data.
fit	Previously fitted models may be supplied in order to approximate standard error and unbiased point estimates. fit should be an object of class "blca.em". Defaults to NULL if no object is supplied.
iter	The maximum number of iterations that the algorithm runs over, for each bootstrapped sample. Will stop earlier if the algorithm converges.
B	The number of bootstrap samples to run. Defaults to 100.
relabel	Logical valued. As the data is recursively sampled, it is possible that label-switching may occur with respect to parameter estimates. If TRUE, parameter estimates are checked at each iteration, and relabeled if necessary. Defaults to FALSE.
verbose	Logical valued. If TRUE, the current number of completed bootstrap samples is printed at regular intervals.
verbose.update	If verbose=TRUE, verbose.update determines the periodicity with which updates are printed.
small	To ensure numerical stability a small constant is added to certain parameter estimates. Defaults to 1e-100.

Details

Bootstrapping methods can be used to estimate properties of a distribution's parameters, such as the standard error estimates, by constructing multiple resamples of an observed dataset, obtained by sampling with replacement from said dataset. The multiple parameter estimates obtained from these resamples may then be analysed. This method is implemented in blca.boot by first running blca.em over the full data set and then using the returned values of the item and class probabilities as the initial values when running the algorithm for each bootstrapped sample. Alternatively, initial parameter estimates may be specified using the fit argument.

Note that if a previously fitted model is supplied, then the prior values with which the model was fitted will be used for the sampling run, regardless of the values supplied to the prior arguments.

Value

A list of class "blca.boot" is returned, containing:

`call`	The initial call passed to the function.
`itemprob`	The item probabilities, conditional on class membership.
`classprob`	The class probabilities.
`Z`	Estimate of class membership for each unique datapoint.
`itemprob.sd`	Posterior standard deviation estimates of the item probabilities.
`classprob.sd`	Posterior standard deviation estimates of the class probabilities.
`classprob.initial, itemprob.initial`	Initial parameter values for classprob and itemprob, used to run over each bootstrapped sample.
`samples`	A list containing the parameter estimates for each bootstrapped sample.
`logpost`	The log-posterior of the estimated model.
`BIC`	The Bayesian Information Criterion for the estimated model.
`AIC`	Akaike's Information Criterion for the estimated model.
`label`	Logical value, indicating whether label switching has been checked for.
`counts`	The number of times each unique datapoint point occured.
`prior`	A list containing the prior values specified for the model.

Note

Earlier versions of this function erroneously referred to posterior standard deviations as standard errors. This also extended to arguments supplied to and returned by the function, some of which are now returned with the corrected corrected suffix `blca.em.sd` (for standard deviation). For backwards compatability reasons, the earlier suffix `.se` has been retained as a returned argument.

Author(s)

Arthur White

References

Wasserman, L, 22nd May 2007, *All of Nonparametric Statistics*, Springer-Verlag.

See Also

blca.em, blca

Examples

```
type1 <- c(0.8, 0.8, 0.2, 0.2)
type2 <- c(0.2, 0.2, 0.8, 0.8)
x <- rlca(1000, rbind(type1,type2), c(0.6,0.4))
fit.boot <- blca.boot(x, 2)
summary(fit.boot)

## Not run: fit <- blca.em(x, 2, se=FALSE)
## Not run: fit.boot <- blca.boot(x, 2, fit=fit)
## Not run: fit.boot
## Not run: plot(fit.boot, which=1:4)
```

---

blca.em *Bayesian Latent Class Analysis via an EM Algorithm*

Description

Latent class analysis (LCA) attempts to find G hidden classes in binary data X. blca.em utilises an expectation-maximisation algorithm to find maximum *a posteriori* (map) estimates of the parameters.

Usage

```
blca.em(X, G, alpha = 1, beta = 1, delta = 1,
start.vals = c("single", "across"), counts.n = NULL,
iter = 500, restarts = 5, verbose = TRUE,
sd = FALSE, se=sd, conv = 1e-06, small = 1e-100)
```

Arguments

X	The data matrix. This may take one of several forms, see data.blca.
G	The number of classes to run lca for.
alpha, beta	The prior values for the data conditional on group membership. These may take several forms: a single value, recycled across all groups and columns, a vector of length G or M (the number of columns in the data), or finally, a $G \times M$ matrix specifying each prior value separately. Defaults to 1, i.e, a uniform prior, for each value.
delta	Prior values for the mixture components in model. Defaults to 1, i.e., a uniform prior. May be single or vector valued (of length G).

start.vals	Denotes how class membership is to be assigned during the initial step of the algorithm. Two character values may be chosen, "single", which randomly assigns data points exclusively to one class, or "across", which assigns class membership via runif. Alternatively, class membership may be pre-specified, either as a vector of class membership, or as a matrix of probabilities. Defaults to "single".
counts.n	If data patterns have already been counted, a data matrix consisting of each unique data pattern can be supplied to the function, in addition to a vector counts.n, which supplies the corresponding number of times each pattern occurs in the data.
iter	The maximum number of iterations that the algorithm runs over. Will stop early if the algorithm is deemed to converge.
restarts	restarts determines how many times the algorithm is run with different starting values. Parameter estimates from the run which achieved the highest log-posterior are returned. If starting values are supplied, these are used for the first run, after which random starting points are used. Defaults to 5.
verbose	Logical valued. If TRUE, the log-posterior from each run is printed.
sd	Specifies whether posterior standard deviation estimates should also be returned. If TRUE, calls to blca.em.sd.
se	Similarly to sd, specifies whether posterior standard deviation estimates should also be returned, however, its use is discouraged. Should always agree with sd. Retained for backwards compatability reasons. See 'Note'.
conv	Convergence criteria, i.e., how small should the log-posterior increase become before the algorithm is deemed to have converged? Set relative to the size of the data matrix.
small	To ensure numerical stability a small constant is added to certain parameter estimates. Defaults to 1e-100.

Details

Regardless of the form of the data supplied to blca.em, it is internally converted to be of the form data.blca. In particular, this should be noted when supplying starting values: the object must be of either the same length or have the same number of rows as the number of **unique** observations in the dataset, as opposed to the total number.

Posterior standard deviations and convergence checks are calculated using blca.em.sd.

Value

A list of class "blca.em" is returned, containing:

call	The initial call passed to the function.
itemprob	The item probabilities, conditional on class membership.
classprob	The class probabilities.
Z	Estimate of class membership for each unique datapoint.
itemprob.sd	If returned, standard error estimates of the item probabilities.

classprob.sd	If returned, standard error estimates of the class probabilities.
logpost	The log-posterior of the estimated model.
BIC	The Bayesian Information Criterion for the estimated model.
AIC	Akaike's Information Criterion for the estimated model.
iter	The number of iterations required before convergence.
poststore	The value of the log-posterior for each iteration.
eps	The value for which the algorithm was deemed to have converged at.
counts	The number of times each unique datapoint point occured.
lpstarts	The log-posterior achieved after each of the multiple starts of the algorithm.
convergence	If posterior standard deviations are calculated, then the Hessian of the model is also checked to determine whether the algorithm has converged to at least a local maximum. The convergence status is calculated by an integer value: 1 denotes acceptable convergence, 2 denotes that it converged at a saddle point, 3 that the algorithm ended before it had satisfactorily converged and 4 denotes that at least one parameter value converged at a boundary value (i.e., a 1 or 0). 0 denotes that the algorithm converged satisfactorily but that the Hessian has not been checked.
prior	A list containing the prior values specified for the model.
sd	A logical value indicating whether standard error estimates were returned.

Note

Earlier versions of this function erroneously referred to posterior standard deviations as standard errors. This also extended to arguments supplied to and returned by the function, some of which are now returned with the corrected corrected suffix `blca.em.sd` (for standard deviation). For backwards compatability reasons, the earlier suffix `.se` has been retained as a returned argument.

Author(s)

Arthur White

References

Dempster AP, Laird NM, Rubin DB (1977). "Maximum Likelihood from Incomplete Data via the EM Algorithm." Journal of the Royal Statistical Society. Series B (Methodological), **39(1)**, pp. 1–38. ISSN 00359246. doi:10.2307/2984875. URL http://dx.doi.org/10.2307/2984875.

See Also

`blca`,`blca.em.sd`, `blca.boot`, `blca.vb`

Examples

```
type1 <- c(0.8, 0.8, 0.2, 0.2)
type2 <- c(0.2, 0.2, 0.8, 0.8)
x <- rlca(1000, rbind(type1,type2), c(0.6,0.4))

fit <- blca.em(x, 2)
print(fit)
fit <- blca.em(x, 2, sd=TRUE) ##Returns posterior standard deviations
summary(fit)
plot(fit)

## Different starting values
fit <- blca.em(x, 2, start.vals="across")
xx <- data.blca(x)
fit <- blca.em(xx, 2, start.vals=sample(1:2, length(xx$counts) , replace=TRUE))
```

`blca.em.sd` *Posterior Standard Deviation Estimates for Bayesian Latent Class Analysis via an EM Algorithm*

Description

Returns posterior standard deviation estimates for point estimates returned by `blca.em`. These are obtained via asymptotic estmation of the Observed Information matrix. The Hessian of the log-posterior is also checked to determine whether point estimates occur at at least a local maximum.

Usage

```
blca.em.sd(fit, x, counts.n = 1)
```

Arguments

`fit`	An object of class "blca.em".
`x`	A binary matrix. An object of class `data.blca` may also be supplied. In this case the argument `counts.n` is ignored.
`counts.n`	A vector which supplies the corresponding number of times each pattern in X occurs in the data.

Details

This function is primarily intended for use in conjunction with `blca.em`, and may be called directly by that function by setting `se=TRUE`. However it can in fact be used with any blca object.

Value

A list containing:

`itemprob`	Posterior standard deviation estimates of the item probabilities.
`classprob`	Posterior standard deviation estimates of the class probabilities.
`convergence`	An integer value denoting whether point estimates occur at at least a local maximum. 1 denotes acceptable convergence, 2 denotes that it converged at a saddle point, 3 that the algorithm ended before it converged and 4 denotes that at least one parameter value converged at a boundary value.

Note

The posterior standard deviation estimates are derived asymptotically, i.e., by inverting the information matrix of the parameters. These values are known to be unreliable in cases where parameters estimates are close to 1 or 0, so caution is advised when checking their values. Bootstrapping methods may provide better estimates.

Computationally, the method becomes becomes unstable for values close to 1 or 0. If the distance of any of the supplied parameter values from 0 or 1 is <1e-5, then posterior standard deviation estimates for these values are returned as 0.

Earlier versions of this function erroneously referred to posterior standard deviations as standard errors. This also extended to the function name, which has now been corrected to `blca.em.sd` (for standard deviation). For backwards compatability reasons, the earlier function `blca.em.se` has been retained as an in internal function.

Author(s)

Arthur White

See Also

`blca.em`, `blca.boot`

Examples

```
type1 <- c(0.8, 0.8, 0.2, 0.2)
type2 <- c(0.2, 0.2, 0.8, 0.8)
x<- rlca(1000, rbind(type1,type2), c(0.6,0.4))
dat<- data.blca(x)

set.seed(1)
fit1 <- blca.em(dat, 2, se=TRUE)
fit1$itemprob.sd
fit1$classprob.sd

set.seed(1)
fit2<- blca.em(dat, 2, se=FALSE)
fit2.sd<- blca.em.sd(fit2, dat)
fit2.sd$itemprob
fit2.sd$classprob
```

`blca.gibbs` *Bayesian Latent Class Analysis via Gibbs Sampling*

Description

Latent class analysis (LCA) attempts to find G hidden classes in binary data X. blca.gibbs performs Gibbs sampling to sample from the parameters' true distribution.

Usage

```
blca.gibbs(X, G, alpha = 1, beta = 1, delta = 1,
   start.vals = c("prior", "single", "across"),
   counts.n = NULL, iter = 5000, thin = 1,
   accept=thin, burn.in = 100, relabel = TRUE,
           verbose = TRUE, verbose.update = 1000)
```

Arguments

`X`	The data matrix. This may take one of several forms, see `data.blca`.
`G`	The number of classes to run lca for.
`alpha, beta`	The prior values for the data conditional on group membership. These may take several forms: a single value, recycled across all groups and columns, a vector of length G or M (the number of columns in the data), or finally, a G x M matrix specifying each prior value separately. Defaults to 1, i.e, a uniform prior, for each value.
`delta`	Prior values for the mixture components in model. Defaults to 1, i.e., a uniform prior. May be single or vector valued (of length G).
`start.vals`	Denotes how class membership is to be assigned during the initial step of the algorithm. One of three character values may be chosen: "prior", which samples parameter values from prior distributions, "single", which randomly assigns data points exclusively to one class, or "across", which assigns class membership via `runif`. Alternatively, class membership may be pre-specified, either as a vector of class membership, or as a matrix of probabilities. Defaults to "single".
`counts.n`	If data patterns have already been counted, a data matrix consisting of each unique data pattern can be supplied to the function, in addition to a vector counts.n, which supplies the corresponding number of times each pattern occurs in the data.
`iter`	The number of iterations to run the gibbs sampler for **after** burn-in.
`thin`	The thinning rate for samples from the distribution, in order to achieve good mixing. Should take a value greater >0 and <=1. Defaults to 1.
`accept`	Similarly to `accept`, specifies the thinning rate for samples from the distribution, in order to achieve good mixing, however, its use is discouraged. Should always agree with `sd`. Retained for backwards compatability reasons. See 'Note'.
`burn.in`	Number of iterations to run the Gibbs sampler for before beginning to store values.

relabel	Logical, indicating whether a mechanism to prevent label-switching be used or not. Defaults to TRUE.
verbose	Logical valued. If TRUE, the current number of completed samples is printed at regular intervals.
verbose.update	If verbose=TRUE, verbose.update determines the periodicity with which updates are printed.

Details

The library coda provide extensive tools to diagnose and visualise MCMC chains. The generic function as.mcmc.blca.gibbs, makes blca.gibbs objects compatible with functions such as summary.mcmc and raftery.diag.

Value

A list of class "blca.gibbs" is returned, containing:

call	The initial call passed to the function.
classprob	The class probabilities.
itemprob	The item probabilities, conditional on class membership.
classprob.sd	Posterior standard deviation estimates of the class probabilities.
itemprob.sd	Posterior standard deviation estimates of the item probabilities.
logpost	The log-posterior of the estimated model.
Z	Estimate of class membership for each unique datapoint.
samples	A list containing Gibbs samples of the item and class probabilities and log-posterior.
DIC	The Deviance Information Criterion for the estimated model.
BICM	The Bayesian Information Criterion (Monte Carlo) for the estimated model.
AICM	Akaike's Information Criterion (Monte Carlo) for the estimated model.
counts	The number of times each unique datapoint point occured.
prior	A list containing the prior values specified for the model.
thin	The acceptance rate for samples from the distribution.
burn.in	The number of iterations the gibbs sampler was run before beginning to store values.
relabel	Logical, indicating whether a mechanism to prevent label-switching was used.
labelstore	The stored labels during the sampling run. If relabel=TRUE, these show how labels were permuted in an attempt to avoid label-switching in the model.

Note

Earlier versions of this function erroneously referred to posterior standard deviations as standard errors. This also extended to arguments supplied to and returned by the function, some of which are now returned with the corrected corrected suffix blca.em.sd (for standard deviation). For backwards compatability reasons, the earlier suffix .se has been retained as a returned argument. The argument thin replaces accept, which appeared in the earliest version of the package. This is to maintain consistency with other packages, such as **rjags**.

Author(s)

Arthur White

References

Spiegelhalter DJ, Best NG, Carlin BP, Linde Avd (2002). "Bayesian Measures of Model Complexity and Fit." Journal of the Royal Statistical Society. Series B (Statistical Methodology), 64(4), pp. 583-639. ISSN 13697412. URL http://www.jstor.org/stable/3088806.

Raftery AE, Newton MA, Satagopan JM, Krivitsky PN (2007). "Estimating the integrated likelihood via posterior simulation using the harmonic mean identity." In Bayesian Statistics, pp. 1-45.

See Also

blca, as.mcmc.blca.gibbs, raftery.diag

Examples

```
## Generate a 4-dim. sample with 2 latent classes of 500 data points each.
## The probabilities for the 2 classes are given by type1 and type2.

type1 <- c(0.8, 0.8, 0.2, 0.2)
type2 <- c(0.2, 0.2, 0.8, 0.8)
x<- rlca(1000, rbind(type1,type2), c(0.6,0.4))

## Not run: fit.gibbs<-blca.gibbs(x,2, iter=1000, burn.in=10)
## Not run: summary(fit.gibbs)
## Not run: plot(fit.gibbs)
## Not run: raftery.diag(as.mcmc(fit.gibbs))


## Not run: fit.gibbs<-blca.gibbs(x,2, iter=10000, burn.in=100, thin=0.5)
## Not run: plot(fit.gibbs, which=4)
## Not run: raftery.diag(as.mcmc(fit.gibbs))
```

`blca.vb` *Bayesian Latent Class Analysis via a variational Bayes algorithm*

Description

Latent class analysis (LCA) attempts to find G hidden classes in binary data X. blca.vb uses a variational EM algorithm to find the distribution which best approximates the parameters' true distribution.

Usage

```
blca.vb(X, G, alpha = 1, beta = 1, delta = 1,
start.vals = c("single", "across"), counts.n = NULL,
        iter = 500, restarts = 1, verbose = TRUE, conv = 1e-06,
small = 1e-100)
```

Arguments

X	The data matrix. This may take one of several forms, see data.blca.
G	The number of classes to run lca for.
alpha, beta	The prior values for the data conditional on group membership. These may take several forms: a single value, recycled across all groups and columns, a vector of length G or M (the number of columns in the data), or finally, a G x M matrix specifying each prior value separately. Defaults to 1, i.e, a uniform prior, for each value.
delta	Prior values for the mixture components in model. Defaults to 1, i.e., a uniform prior. May be single or vector valued (of length G).
start.vals	Denotes how class membership is to be assigned during the initial step of the algorithm. Two character values may be chosen, "single", which randomly assigns data points exclusively to one class, or "across", which assigns class membership via runif. Alternatively, class membership may be pre-specified, either as a vector of class membership, or as a matrix of probabilities. Defaults to "single".
counts.n	If data patterns have already been counted, a data matrix consisting of each unique data pattern can be supplied to the function, in addition to a vector counts.n, which supplies the corresponding number of times each pattern occurs in the data.
iter	The maximum number of iterations that the algorithm runs over. Will stop earlier if the algorithm converges.
restarts	restarts determines how many times the algorithm is run with different starting values. Parameter estimates from the run which achieved the highest log-posterior are returned. If starting values are supplied, these are used for the first run, after which random starting points are used. Defaults to 1.
verbose	Logical valued. If TRUE, the log-posterior from each run is printed.
conv	Convergence criteria, i.e., how small should the log-posterior increase become before the algorithm is deemed to have converged? Set relative to the size of the data matrix.
small	To ensure numerical stability a small constant is added to certain parameter estimates. Defaults to 1e-100.

Details

The variational Bayes method approximates the posterior using as a product of independent distributions. Parameters are then estimated for this approximate distribution using a variational EM algorithm. This method has a tendency to underestimate parameter's variance; as such the standard error and density estimates should be interpreted with caution.

While it is worth starting the algorithm from multiple starting points, variational algorithms have less of a tendency to cpnverge at saddle point or sub-optimal local maxima.

Value

A list of class "blca.vb" is returned, containing:

call	The initial call passed to the function.
itemprob	The item probabilities, conditional on class membership.
classprob	The class probabilities.
itemprob.sd	Posterior standard deviation estimates of the item probabilities.
classprob.sd	Posterior standard deviation estimates of the class probabilities.
parameters	A list containing posterior parameter values for item and class probabilities, which are assumed to follow beta and Dirichlet distributions respectively.
Z	Estimate of class membership for each unique datapoint.
LB	The lower bound estimate of the log-posterior of the estimated model.
lbstore	The value of the lower bound estimate for each iteration.
iter	The number of iterations required before convergence.
eps	The amount that the lower bound increased at the final iteration of the algorithm's run.
counts	The number of times each unique datapoint point occured.
prior	A list containing the prior values specified for the model.

Note

Variational Bayes approximations, are known to often underestimate the standard errors of the parameters under investigation, so caution is advised when checking their values.

Earlier versions of this function erroneously referred to posterior standard deviations as standard errors. This also extended to arguments supplied to and returned by the function, some of which are now returned with the corrected corrected suffix `blca.em.sd` (for standard deviation). For backwards compatability reasons, the earlier suffix `.se` has been retained as a returned argument.

Author(s)

Arthur White

References

Ormerod J, Wand M (2010). "Explaining Variational Approximations." The American Statistician, 64(2), 140-153.

See Also

`blca.em`, `blca.gibbs`

Examples

```
type1 <- c(0.8, 0.8, 0.2, 0.2)
type2 <- c(0.2, 0.2, 0.8, 0.8)
x<- rlca(1000, rbind(type1,type2), c(0.6,0.4))

fit <- blca.vb(x, 2)
print(fit)
```

```
summary(fit)
par(mfrow=c(3,2))
plot(fit)
par(mfrow=c(1,1))

data(Alzheimer)
sj <- blca.vb(Alzheimer, 10, delta=1/10)
sj$classprob    ##Empty Groups
```

`data.blca` *Conveniently Format Data for Bayesian Latent Class*

Description

Conveniently format data for use with `blca`.

Usage

```
data.blca(X)
```

Arguments

`X`	A data matrix intended for latent class analysis. See details.

Details

The data may take of one of two forms, either as a binary matrix, or as a matrix consisting of unique binary rows, with a column of counts. In either case `data.blca` will convert X into a list, with binary matrix and count vector explicitly identified.

Value

A list of class data.blca, containing

`counts.n`	A vector of counts of each unique data entry.
`data`	A matrix consisting of each unique data entry.

Note

This function is used internally by `blca`, so its use is not necessary, though it will speed up computation time to supply the model with data of this form if repeated use of a function is required.

Author(s)

Arthur White

See Also

`blca`

Examples

```
type1 <- c(0.8, 0.8, 0.2, 0.2)
type2 <- c(0.2, 0.2, 0.8, 0.8)
x<- rlca(1000, rbind(type1,type2), c(0.6,0.4)) ##Only 16 unique observations possible

data.blca(x)
```

MAP *Maximum* a posteriori *(MAP) Classification*

Description

MAP obtains maximum *a posteriori* (MAP) classifications. unMAP converts a classification vector into an indicator matrix.

Usage

```
MAP(mat, tie = c("random", "standard"))
unMAP(vec)
```

Arguments

mat	An $N \times G$ matrix, typically N denotes observations from a dataset and G denotes the number of underlying groups in the data. Each row is expected to contain positive entries which sum to 1, but this isn't strictly necessary.
tie	May take one of two values, "random" or "standard". Takes the value "random" by default. See 'Details'.
vec	An vector consisting of integer entries. unMAP is intended to be used with a vector whose entries are classifications of dataset observations.

Details

For each row in mat, MAP assigns an indexing value identifying the entry in the row taking the highest value. In the case where multiple values in a row share a common largest value, tie determines how such a value is chosen. If tie = "random", one of the suitable values is chosen at random; when tie = "standard", the first such suitable value is selected, in common with other packages. Defaults to "random".

Value

MAP returns a classification vector. unMAP returns a classification matrix, with each row indicating group membership by the column entry which is non-zero (and equal to one).

Author(s)

Arthur White

See Also

Zscore

Examples

```
##Simple example
s1<- sample(1:2, 10, replace=TRUE)
unMAP(s1)
MAP(unMAP(s1))

##More to the point
data(Alzheimer)
fit<- blca.em(Alzheimer, 2)
MAP(fit$Z) ## Best estimates of group membership.

mat1<- matrix(1/3, nrow=10, ncol=3) ##demonstrating the use of "tie" argument
MAP(mat1, tie="random")
MAP(mat1, tie="standard")
```

---

plot.blca *Plot Parameter Summaries, Density Estimates and Model Diagnostics for Bayesian Latent Class Analysis*

Description

Five plots are selectable: a plot summarising item and class probability, a mosaic plot representing classification uncertainty, item probability density estimates, conditional class probability density estimates, and a diagnostics plot. The default setting is for the first four plots to be displayed, with the exception of plot.blca.em, which cannot produce density plots and so only produces the first two plots by default.

Usage

```
## S3 method for class 'blca'
plot(x, which = 1L, main = "", col1 = heat.colors(12), ...)
```

Arguments

x	An object of class blca.
which	Which plots to select. May be any subset of 1:5, with some exceptions. See 'Details'.
main	An overall title for the plot: see title.
col1	Specifies a list of colours to be used by the heat map plot used when which = 1.image. Uses heat.colors by default, but several other choices are available. See the help files of image.plot, image and palette for details.
...	Further arguments to be passed onto the plotting devices. When which = 1, the plotting device is image.plot, mosaicplot in the case of which=2, and when which=3:5, plot.

Details

Not all plots are available for some object classes. If the object is of class `blca.em`, density plots (`which = 3:4`) are unavailable, and a warning is returned. Similarly, diagnostic plots (`which = 5`) for `blca.boot` objects are unavailable.

The available diagnostic plots differ depending on the class of the object in question. For `blca.em` and `blca.vb` objects, the plot is intended as visual aid to check whether the respective algorithms have converged, i.e., that the log-posterior or lower bound have ceased increasing after successive iterations. The main aim of the diagnostic plot for `blca.gibbs` objects is to visually check diagnostic measures such as mixing and burn-in, and also to assess whether label-switching has occurred, or been corrected for satisfactorily.

Currently, the colors used in a plot can only be specified directly for `which = 1`. For classification uncertainty (`which = 2`) and density plots (`which = 3:4`), each group is colored by the `palette` function so that Group g takes color `palette()[g+1]`. For the default settings, Group 1 will then be colored red, Group 2 green, and so on.

Author(s)

Arthur White

References

Arthur White, Thomas Brendan Murphy (2014). BayesLCA: An R Package for Bayesian Latent Class Analysis." Journal of Statistical Software, 61(13), 1-28. URL: http://www.jstatsoft.org/v61/i13/.

See Also

`image.plot`, `mosaicplot`.

Examples

```
type1 <- c(0.8, 0.8, 0.2, 0.2)
type2 <- c(0.2, 0.2, 0.8, 0.8)
x<- rlca(1000, rbind(type1,type2), c(0.6,0.4))

fit <- blca.em(x, 2)
plot(fit, which = 1:2) ## Parameter summary and classification uncertainty plots.

palette(rainbow(6)) ## Change color scheme
plot(fit, which = 2)
palette("default") ## Restore default color scheme

fit2<- blca.vb(x,2)
par(mfrow = c(3,4))
plot(fit2, which = 3) ## Approximate density plots for item probability parameters.
par(mfrow = c(1,1))
```

print.blca *Bayesian Latent Class Analysis*

Description

Print a blca object.

Usage

```
## S3 method for class 'blca'
print(x, ...)
```

Arguments

x	An object of class blca.
...	Additional arguments to be passed onto lower-level functions at a later stage of development.

Details

Prints parameter maximum *a posteriori* (map) and standard deviation estimates. The latter are sometimes unavailable for blca.em objects.

Value

The blca object itself.

Author(s)

Arthur White

Examples

```
data(Alzheimer)
fit1<- blca(Alzheimer, 2, method="em")
class(fit1)
print(fit1)
fit2<- blca(Alzheimer, 2, method="em", sd=TRUE)
print(fit2) ## Standard Errors also printed

fit3<- blca(Alzheimer, 2, method="vb")
print(fit3) ## Standard Errors as standard
```

rlca *Randomly Generate Binary Data with Underlying Latent Classes*

Description

A function which randomly generates data with respect to some underlying latent class. Data may be generated either by specifying item and class probabilities or by utilising an object previously fitted to data.

Usage

```
rlca(n, itemprob = 0.5, classprob = 1, fit = NULL)
```

Arguments

n	Number of data points to be generated.
itemprob	The item probabilities, conditional on class membership. Defaults to 0.5.
classprob	The class probabilities. Defaults to 1, i.e., a one class model.
fit	An object of class blca. If fit is supplied, data is generated using the class and item probabilities obtained. Defaults to NULL.

Author(s)

Arthur White

See Also

data.blca

Examples

```
type1 <- c(0.8, 0.8, 0.2, 0.2)
type2 <- c(0.2, 0.2, 0.8, 0.8)
x<- rlca(1000, rbind(type1,type2), c(0.6,0.4))

fit <- blca.em(x, 2)

x2<- rlca(1000, fit=fit)
fit2<- blca.em(x2,2)
```

summary.blca *Bayesian Latent Class Analysis*

Description

Summary method for class "blca".

Usage

```
## S3 method for class 'blca'
summary(object, ...)
```

Arguments

object	Object of class blca.
...	Additional arguments to be passed onto lower-level functions at a later stage of development.

Value

A brief summary consisting of two parts: the prior values specified to the model, and model diagnostics specific to the inference method used, such as information about the log-posterior (or lower bound in the case of blca.vb), as well the number of iterations the algorithm ran for, etc..

Author(s)

Arthur White

Examples

```
data(Alzheimer)
summary(blca.em(Alzheimer, 2))
summary(blca.vb(Alzheimer, 2, alpha=2, beta=2, delta=0.5))

## Not run: (fit.gibbs)<- blca.gibbs(Alzheimer, 2, delta=2)
## Not run: summary(fit.gibbs)
```

Zscore *Evaluating Class Membership of Binary Data*

Description

For a fitted model of class blca, and binary data X, the probability of class membership for each data point is provided.

Usage

```
Zscore(X, fit = NULL, itemprob = NULL, classprob = NULL)
```

Arguments

X	A binary data matrix. X must have the same number of columns as the data that fit was applied to.
fit	An object of class blca.
itemprob	A matrix of item probabilities, conditional on class membership.
classprob	A vector denoting class membership probability.

Details

Calculation of the probability of class membership for a data point relies on two parameters, class membership and item probability. These may be supplied directly to Zscore, or alternatively, a blca object containing both parameters can be used instead.

Value

A matrix of equal rows to X and with G, the number of classes, columns, where each row is a score denoting the probability of class membership. Each row should therefore sum to 1.

Note

Zscore.internal has the same functionality as Zscore, but is only intended for internal use.

Author(s)

Arthur White

Examples

```
set.seed(1)
type1 <- c(0.8, 0.8, 0.05, 0.2)
type2 <- c(0.2, 0.2, 0.05, 0.8)
x<- rlca(250, rbind(type1,type2), c(0.5,0.5))

fit <- blca.em(x, 2)
fit$Z ## Unique data types
```

```
Zscore(x, fit=fit) ## Whole data set
Zscore(c(0, 1, 1, 0), fit=fit) ## Not in data set
Zscore(x, itemprob=rbind(type1,type2), classprob=c(0.5,0.5))
```

Index