Package 'dbd'

July 22, 2025

Version 0.0-22

Date 2021-08-19

Title Discretised Beta Distribution

Author Rolf Turner <r.turner@auckland.ac.nz>

Maintainer Rolf Turner <r.turner@auckland.ac.nz>

Description Tools for working with a new versatile discrete distribution, the db (``discretised Beta'') distribution. This package provides density (probability), distribution, inverse distribution (quantile) and random data generation functions for the db family. It provides functions to effect conveniently maximum likelihood estimation of parameters, and a variety of useful plotting functions. It provides goodness of fit tests and functions to calculate the Fisher information, different estimates of the hessian of the log likelihood and Monte Carlo estimation of the covariance matrix of the maximum likelihood parameter estimates. In addition it provides analogous tools for working with the beta-binomial distribution which has been proposed as a competitor to the db distribution.

Depends R (>= 3.2.2)

Suggests hmm.discnp, MASS, rmutil, spcadjust

LazyData true

License GPL (>= 2)

NeedsCompilation no

Repository CRAN

Date/Publication 2021-08-19 13:40:05 UTC

Contents

`aHess` *Analytic hessian.*

Description

Compute the hessian of the **negative** log likelihood of a db or beta binomial distribution from an analytic expression for this quantity.

Usage

```
aHess(object,x)
```

Arguments

`object`	An object of class `"mleDb"` or `"mleBb"` as returned by the function `mleDb()` or the function `mleBb()`.
`x`	A numeric vector of observations appropriate for the model that was fitted to produce `object`. Needed only if `object` is of class `"mleBb"`; the hessian for the db distribution depends only upon the parameters and not upon the data.

Details

This function is essentially the same as the `finfo()` functions and differs from it only in that it is designed to act up `"mleDb"` or `"mleBb"` objects, from which (estimates of) the relevant parameters are extracted.

Value

A two-by-two positive definite (with any luck!) numeric matrix. Its inverse is an estimate of the covariance matrix of the parameter estimates.

Author(s)

Rolf Turner `<r.turner@auckland.ac.nz>`

See Also

`nHess()` `finfo()` `mleDb()`

Examples

```
X   <- hmm.discnp::SydColDisc
X$y <- as.numeric(X$y)
X   <- split(X,f=with(X,interaction(locn,depth)))
x   <- X[[19]]$y
fit <- mleDb(x, ntop=5)
H   <- aHess(fit)
print(solve(H)) # Equal to ...
print(vcov(fit))
X     <- hrsRcePred
top1e <- X[X$sbjType=="Expert","top1"]
fit   <- mleBb(top1e,size=10)
H     <- aHess(fit,x=top1e)
print(solve(H)) # Equal to ...
print(vcov(fit))
```

---

db *The db ("discretised Beta") distribution.*

Description

Density, distribution function, quantile function and random generation for the db distribution with parameters `alpha`, `beta` and `ntop`.

Usage

```
ddb(x, alpha, beta, ntop, zeta=FALSE, log=FALSE)
pdb(x, alpha, beta, ntop, zeta=FALSE)
qdb(p, alpha, beta, ntop, zeta=FALSE)
rdb(n, alpha, beta, ntop, zeta=FALSE)
```

Arguments

x	Numeric vector of values at which the "density" (probability mass function) ddb() and the cumulative distribution function pdb() are evaluated. Normally these would be integer values between nbot and ntop, but they need not be. Note that nbot is 0 if zeta is TRUE, and is 1 if zeta is FALSE. A result of 0 is returned by ddb() for values of x that do not satisfy the foregoing criterion. A warning is issued by ddb() if any of the values in x are non-integer. See section **Note** for a little more information. Missing values (NA) are allowed; the corresponding results are NA.
alpha	Positive scalar. The first "shape" parameter of the db distribution.
beta	Positive scalar. The second "shape" parameter of the db distribution.
ntop	Integer scalar, strictly greater than 1. The maximum possible value of the db distribution.
zeta	Logical scalar. Should zero origin indexing be used? I.e. should the range of values of the distribution be taken to be {0,1,2,...,ntop} rather than {1,2,...,ntop}? Setting zeta=TRUE may be useful for example when the values of the distribution are to be interpreted as counts.
log	Logical scalar. Should logs of the probabilities calculated by ddb() be returned, rather than the actual probabilities?
p	Vector of probablilities (i.e. values between 0 and 1). The corresponding quantiles of the db distribution are calculated by qdb(). Missing values (NA) are allowed.
n	Integer scalar. An independent sample of size n from the db distribution is generated by rdb().

Details

In the predecessor of this package (hse versions 0.1-15 and earlier), the probability function of the distribution was calculated as dbeta(x/(ntop+1),alpha,beta)/ sum(dbeta((nbot:ntop)/(ntop+k),alpha,beta)) where nbot and k were set to 1 if zeta was FALSE, and nbot was set to 0 and k to 2 if zeta was TRUE.

However the probability function is calculated in a more "direct" manner, using an exponential family representation of this function. The Beta distribution is no longer called upon (although it still of course conceptually underlies the distribution).

The function ddb() is a probability mass function for an ad hoc finite discrete distribution of *ordered* values, with a "reasonably flexible" shape.

The pth quantile of a random variable X is defined to be the infimum *over the range of* X of those values of x such that $F(x) \geq p$ where $F(x)$ is the cumulative distribution function for X. Note that if we did not impose the "over the range of X" restriction, then the 0th quantile of e.g. an exponential distribution would be $-\infty$ (since $F(x) \geq 0$ for *all* x) whereas we actually want this quantile to be 0.

Consequently qdb(p,alpha,beta,ntop) is equal to the least value of i such that pdb(i,alpha,beta,ntop) $\geq$ p. The set of values of i to be considered is {1,2,...,ntop} if zeta is FALSE and is {0,1,2,...,ntop} if zeta is TRUE.

Value

- For ddb() and pdb() vectors of probabilities.
- For qdb() a vector of quantiles.
- For rdb() a vector of length n, of integers between nbot and ntop, independently sampled from the db distribution, where nbot is 1 if zeta is FALSE and is 0 if zeta is TRUE.

Note

In the predecessor of this package (hse, versions 0.1-14 and earlier) the density/probability function threw an error if any values of argument i were not in the set of integers nbot:ntop. In accordance with a suggestion from Duncan Murdoch this behaviour was changed so that the density/probability function returns 0 for such values. It also issues a warning if any of the values are non-integer. The criterion used for "non-integer" is that abs(i-round(i)) > sqrt(.Machine$double.eps). The new behaviour is analogous to that of other probability functions used in R, dbinom() in particular.

Author(s)

Rolf Turner <r.turner@auckland.ac.nz>

See Also

meDb() mleDb()

Examples

```
parz <- list(c(0.5,0.5),c(5,1),c(1,3),c(2,2),c(2,5))
for(i in 1:5) {
    p1 <- ddb(1:15,parz[[i]][1],parz[[i]][2],15)
    names(p1) <- 1:15
    eckslab <- paste0("alpha=",parz[[i]][1]," beta=",parz[[i]][2])
    barplot(p1,xlab=eckslab,main="db probabilities",
            space=1.5,col="black")
    abline(h=0)
    if(i < 5) readline("Go? ")
}
x <- c(-1.5,-1,-0.5,0,0.5,1,1.5)
ddb(x,2.5,1,5,TRUE) # Produces 0 for all but the 4th and 6th
                    # entries of x, and issues a warning.
```

---

eow *Set or query the value of the "maxitErrorOrWarn" option.*

Description

Chooses (set.eow()) or queries (get.eow()), the reaction to maxit being exceeded in mleDb() or mleBb(). The possible reactions are to throw an error or to issue a warning. The choice is effected by calling set.eow() which sets the value of options()[["maxitErrorOrWarning"]]. The current choice is revealed by get.eow(). This choice is set equal to "error" at startup.

Usage

```
set.eow(eow = c("error", "warn"))
get.eow()
```

Arguments

eow	Character string that specifies the reaction to maxit being exceeded in mleDb() or mleBb(). May be abbreviated.

Value

No value is returned by set.eow(). the value of "maxitErrorOrWarn" in options(). The function get.eow() returns the current value of options[["maxitErrorOrWarn"]].

Note

It seems unlikely that you would want to change the option from the value that is set at startup. This function is provided "just in case".

Author(s)

Rolf Turner <r.turner@auckland.ac.nz>

See Also

mleDb() mleBb() options()

Examples

```
get.eow()    # Is "error" at startup.
set.eow("w") # Changes the option from "error" to "warning".
set.eow("e") # Changes it back again.
```

---

exactMeDb *Exact moment estimates for the db distribution.*

Description

Attempts to calculate "exact" moment estimates of the parameters of a db distribution. This is done by minimising the sum of squared differences between the sample mean and variance (xbar and s2) and the theoretical mean and variance. Calls upon optim() with the "BFGS" method.

Usage

```
exactMeDb(x, ntop, zeta=FALSE, par0 = NULL, maxit = 1000)
```

Arguments

x	A random sample from the db distribution whose parameters are being estimated. Missing values are *allowed*.
ntop	The ntop parameter of the db distribution whose parameters are being estimated. I.e. it is the maximum possible value of the distribution, whose values are integers between 1 and ntop, or between 0 and ntop if zeta (see below) is TRUE.
zeta	See ddb().
par0	Optional starting values for the iterative estimation procedure. A vector with entries alpha and beta. Ideally this vector should be named; if not it is *assumed* that the entries are in the order alpha, beta. If not supplied starting values are calculated using the undocumented function meDb().
maxit	Integer scalar. The maximum number of iterations to be undertaken by optim(). What happens if this number is exceeded depends on the value of options()[["maxitErrorOrWarning"] This may be "error" (in which case an error is thrown if maxit is exceeded) or "warning" (in which case a warning is issued). The values is set equal to "error" at startup. It may be switched, from on possibility to the other, by means of the function set.eow().

Details

This function is really an "intellectual curiosity". The results produced may be compared with those produced via maximum likelihood (using mleDb()) which in theory should be "better". Since numerical optimisation has to be applied to calculate the "exact" moment estimates, there is no real saving in terms of computation cost.

Value

An object of class "exactMeDb". Such an object consists of a named vector with entries "alpha" and "beta", which are the "exact" moment estimates of the corresponding parameters. It has a number of attributes:

- "ntop" The value of the ntop argument.
- "zeta" The value of the zeta argument.
- "minSqDiff" The (minimised) value of the sum of the squared differences between the sample mean and variance (xbar and s2) and the theoretical mean and variance. Ideally this minimised value should be zero.
- ndata The number of *non-missing* values in the data set for which the likelihood was maximised, i.e. sum(!is.na(x)).

Author(s)

Rolf Turner <r.turner@auckland.ac.nz>

See Also

ddb meDb() mleDb() expValDb() varDb() optim()

Examples

```
set.seed(42)
x <- rdb(500,3,5,2)
eMom <- exactMeDb(x,ntop=2,zeta=FALSE)
eMle <- mleDb(x,ntop=2)

# Get much better results using true parameter values
# as starting values; pity we can't do this in real life!
eMom <- exactMeDb(x,ntop=2,zeta=FALSE,par0=c(alpha=3,beta=5))
eMle <- mleDb(x,2,par0=c(alpha=3,beta=5))

# Larger ntop value
x <- rdb(500,3,5,20)
eMom <- exactMeDb(x,ntop=20,zeta=FALSE)
eMle <- mleDb(x,ntop=20)

# Binomial, n = 10, p = 0.3.
set.seed(42)
x    <- rbinom(1000,10,0.3)
eMom <- exactMeDb(x,ntop=10,zeta=TRUE)
eMle <- mleDb(x,ntop=10,zeta=TRUE)
p1   <- dbinom(0:10,10,0.3)
p2   <- dbinom(0:10,10,mean(x)/10)
p3   <- table(factor(x,levels=0:10))/1000
p4   <- ddb(0:10,alpha=eMom["alpha"],beta=eMom["beta"],ntop=10,zeta=TRUE)
plot(eMle,obsd=x,legPos=NULL,ylim=c(0,max(p1,p2,p3,p4)))
lines(0.2+(0:10),p1,col="orange",type="h",ylim=c(0,max(p1,p2)))
lines(0.3+(0:10),p2,col="green",type="h")
legend("topright",lty=1,col=c("red","blue","orange","green","black"),
       legend=c("dbMle","observed","true binomial","fitted binomial","dbMom"),bty="n")
```

`expValBb` *Expected value of a beta binomial distribution.*

Description

Calculate the expected value (theoretical mean) of a random variable having a beta binomial distribution.

Usage

```
expValBb(mo,...)
## S3 method for class 'mleBb'
expValBb(mo,...)
## Default S3 method:
expValBb(mo, size, ...)
```

Arguments

mo	For the "mleBb" method this argument is an object of class "mleBb" as returned by mleBb(). For the default method it is a numeric scalar, between 0 and 1, playing the role of m (which may be interpreted as the "success" probability. (See the help for dbetabinom().)
size	Integer scalar specifying the upper limit of the "support" of the beta binomial distribution under consideration. The support is the set of integers {0, 1, ..., size}. (See the help for dbetabinom().)
...	Not used.

Details

For the "mleBb" method, the single argument should really be called (something like) "object" and for the default method the first argument should be called m. However the argument lists must satisfy the restrictions that "*A method must have all the arguments of the generic, including ... if the generic does.*" and "*A method must have arguments in exactly the same order as the generic.*"

For the "mleBb" method, the values of m and size are extracted from the attributes of mo.

The expected value of a beta binomial distribution is trivial to calculate "by hand". These functions are provided for convenience and to preserve parallelism with the db distribution.

Value

Numeric scalar equal to the expected value of a beta binomial distributed random variable with the given parameters.

Author(s)

Rolf Turner <r.turner@auckland.ac.nz>

See Also

expValDb() varDb() varBb()

Examples

```
expValBb(0.3,15)
X   <- hmm.discnp::Downloads
fit <- mleBb(X,size=15)
expValBb(fit)
```

expValDb *Expected value of a db distribution.*

Description

Calculate the expected value (theoretical mean) of a random variable having a db distribution.

Usage

```
expValDb(ao,...)
## S3 method for class 'mleDb'
expValDb(ao,...)
## Default S3 method:
expValDb(ao, beta, ntop, zeta=FALSE,...)
```

Arguments

ao	For the "mleDb" method this argument is an object of class "mleDb" as returned by mleDb(). For the default method it is a numeric scalar playing the role of alpha (see ddb()).
beta	See ddb().
ntop	See ddb().
zeta	See ddb().
...	Not used.

Details

For the "mleDb" method, the single argument should really be called (something like) "object" and for the default method the first argument should be called alpha. However the argument lists must satisfy the restrictions that "*A method must have all the arguments of the generic, including ... if the generic does.*" and "*A method must have arguments in exactly the same order as the generic.*"

For the "mleDb" method, the values of alpha, beta, ntop and zeta (passed to ddb()) are extracted from the attributes of ao.

The expected value of a db distribution is theoretically intractable but is readily calculable numerically as

$$\sum x \times \Pr(X = x)$$

.

Value

Numeric scalar equal to the expected value of a db distributed random variable with the given parameters.

Author(s)

Rolf Turner <r.turner@auckland.ac.nz>

See Also

ddb() varDb()

Examples

```
expValDb(3,4,15)
X   <- hmm.discnp::Downloads
fit <- mleDb(X,ntop=15,zeta=TRUE)
expValDb(fit)
```

---

finfo *Fisher information.*

Description

Compute the Fisher information for a db distribution or a beta binomial distribution given the parameters of that distribution. In the case of the db distribution a specified number of observations must be supplied. In the case of the beta binomial distribution the actual observations must be supplied. The inverse of the Fisher information is an estimate of the covariance matrix of the parameter estimates.

Usage

```
finfo(distr=c("db","betabinom"),alpha, beta, ntop, ndata,
      zeta = FALSE, x, m, s, size)
```

Arguments

distr	Text string specifying which distribution to consisder. May be abbreviated (e.g. to "d" or "b").
alpha	See ddb(). Ignored if distr is "betabinom".
beta	See ddb(). Ignored if distr is "betabinom".
ntop	See ddb(). Ignored if distr is "betabinom".
ndata	The number of observations for which the Fisher information is being determined. Ignored if distr is "betabinom"; must be supplied if distr is "db".
zeta	See ddb(). Ignored if distr is "betabinom".
x	A numeric vector of observations appropriate for the model under consideration. Ignored if distr is "db"; the Fisher information for the db distribution depends only upon the parameters and not upon the data. Must be supplied if distr is "betabinom".
m	A numeric scalar, between 0 and 1, which may be interpreted as the "success" probability. (See the help for dbetabinom().) Ignored if distr is "db".
s	Numeric scalar, greater than 0. The overdispersion parameter of the distribution. (See the help for dbetabinom().) Ignored if distr is "db".
size	Integer scalar specifying the upper limit of the "support" of the betabinom distribution under consideration. The support is the set of integers {0, 1, ..., size}. (See the help for dbetabinom().) Ignored if distr is "db".

Details

This function differs from aHess() in that its arguments are prescribed "individually" rather than being extracted from an "mleDb" or "mleBb" object. This allows finfo() to be applied to "true" parameters (where these are known) rather than estimated ones.

Note that if distr is "db", the number of observations must be supplied explicitly, whereas for aHess() this number is extracted from the object argument. If distr is "betabinom" then a vector of actual observations must be supplied.

If distr is "db" then finfo() in effect calculates the *expected* information, since the information matrix does not depend on the parameters. This is not the case if distr is "betabinom". If the parameters supplied are the maximum likelihood estimates based on the supplied vector of observations x, then the value returned by finfoBb() is the *observed* Fisher information.

Value

A two-by-two positive definite (with any luck!) numeric matrix. Its inverse is an estimate of the covariance matrix of the parameter estimates.

Author(s)

Rolf Turner <r.turner@auckland.ac.nz>

See Also

link{aHess}() link{nHess}() link{mleDb}() link{mleBb}()

Examples

```
print(finfo(alpha=0.6,beta=0.3,ntop=5,ndat=54))
X   <- hmm.discnp::SydColDisc
X$y <- as.numeric(X$y)
X   <- split(X,f=with(X,interaction(locn,depth)))
x   <- X[[19]]$y
fit <- mleDb(x, ntop=5)
alpha <- fit["alpha"]
beta  <- fit["beta"]
ntop  <- attr(fit,"ntop")
zeta  <- attr(fit,"zeta")
ndat  <- ndata(fit)
print(finfo(alpha=alpha,beta=beta,ntop=ntop,ndat=ntop,zeta=zeta))
print(aHess(fit)) # Same
X     <- hrsRcePred
top1e <- X[X$sbjType=="Expert","top1"]
fit   <- mleBb(top1e,size=10)
print(finfo(distr="b",x=top1e,m=fit["m"],s=fit["s"],
            size=10)) # Observed Fisher info.
print(aHess(fit,x=top1e)) # Same
```

gof *Goodness of fit test for db and beta binomial distributions.*

Description

Either a chi-squared or a Monte Carlo test of goodness of fit of a db distribution.

Usage

```
gof(object, obsd, ...)
## S3 method for class 'mleDb'
gof(object,obsd,...,test=TRUE,MC=FALSE,seed=NULL,
                    nsim=99,maxit=1000,verb=FALSE)
## S3 method for class 'mleBb'
gof(object,obsd,...,test=TRUE,MC=FALSE,seed=NULL,
                    nsim=99,maxit=1000,verb=FALSE)
```

Arguments

object	An object of class "mleDb" or "mleBb" as returned by the function mleDb() or by mleBb().
obsd	The data to which object was fitted.
...	Not used.
test	Logical scalar. Should a hypothesis test be carried out? If test is FALSE then only the test statistic is returned. This argument is present so as to facilitate the calculations used in effecting a Monte Carlo test, by allowing gof() to recursively call itself.
MC	Logical scalar. Should a Monte Carlo test be used rather than a chi squared test?
seed	Integer scalar. The seed for the random number generator used when MC is TRUE. If not supplied, seed is created by sampling one integer from 1:1e5. This argument is ignored if MC is FALSE.
nsim	The number of simulated replicates on which the Monte Carlo test is to be based. Ignored if MC is FALSE.
maxit	Integer scalar. The maximum number of iterations to be undertaken by optim() when fitting models to the simulated data. Ignored if MC is FALSE.
verb	Logical scalar. Should rudimentary "progress reports" be issued during the course of the simulations invoked by the Monte Carlo test procedure? Ignored if MC is FALSE.

Details

The function gof() is a generic function with two methods, gof.mleDb() and gof.mleBb().

The test statistic is calculated as

$$\sum((O-E)^2/E)$$

where O means "observed" and E means "expected". If the mean of E is less than 5 or if any of the entries of E is less than 1, then the chi squared test is invalid and a warning to this effect is issued. In this case the expected values are returned as an attribute of the value returned by `gof()`. The foregoing applies of course only if a chi squared test (as opposed to a Monte Carlo test) is being used.

The degrees of freedom for the chi squared test are `length(E) - 3`. The value 3 is equal to 2 (for the number of parameters estimated) plus 1 (for the costraint that the probabilities of the values sum to 1).

If it were actually true that, under the null hypothesis, the observed test statistic and those calculated from simulated data are *exchangeable*, the Monte Carlo test would be *exact*. However the real data are distributed as $f(x, \theta)$ whereas the simulated data are distributed as $f(x, \hat{\theta})$ where $\hat{\theta}$ is the estimate of θ based on the observed data. Consequently the observed test statistic and simulated test statistics are "not quite" exchangeable. Nevertheless it appears that in practice the Monte Carlo test is very close to being exact.

The meaning of "exact" here is that if the null hypothesis is true then, over the set of instances of collecting the data **and** simulating the required replicates, the p-value is uniformly distributed on the set $\{1/N, 2/N, \ldots, (N-1)/N, 1\}$ where N is equal to `nsim`.

Value

A list with components

`stat`	The test statistic.
`pval`	The p-value of the test.
`degFree`	The degrees of freedom of the chi squared test.

The last component is present only if a chi squared test (rather than a Monte Carlo test) is used.

If a chi squared test is used and turns out to be invalid, then the returned value has an attribute `"expVals"`, consisting of the (problematic) expected values.

If a Monte Carlo test is used the returned value has an attribute `"seed"` which is equal to the `seed` argument or to the random value selected to replace it if the `seed` argument was not supplied.

Notes

The Monte Carlo p-value is calculated as `(m+1)/(nsim+1)` where `m` is the number of simulated statistics which greater than or equal to the observed statistic (computed from the "real" data.

The *smallest* that the Monte Carlo p-value can be is `1/(nsim + 1)`, e.g. 0.01 when `nsim` is 99. For "finer distinctions" you must use larger values of `nsim`, such as 999 or 9999.

The p-value is *random*; if you repeat the test (with the same data) you may well get a different p-value. Resist the temptation to repeat the test until you get a p-value that you like!!! This invalidates your inference!

Remark on the Examples

In the **Examples**, db and beta binomial distributions are fitted to the *Parsonnet scores* from the `cardiacsurgery` data set which comes from the `spcadjust` package. It is not completely clear what the value of `ntop` (db distribution) or `size` (beta binomial distribution) should be. The data are

not actually counts, and in particular they are not counts of successes out of a given number ("size") of trials. In the event I chose to use the value 71, the maximium value of the Parsonnet scores, for the value of both ntop and size. This was the value chosen for use as size by Wittenberg (2021) when he fitted a beta binomial distribution to these data.

Author(s)

Rolf Turner <r.turner@auckland.ac.nz>

References

Philipp Wittenberg (2021). Modeling the patient mix for risk-adjusted CUSUM charts. To appear in *Statistical Methods in Medical Research*.

Axel Gandy and Jan Terje Kvaloy (2013). Guaranteed conditional performance of control charts via bootstrap methods. *Scandinavian Journal of Statistics* **40**, pp. 647–668. (Reference for spcadjust package.)

See Also

mleDb()

Examples

```
X <- hmm.discnp::Downloads
f <- mleDb(X,15,TRUE)
tst1 <- gof(f,X) # Gives warning that the chi squared test is invalid.
tst2 <- gof(f,X,MC=TRUE,seed=42)
# The p-value is 0.03 so we reject the adequacy of the fit at the 0.05
# significance level.  Note that the p-value that we get, when the
# random number generator seed is set equal to 42, is very similar in
# value to the p-value (0.0347) from the "invalid" chi squared test.
#
## Not run:  # Takes too long.
if(requireNamespace("spcadjust")) {
    data("cardiacsurgery", package = "spcadjust")
    xxx  <- cardiacsurgery$Parsonnet
    fit1 <- mleDb(xxx,ntop=71,zeta=TRUE)
    g1   <- gof(fit1,obsd=xxx,MC=TRUE,verb=TRUE,seed=42)
    fit2 <- mleBb(xxx,size=71)
    g2   <- gof(fit2,obsd=xxx,MC=TRUE,verb=TRUE,seed=17)
}

## End(Not run)
```

`hrsRcePred` *Horse race prediction data.*

Description

Counts of correct predictions of the outcomes of 10 harness races made by "experts" and "non-experts".

Usage

```
hrsRcePred
```

Format

A data frame with 30 observations on the following 4 variables.

`sbjType` A character vector with entries `"NonXpert"` and `"Expert"`, which classifies the "subjects" (the people making the predictions of the race outcomes).

`subject` An integer vector indexing the subjects. (Not of any real consequence.)

`top1` An integer vector giving the counts of correct predictions of the winners of 10 harness races.

`top3` An integer vector giving the counts of correct predictions of the top three horses ("win/place/show" in 10 harness races.

Details

In Ceci and Liker (1986) it is stated that subjects were classified as "experts" and "nonexperts" based on their ability to predict post-time odds on the basis of factual information about horses.

It appears that the counts in `top1` and `top3` pertain to the *same* 10 races, but this is not completely clear.

Source

These data are taken from the paper cited in the first of the two given in the **References** below. They were provided by a generous email correspondent who prefers to remain anonymous.

References

Ceci, S. J. and Liker, J. K. (1986). A day at the races: A study of IQ, expertise, and cognitive complexity. *Journal of Experimental Psychology, General* **115**, pp. 255 – 266.

Ceci, S. J. and Liker, J. K. (1988). Stalking the IQ-expertise relation: When the critics go fishing. *Journal of Experimental Psychology, General* **117**, pp. 96 – 100.

Examples

```
      X     <- hrsRcePred
      top1e <- X[X$sbjType=="Expert","top1"]
      top1n <- X[X$sbjType=="NonXpert","top1"]
      top3e <- X[X$sbjType=="Expert","top3"]
      top3n <- X[X$sbjType=="NonXpert","top3"]
      dbfit1e <- mleDb(top1e,ntop=10,zeta=TRUE)
      dbfit1n <- mleDb(top1n,ntop=10,zeta=TRUE)
      dbfit3e <- mleDb(top3e,ntop=10,zeta=TRUE)
      dbfit3n <- mleDb(top3n,ntop=10,zeta=TRUE)
  # Set seeds to get repeatable Monte Carlo p-values.
      ## Not run:  # Takes too long.
          print(gof(dbfit1e,obsd=top1e,MC=TRUE,maxit=5000,verb=TRUE,seed=49)$pval) # 0.02
          print(gof(dbfit1n,obsd=top1n,MC=TRUE,verb=TRUE,seed=128)$pval)           # 0.79
          print(gof(dbfit3e,obsd=top3e,MC=TRUE,verb=TRUE,seed=303)$pval)           # 0.35
          print(gof(dbfit3n,obsd=top3n,MC=TRUE,maxit=3000,verb=TRUE,seed=24)$pval) # 0.40

  ## End(Not run)
      bbfit1e <- mleBb(top1e,size=10)
      bbfit1n <- mleBb(top1n,size=10)
      bbfit3e <- mleBb(top3e,size=10)
      bbfit3n <- mleBb(top3n,size=10)
  # Set seeds to get repeatable Monte Carlo p-values.
      ## Not run:  # Takes too long.
          print(gof(bbfit1e,obsd=top1e,MC=TRUE,verb=TRUE,seed=792)$pval) # 0.11
          print(gof(bbfit1n,obsd=top1n,MC=TRUE,verb=TRUE,seed=48)$pval)  # 0.64
          print(gof(bbfit3e,obsd=top3e,MC=TRUE,verb=TRUE,seed=969)$pval) # 0.62
          print(gof(bbfit3n,obsd=top3n,MC=TRUE,verb=TRUE,seed=834)$pval) # 0.75

  ## End(Not run)
  # Reality check: goodness of fit tests for the fit of just plain *binomial*
  # distributions to these data sets yielded Monte Carlo p-values equal to
  # 0.22, 0.17, 0.32 and 0.73 respectively.  I.e. binomial fits appear to
  # work just fine!
```

`llPlot` *Plot the log likelihood surface for the data.*

Description

Plot, as a perspective plot or a contour plot, the log likelihood surface for the data set from which parameters are being estimated.

Usage

```
llPlot(x, distr=c("db","betabinom"),ntop, zeta, size, alim = NULL, blim = NULL,
       ngrid = c(100, 100), plotType = c("persp", "contour", "none"),
       theta = -30, phi = 40, ...)
```

Arguments

x	A vector of numeric data purportedly arising from a db or beta binomial distribution.
distr	Character string specifying which of the two relevant distributions (db, or beta binomial) is to be considered.
ntop	See mleDb() and ddb(). Ignored if distr is "betabinom".
zeta	See mleDb() and ddb(). Ignored if distr is "betabinom".
size	Integer scalar specifying the upper limit of the "support" of the beta binomial distribution under consideration. The support is the set of integers {0, 1, ..., size}. (The values of x may sometimes be considered to be the number of "successes" in size trials. The size argument is ignored if distr is "db".
alim	Numeric vector of length 2; the range of alpha values over which the surface is to be plotted. Defaults to c(0,10) if distr is "db" and to c(0,1) if distr is "betabinom".
blim	Numeric vector of length 2; the range of beta values over which the surface is to be plotted. Defaults to c(0,10) if distr is "db" and to c(0,100) if distr is "betabinom".
ngrid	The dimensions of the grid of paramter values at which the log likelihood is to be evaluated in order to plot the surface. Note that ngrid may be supplied as an integer scalar, in which case it is replicated to a vector of length 2.
plotType	Character string specifying the nature of the plot to be produced. If it is "none" then no plot is produced. The value returned may be plotted at a later occasion.
theta	An argument to be passed to persp(). Ignored unless plotType is "persp".
phi	An argument to be passed to persp(). Ignored unless plotType is "persp".
...	Other arguments that may be passed to persp() or to contour()

Details

This function could conceivably be useful in diagnosing problems with parameter estimation should these arise.

Value

A list with entries

x	The vector of values of the first parameter (alpha for distr="db", m for distr="betabinom") over which the surface is to be plotted. There are ngrid[1] such values, ranging from alim[1] to alim[2].
y	The vector of values of the second parameter (beta for distr="db", s for distr="betabinom") over which the surface is to be plotted. There are ngrid[2] such values, ranging from blim[1] to blim[2].
z	An ngrid[1] x ngrid[2] numeric matrix, specifying the surface. the value of z[i,j] is ll(x[i],y[j]) where ll() is the log likelihood function.

dxy — A data frame with columns named "alpha" and "beta" for distr="db" or "m" and "s" for distr="betabinom", and ngrid[1]*ngrid[2] rows. It is formed by applying expand.grid() to the x and y entries of this list.

fxy — A numeric vector of length ngrid[1]*ngrid[2]. Its ith value is the log likelihood evaluated at the ith row of dxy. Its entries are the same as the entries of z.

There is obviously considerable redundancy in the returned value.

The names x and y that are used for the first two entries of this list conform to the names of the arguments of persp() and contour.

If plotType is "persp" or "contour" the value is returned invisibly.

Author(s)

Rolf Turner <r.turner@auckland.ac.nz>

See Also

link{mleDb}() link{mleBb}() link{persp}() link{contour}()

Examples

```
X <- hmm.discnp::SydColDisc
X$y <- as.numeric(X$y)
X <- split(X,f=with(X,interaction(locn,depth)))
x <- X[[19]]$y
srf <- llPlot(x,ntop=5,zeta=FALSE,alim=c(0.5,0.7),blim=c(0.2,0.4),plotType="c")
## Not run:
if(require(rgl)) {
   with(srf,plot3d(ab$alpha,ab$beta,fab)
# Allows dynamic rotation of the surface.
}

## End(Not run)
# Negative (!) parameters for the db distribution.
set.seed(42)
xs <- rdb(100,-1,-1,5)
fit <- mleDb(xs,5)
llPlot(xs,ntop=5,zeta=FALSE,alim=c(-4,2),blim=c(-4,2),plotType="c",
       main="log likelihood contours")
points(fit[1],fit[2],pch=20,col="red")
points(-1,-1,pch=20,col="blue")
legend("topright",pch=20,col=c("red","blue"),
       legend=c("estimate","true value"),bty="n")
```

logLik *Retrieve the (maximised) log likelihood from an* "mleDb" *or an* "mleBb" *object.*

Description

Extract the log likelihood attribute an object of class "mleDb" or "mleBb". I.e. obtain the maximum log likelihood in respect of the estimation of the parameters of a db or beta-binomial distribution.

Usage

```
   ## S3 method for class 'mleDb'
logLik(object, ...)
   ## S3 method for class 'mleBb'
logLik(object, ...)
```

Arguments

object	An object of class "mleDb" as returned by mleDb() or of class "mleBb" as returned by mleBb().
...	Not used.

Value

An object of class "logLik", which consists of a numeric scalar equal to the maximum log likelihood for the parameters of a db or beta-binomial distribution. It has an attribute "df" equal to 2.

Author(s)

Rolf Turner <r.turner@auckland.ac.nz>

See Also

mleDb() mleBb()

Examples

```
X <- hmm.discnp::SydColDisc
X$y <- as.numeric(X$y)
X <- split(X,f=with(X,interaction(locn,depth)))
fitz <- lapply(X,function(x){mleDb(x$y,ntop=5)})
sapply(fitz,logLik)
X     <- hrsRcePred
top1e <- X[X$sbjType=="Expert","top1"]
fit   <- mleBb(top1e,10)
logLik(fit)
```

makeBbdpars *Create an object of class* "Bbdpars".

Description

Create an object of class "Bbdpars" which may be used as an argument of the simulate() function.

Usage

```
makeBbdpars(m, s, size, ndata)
```

Arguments

m	Numeric scalar between 0 and 1. May be interpreted as a "success probability".
s	Numeric scalar, greater than 0. The overdispersion parameter of the beta binomial distribution. Note that if overdispersion is defined to equal the ratio of the variance of the data to the corresponding "binomial variance" (i.e. the actual variance over m*(1-m)*size) the overdispersion tends to 0 as s tends to infinity and to size as s tends to 0.
size	Integer scalar specifying the upper limit of the "support" of the beta binomial distribution under consideration. The support is the set of integers {0, 1, ..., size}.
ndata	Integer vector specifying the lengths of the data sets to be simulated. If it is of length less than the nsim argument of simulate() (e.g. if it is a scalar) then it is "recycled" to provide a vector of length nsim. If is longer than nsim, then only the first nsim entries are used and the others are ignored. If the argument ndata of the simulate() function is supplied then the ndata component specified here is ignored by simulate().

Value

An object of class "Bbdpars" which is a list with components m, s, size and ndata. The entries of this list are simply the corresponding function arguments.

Author(s)

Rolf Turner <r.turner@auckland.ac.nz>

See Also

simulate.Bbdpars()

Examples

```
obj1  <- makeBbdpars(m=0.35,s=0.3,size=20,ndata=500)
obj2  <- makeBbdpars(m=0.85,s=1.7,size=20,ndata=30*(1:10))
## Not run:
    sdat1 <- simulate(obj1,nsim=100)
    sdat2 <- simulate(obj2,nsim=100)

## End(Not run)
obj2 is ignored.
sdat3 <- simulate(obj2,nsim=10)
## Not run:
    sdat4 <- simulate(obj2,nsim=100,ndata=100*(2:6)) # The ndata component of

## End(Not run)
```

---

makeDbdpars *Create an object of class* "Dbdpars".

Description

Create an object of class "Dbdpars" which may be used as an argument of the simulate() function.

Usage

```
makeDbdpars(alpha, beta, ntop, zeta, ndata)
```

Arguments

alpha	The first "shape" parameter of the db distribution.
beta	The second "shape" parameter of the db distribution.
ntop	Integer scalar, strictly greater than 1. The maximum possible value of the db distribution.
zeta	Logical scalar. Should zero origin indexing be used? I.e. should the range of values of the distribution be taken to be {0,1,2,...,ntop} rather than {1,2,...,ntop}? Setting zeta=TRUE may be appropriate for example when the values of the distribution are to be interpreted as counts.
ndata	Integer vector specifying the lengths of the data sets to be simulated. If it is of length less than the nsim argument of simulate() (e.g. if it is a scalar) then it is "recycled" to provide a vector of length nsim. If is longer than nsim, then only the first nsim entries are used and the others are ignored. If the argument ndata of the simulate() function is supplied then the ndata component specified here is ignored by simulate().

Value

An object of class "Dbdpars" which is a list with components alpha, beta, ntop, zeta and ndata. The entries of this list are simply the corresponding function arguments.

Author(s)

Rolf Turner <r.turner@auckland.ac.nz>

See Also

simulate.Dbdpars()

Examples

```
obj1  <- makeDbdpars(alpha=2,beta=3,ntop=20,zeta=TRUE,ndata=500)
obj2  <- makeDbdpars(alpha=0.2,beta=0.25,ntop=20,zeta=FALSE,ndata=30*(1:10))
sdat1 <- simulate(obj1,nsim=100)
sdat2 <- simulate(obj2,nsim=100)
sdat3 <- simulate(obj2,nsim=10)
sdat4 <- simulate(obj2,nsim=100,ndata=100*(2:6)) # The ndata component of
                                                 # obj2 is ignored.
```

---

mcCovMat *Monte Carlo estimation of a covariance matrix.*

Description

Calculate an estimate of the covariance matrix for the parameter estimates of a db or beta binomial distribution via simulation.

Usage

```
mcCovMat(object, nsim = 100, seed=NULL, maxit=1000)
## S3 method for class 'mleDb'
mcCovMat(object, nsim = 100, seed=NULL, maxit=1000)
## S3 method for class 'mleBb'
mcCovMat(object, nsim = 100, seed=NULL, maxit=1000)
## S3 method for class 'Dbdpars'
mcCovMat(object, nsim = 100, seed=NULL, maxit=1000)
## S3 method for class 'Bbdpars'
mcCovMat(object, nsim = 100, seed=NULL, maxit=1000)
## Default S3 method:
mcCovMat(object, nsim = 100, seed=NULL, maxit=1000)
```

Arguments

object	An object of class either "mleDb", "mleBb", Dbdpars or Bbdpars. In the first two cases such an object would be returned by the function mleDb() or by mleBb(). In the second two cases such an object would be returned by the function makeDbdpars() or by makeBbdpars().
nsim	Integer scalar. The number of simulations to be used to produce the Monte Carlo estimate of the covariance matrix.

seed	Integer scalar. The seed for the random number generator. If not specified it is randomly sampled from the sequence 1:1e5.
maxit	Integer scalar. The maximum number of iterations to be undertaken by optim() when fitting models to the simulated data.

Details

The procedure is to simulate nsim data sets, all of the same size. This will be the size of the data set to which object was fitted), in the case of the "mleDb" and "mleBb" methods, and will be the value of the ndata argument supplied to the "make" function in the case of the "Dbdpars" and "Bbdpars" methods. The simulations are from models determined by the parameter value contained in object.

From each such simulated data, parameter estimates are obtained. The covariance matrix of these latter parameter estimates (adjusted for the fact that the true parameters are known in a simulation) is taken to be the required covariance matrix estimated.

The default method simply throws an error.

Value

A two-by-two positive definite (with any luck!) numeric matrix. It is an estimate of the covariance matrix of the parameter estimates.

It has an attribute "seed" which is the seed that was used for the random number generator. This is either the value of the argument seed or (if this argument was left NULL) the value that was randomly sampled from 1:1e5.

Author(s)

Rolf Turner <r.turner@auckland.ac.nz>

See Also

link{aHess}() link{nHess}() link{vcov.mleDb}() link{vcov.mleBb}()

Examples

```
X   <- hmm.discnp::SydColDisc
X$y <- as.numeric(X$y)
X   <- split(X,f=with(X,interaction(locn,depth)))
x   <- X[[19]]$y
fit <- mleDb(x, ntop=5)
set.seed(42)
CM.m <- mcCovMat(fit,nsim=500) # Lots of simulations!
CM.a <- vcov(fit)
CM.n <- solve(nHess(fit,x))
cat("Monte Carlo:\n\n")
print(CM.m)
cat("Analytic:\n\n")
print(CM.a)
cat("Numeric:\n\n")
print(CM.n)
X     <- hrsRcePred
```

```
top1e <- X[X$sbjType=="Expert","top1"]
fit   <- mleBb(top1e,size=10)
CM.m  <- mcCovMat(fit,nsim=500) # Lots of simulations!
CM.a  <- vcov(fit)
CM.n  <- solve(nHess(fit,top1e))
cat("Monte Carlo:\n\n")
print(CM.m)
cat("Analytic:\n\n")
print(CM.a)
cat("Numeric:\n\n")
print(CM.n)
```

`mleBb` *Maximum likelihood estimation of the parameters of a beta binomial distribution.*

Description

Calculates maximum likelihood estimates of the `m` and `s` parameters of a beta binomial distribution. Calls upon `optim()` with the `"L-BFGS-B"` method.

Usage

```
mleBb(x, size, par0, maxit=1000, covmat=TRUE, useGinv=FALSE)
```

Arguments

`x`	Integer vector of counts to which a beta binomial distribution is to be fitted. Missing values are allowed. (These are discarded before the data are analysed.)
`size`	Integer scalar specifying the upper limit of the "support" of the beta binomial distribution under consideration. The support is the set of integers `{0, 1, ..., size}`. (The values of `x` may sometimes be considered to be the number of "successes" in `size` trials.
`par0`	Optional starting values for the iterative estimation procedure. A vector with entries `m` and `s`. Ideally this vector should be named; if not it is *assumed* that the entries are in the order `m`, `s`. If not supplied starting values are calculated using `meBb()`.
`maxit`	Integer scalar. The maximum number of iterations to be undertaken by `optim()`. What happens if this number is exceeded depends on the value of `options()[["maxitErrorOrWarning"]]`. This may be `"error"` (in which case an error is thrown if `maxit` is exceeded) or `"warning"` (in which case a warning is issued). The values is set equal to `"error"` at startup. It may be switched, from on possibility to the other, by means of the function `set.eow()`.
`covmat`	Logical scalar. Should the covariance matrix of the parameter estimates be calculated? In simulation studies, in which the covariance matrix is not of interest, calculations might be speeded up a bit by setting `covmat=FALSE`.

useGinv — Logical scalar. Should the ginv() (generalised inverse) function from the MASS package be used to calculate a surrogate covariance matrix if the hessian is numerically singular? This is probably not advisable; the possibility of using the generalised inverse is provided for the sake of completeness. *Caveat utilitor*. This argument is ignored if covmat is FALSE.

Details

This function is provided so as to give a convenient means of comparing the fit of a beta binomial distribution with that of the discretised Beta (db) distribution which is the focus of this package.

Value

An object of class "mleBb" which is a vector of length two. Its first entry m is the estimate of the (so-called) success probability m; its second entry s is the estimate of the overdispersion parameter s. It has a number of attributes:

- "size" The value of the size argument.
- "log.like" The (maximised) value of the log likelihood of the data.
- "covMat" An estimate of the (2×2) covariance matrix of the parameter estimates. This is formed as the inverse of the hessian (of the negative log likelihood) calculated by aHess().
- ndata The number of *non-missing* values in the data set for which the likelihood was maximised, i.e. sum(!is.na(x)).

Author(s)

Rolf Turner <r.turner@auckland.ac.nz>

References

Bruce Swihart and Jim Lindsey (2020). rmutil: Utilities for Nonlinear Regression and Repeated Measurements Models. R package version 1.1.4. https://CRAN.R-project.org/package=rmutil

Wikipedia, https://en.wikipedia.org/wiki/Beta-binomial_distribution

See Also

mleDb() optim() aHess() vcov.mleBb() hrsRcePred visRecog

Examples

```
if(require(hmm.discnp)) {
   X <- hmm.discnp::Downloads
   f <- mleBb(X,15)
}
set.seed(42)
X <- c(rbinom(20,10,0.3),rbinom(20,10,0.7))
f <- mleBb(X,10)
g <- mleDb(X,10,TRUE)
print(attr(f,"log.like"))
print(attr(g,"log.like")) # Not much difference.
```

```
dbfit5  <- with(visRecog,mleDb(tot5,20,TRUE))
print(vcov(dbfit5))
# See the help for data sets "hrsRcePred" and "visRecog" for
# other examples.
```

`mleDb` *Maximum likelihood estimates of db parameters.*

Description

Calculates maximum likelihood estimates of the `alpha` and `beta` parameters of a db distribution. Calls upon `optim()` with the `"BFGS"` method.

Usage

```
mleDb(x, ntop, zeta=FALSE, par0=NULL, UB=10, maxit=1000,
      covmat=TRUE, useGinv=FALSE)
```

Arguments

`x`	A random sample from the db distribution whose parameters are being estimated. Missing values are *allowed*.
`ntop`	The `ntop` parameter of the db distribution whose parameters are being estimated. I.e. it is the maximum possible value of the distribution, whose values are integers between 1 and `ntop`, or between 0 and `ntop` if `zeta` (see below) is `TRUE`.
`zeta`	See `ddb()`.
`par0`	Optional starting values for the iterative estimation procedure. A vector with entries `alpha` and `beta`. Ideally this vector should be named; if not it is *assumed* that the entries are in the order `alpha`, `beta`. If not supplied starting values are calculated using `meDb()`.
`UB`	Positive numeric scalar, providing an upper bound on the starting values used by `mleDb()`. It appears that if these starting values are too large (it is not clear *how* large) then `optim()` will throw an error. This bound is ignored if `par0` is supplied.
`maxit`	Integer scalar. The maximum number of iterations to be undertaken by `optim()`. What happens if this number is exceeded depends on the value of `options()[["maxitErrorOrWarning"]` This may be `"error"` (in which case an error is thrown if `maxit` is exceeded) or `"warning"` (in which case a warning is issued). The values is set equal to `"error"` at startup. It may be switched, from on possibility to the other, by means of the function `set.eow()`.
`covmat`	Logical scalar. Should the covariance matrix of the parameter estimates be calculated? In simulation studies, in which the covariance matrix is not of interest, calculations might be speeded up a bit by setting `covmat=FALSE`.
`useGinv`	Logical scalar. Should the `ginv()` (generalised inverse) function from the `MASS` package be used to calculate a surrogate covariance matrix if the hessian is numerically singular? This is probably not advisable; the possibility of using the generalised inverse is provided for the sake of completeness. *Caveat utilitor*. This argument is ignored if `covmat` is `FALSE`.

Details

The ntop and zeta parameters must be supplied; they are not formally estimated (although the choice of ntop may be influenced by the data — see below). The parameter zeta has a default value, FALSE.

If the generating mechanism from which the observed data x arose has a (known) theoretical least upper bound, then ntop should probably be set equal to this upper bound. If the data are theoretically unbounded, then ntop should probably be set equal to 1 + max(x). In this case $\Pr(X = \text{ntop})$ should probably be interpreted as $\Pr(X \geq \text{ntop})$. Otherwise ntop should should probably be set equal to max(x). The choice depends on circumstances and is up to the user.

Missing values are removed from x before it is passed to optim(). (Note that ddb() doesn't mind missing values but returns missing values when evaluated at them. This in turn produces a missing value for the log likelihood.)

In previous versions of this package (0.1-17 and earlier) optim() was called with method "L-BFGS-B". The change was made possible by the fact that, with the new "direct" version of ddb(), it is no longer necessary to bound the parameters away from (above) zero.

Value

An object of class "mleDb". Such an object consists of a named vector with entries "alpha" and "beta", which are the estimates of the corresponding parameters. It has a number of attributes:

- "ntop" The value of the ntop argument.
- "zeta" The value of the zeta argument.
- "log.like" The (maximised) value of the log likelihood of the data.
- "covMat" An estimate of the (2×2) covariance matrix of the parameter estimates. This is formed as the inverse of the hessian (of the negative log likelihood) calculated by aHess().
- ndata The number of *non-missing* values in the data set for which the likelihood was maximised, i.e. sum(!is.na(x)).

Author(s)

Rolf Turner <r.turner@auckland.ac.nz>

See Also

ddb meDb() optim() aHess() vcov.mleDb()

Examples

```
set.seed(42)
x <- rdb(500,3,5,2)
ests <- mleDb(x,2)  # Bad!  Mind you, 2 is a "bad" value for ntop!
                    # Hessian is singular; covMat is NA.
# Get much better results using true parameter values
# as starting values; pity we can't do this in real life!
ests <- mleDb(x,2,par0=c(alpha=3,beta=5))
x <- rdb(500,3,5,20)
ests <- mleDb(x,20) # Pretty good.
```

```
print(vcov(ests))

# Binomial, n = 10, p = 0.3.
set.seed(42)
x   <- rbinom(1000,10,0.3)
fit <- mleDb(x,10,zeta=TRUE)
print(vcov(fit))
p1  <- dbinom(0:10,10,0.3)
p2  <- dbinom(0:10,10,mean(x)/10)
p3  <- table(factor(x,levels=0:10))/1000
plot(fit,obsd=x,legPos=NULL,ylim=c(0,max(p1,p2,p3,
     ddb(0:10,fit[1],fit[2],10,zeta=TRUE))))
lines(0.2+(0:10),p1,col="orange",type="h",ylim=c(0,max(p1,p2)))
lines(0.3+(0:10),p2,col="green",type="h")
legend("topright",lty=1,col=c("red","blue","orange","green"),
       legend=c("db","observed","true binomial","fitted binomial"),bty="n")
print(attr(fit,"log.like")) # -1778.36
print(sum(dbinom(x,10,mean(x)/10,log=TRUE))) # -1777.36
# Slightly better fit with only one estimated parameter,
# but then binomial is the true distribution, so you'd
# kind of expect a better fit.
print(sum(dbinom(x,10,0.3,log=TRUE))) # -1778.37

# Poisson mean = 5
set.seed(42)
x    <- rpois(1000,5)
fit  <- mleDb(x,14,zeta=TRUE) # max(x) = 13, take ntop = 1+13
print(vcov(fit))
p1   <- c(dpois(0:13,5),1-ppois(13,5))
lhat <- mean(x)
p2   <- c(dpois(0:13,lhat),1-ppois(13,lhat))
plot(fit,obsd=x,legPos=NULL,ylim=c(0,max(p1,p2,p3,
     ddb(0:14,fit[1],fit[2],14,zeta=TRUE))))
lines(0.2+0:14,p1,col="orange",type="h")
lines(0.3+(0:14),p2,col="green",type="h")
legend("topright",lty=1,col=c("red","blue","orange","green"),
       legend=c("db","observed","true Poisson","fitted Poisson"),bty="n")
print(attr(fit,"log.like")) # -2198.594
print(sum(dpois(x,lhat,log=TRUE))) # -2197.345
# Slightly better fit with only one estimated parameter,
# but then Poisson is the true distribution, so you'd
# kind of expect a better fit.
print(sum(dpois(x,5,log=TRUE))) # -2198.089
```

`ndata` *Retrieve the* `"ndata"` *attribute of an* `"mleDb"` *object.*

Description

Retrieve the number of (non-missing) values in the data set to which an `"mleDb"` object was fitted.

Usage

```
ndata(object)
```

Arguments

object	An object of class "mleDb" as returned by mleDb().

Value

Integer scalar equal to the number of (non-missing) values in the data set to which object was fitted.

Author(s)

Rolf Turner <r.turner@auckland.ac.nz>

See Also

mleDb()

Examples

```
X <- hmm.discnp::SydColDisc
X$y <- as.numeric(X$y)
X <- split(X,f=with(X,interaction(locn,depth)))
fitz <- lapply(X,function(x){mleDb(x$y,ntop=5)})
sapply(fitz,ndata)
```

---

nHess *Numerical hessian calculation.*

Description

Calculate an approximation to the hessian of the **negative** log likelihood of a db or beta binomial distribution via a numerical (finite differencing based) procedure as effected by optimHess().

Usage

```
nHess(object, x, silent=TRUE)
```

Arguments

object	An object of class "mleDb" or "mleBb" as returned by the function mleDb() or mleBb().
x	Numeric vector of non-negative integer data, presumably the data set on the basis of which object was calculated. Ignored if object is of class "mleDb".
silent	Logical scalar. If the call to optimHess() throws an error, should the error message be suppressed? (A possibly less informative warning will be issued in any case.)

Details

It is up to the user to make sure that (when `object` is of class `"mleBb"`) `object` and `x` are "mutually compatible", i.e. are appropriately paired up.

Note that this function calculates the hessian of the **negative** log likelihood of the distribution in question, as *minimised* by `optim()`. Hence its inverse is an estimate of the covariance matrix of the parameter estimates. (Do *not* take the negative of this hessian before inverting it to get the desired covariance matrix!)

This function is mainly present to investigate possible differences between the numerical approximation to the hessian, which is what `optim()` uses in its maximisation procedure, and the analytic form of the hessian.

Value

A two-by-two positive definite (with any luck!) numeric matrix. Its inverse is an estimate of the covariance matrix of the parameter estimates.

Author(s)

Rolf Turner `<r.turner@auckland.ac.nz>`

See Also

`aHess()` `mleDb()` `mleBb()` `optim()` `optimHess()`

Examples

```
X <- hmm.discnp::SydColDisc
X$y <- as.numeric(X$y)
X <- split(X,f=with(X,interaction(locn,depth)))
x <- X[[19]]$y
fit <- mleDb(x, ntop=5)
H   <- nHess(fit,x)
print(solve(H)) # Compare with ...
print(vcov(fit))
```

`plot.mleBb` *Plot a maxium likelihood fit to data from a beta binomial distribution.*

Description

Creates a plot of type `"h"` of the probabilities of each possible `x` value of a beta binomial distribution where the probabilities are calculated on the basis of parameters estimated by the function `mleBb()`. If `obsd` is supplied it also superimposes/juxtaposes vertical lines representing the observed proportions.

Usage

```
## S3 method for class 'mleBb'
plot(x, ..., plot = TRUE, col.fit = "red", col.obsd = "blue",
             tikx = NULL, xlim=NULL, ylim=NULL, xlab = NULL,
             ylab = NULL, obsd = NULL, incr = NULL,
             main = "", legPos = "topright")
```

Arguments

x	An object of class "mleBb" as returned by the function mleBb()
...	Not used.
plot	Logical scalar; should a plot be produced (or should the function simply return a data frame consisting of the relevant values)?
col.fit	The colour for the (vertical) lines corresponding to the "fitted" probabilities, i.e. the probabilities calculated from the fitted parameters.
col.obsd	The colour for the (vertical) lines corresponding to the "observed" probabilities (proportions), i.e. the probabilities calculated by tabulating the data (from which the parameters were estimated. Ignored if obsd is not supplied.
tikx	(Optional) vector of locations of the tick marks on the x-axis.
xlim	A numeric vector of length 2 specifying the limits of the x-axis. Defaults to c(0,size) Note that \ codesize is extracted from argument x.
ylim	A numeric vector of length 2 specifying the limits of the y-axis. There is a "sensible" default.
xlab	A label for the x-axis; defaults to x.
ylab	A label for the y-axis; defaults to probability.
obsd	The data set from which the parameters were estimated, i.e. from which x was obtained. (Optional.)
incr	Numeric scalar; defaults to 0.1 if size (extracted from x) is less than 20 and to 0.5 otherwise. This number should be non-zero and less than 1 in absolute value. (One would usually want it to be positive, but it could conceiveably be set to a small negative value.) It gives the value of the "increment" or "shift" that separates the vertical lines representing the fitted probabilities and those representing the observed proportions (calculated from obsd). Ignored if obsd is not supplied.
main	A main title for the plot; defaults to the empty string.
legPos	A list with components x and y, or a text string, specifying the placement of the legend. See legend() for details. A legend is plotted only if obsd is specified, whence legPos is otherwise ignored. The plotting of a legend may be suppressed (even when obsd is supplied) by setting legPos=NULL.

Value

A data frame with numeric columns x, p and possibly po. The x column consists of the integers from 0 to size. The p column consists of the appropriate probabilities of the x values, calculated

by dbetabinom() from the rmutil package. The po column is present only if obsd is supplied and consists of the observed proportions. The value is returned invisibly. A plot is produced as a side-effect if plot is TRUE.

Note

This function calls plotBb() to do the heavy lifting.

Warning

It is up to the user to make sure that the obsd argument, if specified, is indeed the data set from which the object x was calculated.

Author(s)

Rolf Turner <r.turner@auckland.ac.nz>

See Also

mleBb() plotBb()

Examples

```
if(require(hmm.discnp)) {
   xxx <- hmm.discnp::Downloads
   fit <- mleBb(xxx,size=14)
   plot(fit)
   plot(fit,obsd=xxx)
   plot(fit,obsd=xxx,legPos=list(x=3,y=0.25))
   plot(fit,obsd=xxx,legPos=NULL) # No legend is plotted.
}
set.seed(42)
yyy <- rbinom(300,10,0.7)
fit <- mleBb(yyy,size=10)
plot(fit,obsd=yyy,legPos="topleft")
```

---

plot.mleDb *Plot a maxium likelihood fit to data from a db distribution.*

Description

Creates a plot of type "h" of the probabilities of each possible x value of a db distribution where the probabilities are calculated on the basis of parameters estimated by the function mleDb(). If obsd is supplied it also superimposes/juxtaposes vertical lines representing the observed proportions.

Usage

```
## S3 method for class 'mleDb'
plot(x, ..., plot = TRUE, col.fit = "red", col.obsd = "blue",
             tikx=NULL, xlim=NULL, ylim=NULL, xlab = NULL, ylab = NULL,
             obsd = NULL, incr = NULL, main = "", legPos = "topright")
```

Arguments

x	An object of class "mleDb" as returned by the function mleDb()
...	Not used.
plot	Logical scalar; should a plot be produced (or should the function simply return a data frame consisting of the relevant values)?
col.fit	The colour for the (vertical) lines corresponding to the "fitted" probabilities, i.e. the probabilities calculated from the fitted parameters.
col.obsd	The colour for the (vertical) lines corresponding to the "observed" probabilities (proportions), i.e. the probabilities calculated by tabulating the data (from which the parameters were estimated. Ignored if obsd is not supplied.
tikx	(Optional) vector of locations of the tick marks on the x-axis.
xlim	A numeric vector of length 2 specifying the limits of the x-axis. Defaults to c(nbot,ntop) where nbot is 0 if x[["zeta"]] is TRUE (i.e. zero origin indexing is uses) and is 1 otherwise. Note that ntop and zeta are extracted from argument x.
ylim	A numeric vector of length 2 specifying the limits of the y-axis. There is a "sensible" default.
xlab	A label for the x-axis; defaults to x.
ylab	A label for the y-axis; defaults to probability.
obsd	The data set from which the parameters were estimated, i.e. from which x was obtained. (Optional.)
incr	Numeric scalar; defaults to 0.1 if ntop (extracted from x) is less than 20 and to 0.5 otherwise. This number should be non-zero and less than 1 in absolute value. (One would usually want it to be positive, but it could conceiveably be set to a small negative value.) It gives the value of the "increment" or "shift" that separates the vertical lines representing the fitted probabilities and those representing the observed proportions (calculated from obsd). Ignored if obsd is not supplied.
main	A main title for the plot; defaults to the empty string.
legPos	A list with components x and y, or a text string, specifying the placement of the legend. See legend() for details. A legend is plotted only if obsd is specified, whence legPos is otherwise ignored. The plotting of a legend may be suppressed (even when obsd is supplied) by setting legPos=NULL.

Value

A data frame with numeric columns x, p and possibly po. The x column consists of the integers from 0 to ntop or from 1 to ntop depending on whether zeta is TRUE. The p column consists of the appropriate probabilities of the x values, calculated by link{ddb}(). The po column is present only if obsd is supplied and consists of the observed proportions. The value is returned invisibly. A plot is produced as a side-effect if plot is TRUE.

Note

This function calls plotDb() to do the heavy lifting.

Warning

It is up to the user to make sure that the obsd argument, if specified, is indeed the data set from which the object x was calculated.

Author(s)

Rolf Turner <r.turner@auckland.ac.nz>

See Also

mleDb() plotDb() ddb()

Examples

```
if(require(hmm.discnp)) {
   xxx <- hmm.discnp::Downloads
   fit <- mleDb(xxx,ntop=14,z=TRUE)
   plot(fit)
   plot(fit,obsd=xxx)
   plot(fit,obsd=xxx,legPos=list(x=3,y=0.25))
   plot(fit,obsd=xxx,legPos=NULL) # No legend is plotted.
}
set.seed(42)
yyy <- rbinom(300,10,0.7)
fit <- mleDb(yyy,ntop=10,z=TRUE)
plot(fit,obsd=yyy,legPos="topleft")
```

---

plotBb *Plot a beta binomial distribution.*

Description

Plots the probabilities of a specified beta binomial distributon.

Usage

```
plotBb(m, s, size, ..., plot = TRUE, tikx = NULL, xlim = NULL,
        ylim = NULL, xlab = NULL, ylab = NULL, main = "")
```

Arguments

m	Numeric scalar between 0 and 1. May be interpreted as the "success probability".
s	Numeric scalar, greater than 0. The overdispersion parameter of the distribution.
size	Integer scalar specifying the upper limit of the "support" of the beta binomial distribution under consideration. The support is the set of integers {0, 1, ..., size}.

...	Extra arguments that are passed to the plot() function.
plot	Logical scalar; should a plot be produced (or should the function simply return a data frame consisting of the relevant values)?
tikx	(Optional) vector of locations of the tick marks on the x-axis.
xlim	The x-limits of the plot. (See plot.default().)
ylim	The y-limits of the plot. (See plot.default().)
xlab	A label for the x-axis. (See plot.default().)
ylab	A label for the y-axis. (See plot.default().)
main	An overall title for the plot. (See plot.default(); see also title().)

Value

A data frame with numeric columns x and p. The x column consists of the integers from 0 to size. The p column consists of the appropriate probabilities of the x values, calculated by dbetabinom() from the rmutil package. The value is returned invisibly. A plot is produced as a side-effect if plot is TRUE.

Author(s)

Rolf Turner <r.turner@auckland.ac.nz>

See Also

plot.mleBb() plotDb() plot.mleDb()

Examples

```
plotBb(0.7,3,14,main="An exempular plot")
plotBb(0.3,3,14,col="red",xlab="count",main="A communist plot")
plotBb(0.1,10,14,col="blue",main="A royal plot")
plotBb(0.5,20,14,col="green",main="An ecological plot")
plotBb(0.5,20,14,xlim=c(0,15))
plotBb(0.5,20,14,xlim=c(0,15),tikx=3*(0:5))
```

---

plotDb *Plot a db distribution.*

Description

Plots the probabilities of a specified db distributon.

Usage

```
plotDb(alpha, beta, ntop, zeta, ..., plot = TRUE, tikx = NULL, xlim = NULL,
        ylim = NULL, xlab = NULL, ylab = NULL, main = "")
```

Arguments

alpha	See ddb().
beta	See ddb().
ntop	See ddb().
zeta	See ddb().
...	Extra arguments that are passed to the plot() function.
plot	Logical scalar; should a plot be produced (or should the function simply return a data frame consisting of the relevant values)?
tikx	(Optional) vector of locations of the tick marks on the x-axis.
xlim	The x-limits of the plot. (See plot.default().)
ylim	The y-limits of the plot. (See plot.default().)
xlab	A label for the x-axis. (See plot.default().)
ylab	A label for the y-axis. (See plot.default().)
main	An overall title for the plot. (See plot.default(); see also title().)

Value

A data frame with numeric columns x and p. The x column consists of the integers from 0 to ntop or from 1 to ntop depending on whether zeta is TRUE. The p column consists of the appropriate probabilities of the x values, calculated by ddb(). The value is returned invisibly. A plot is produced as a side-effect if plot is TRUE.

Author(s)

Rolf Turner <r.turner@auckland.ac.nz>

See Also

plot.mleDb()

Examples

```
plotDb(2,3,14,FALSE,main="An exempular plot")
plotDb(2,3,14,TRUE,col="red",xlab="count",main="A communist plot")
plotDb(0.1,3,14,TRUE,col="blue",main="A royal plot")
plotDb(0.1,0.3,14,TRUE,col="green",main="An ecological plot")
plotDb(2,3,14,FALSE,xlim=c(0,15))
plotDb(2,3,14,FALSE,xlim=c(0,15),tikx=3*(0:5))
par(mfrow=c(2,1))
plotDb(2,2,5,FALSE,main=bquote(paste(alpha == 2,", ",beta == 2)),col="red")
plotDb(-2,-2,5,FALSE,main=bquote(paste(alpha == -2,", ",beta == -2)),col="blue")
```

simulate *Simulate data from a db or beta binomial distribution.*

Description

Simulate one or more data sets from a db or beta binomial distribution. The parameters of the distribution may be equal to those obtained from fitting the distibution to data, using `mleDb()` or `mleBb()`. They may also be specified by the user via the function `makeDbdpars()` or `makeBbdpars()`.

Usage

```
   ## S3 method for class 'mleDb'
simulate(object, nsim = 1, seed = NULL, ...,
                           ndata = NULL, drop = TRUE)
   ## S3 method for class 'mleBb'
simulate(object, nsim = 1, seed = NULL, ...,
                           ndata = NULL, drop = TRUE)
   ## S3 method for class 'Dbdpars'
simulate(object, nsim = 1, seed = NULL, ...,
                           ndata = NULL, drop = TRUE)
   ## S3 method for class 'Dbdpars'
simulate(object, nsim = 1, seed = NULL, ...,
                           ndata = NULL, drop = TRUE)
```

Arguments

`object`	An object of class `"mleDb"` as returned by `mleDb()`, or of class `"mleBb"` as returned by `mleBb()`, or of class `"Dbdpars"` as returned by `makeDbdpars()` or of class `"Bbdpars"` as returned by `makeBbdpars()`.
`nsim`	The number of data sets to simulate.
`seed`	Integer vector of seeds for random number generation. It should be of length either 1 or `nsim`. If it is of length less than `nsim` then `set.seed(seed[1])` is called and `seed` is replaced by a vector of seeds of length `nsim` which is created by sampling from `1:1e5`. Note that in this case all entries but the first of `seed` are *ignored*. If it is longer than `nsim`, then only the first `nsim` entries are used and the others are ignored. If `seed` it is not supplied it is created by sampling `nsim` values from `1:1e5`.
`...`	Not used.
`ndata`	Integer vector specifying the lengths of the data sets to be simulated. If it is of length less than `nsim` it is "recycled" to provide a vector of length `nsim`. If is longer than `nsim`, then only the first `nsim` entries are used and the others are ignored. If `ndata` is not supplied it is taken to be equal to the `"ndata"` attribute of `object` (i.e. the length of the data set from which the parameters in `object` were estimated.
`drop`	Logical scalar; if `TRUE` and if `nsim==1` then this function simply returns the simulated data set (an integer vector) rather than a list of length 1 whose sole entry is that data set. If `nsim>1` then `drop` is ignored.

Details

The actual simulation is done by `rdb()` or by the `rbetabinom()` function from the `rmutil` package.

Value

A list, of length `nsim`, whose entries are integer vectors, the length of of the `i`th entry being equal to `ndata[i]`. Each entry has an attribute `"seed"` which is the random number generation seed that was used in the generation of the data in that entry. If `nsim==1` and if `drop` is `TRUE`, then the value is simply an integer vector (of length `ndata[1]`).

See Also

`simulate()` `rdb()`

Examples

```
X   <- hmm.discnp::Downloads
fit <- mleDb(X,ntop=15,zeta=TRUE)
s1  <- simulate(fit)
s2  <- simulate(fit,nsim=5) # All data sets of length 267.
s3  <- simulate(fit,nsim=5,ndata=100*(2:6))
obj <- makeDbdpars(alpha=2,beta=3,ntop=20,zeta=TRUE,ndata=500)
s4  <- simulate(obj,nsim=5,seed=1:5)
```

---

varBb *Variance of a beta binomial distribution.*

Description

Calculate the variance of a random variable having a beta binomial distribution.

Usage

```
varBb(mo,...)
## S3 method for class 'mleBb'
varBb(mo,...)
## Default S3 method:
varBb(mo, s, size, ...)
```

Arguments

`mo`	For the `"mleBb"` method this argument is an object of class `"mleBb"` as returned by `mleBb()`. For the default method it is a numeric scalar, between 0 and 1, playing the role of `m` (which may be interpreted as the "success" probability). (See the help for `dbetabinom()`.)
`s`	Numeric scalar, greater than 0. The overdispersion parameter of the distribution. (See the help for `dbetabinom()`.)

size	Integer scalar specifying the upper limit of the "support" of the beta binomial distribution under consideration. The support is the set of integers {0, 1, ..., size}. (See the help for dbetabinom().)
...	Not used.

Details

For the "mleBb" method, the single argument should really be called (something like) "object" and for the default method the first argument should be called m. However the argument lists must satisfy the restrictions that "*A method must have all the arguments of the generic, including ... if the generic does.*" and "*A method must have arguments in exactly the same order as the generic.*"

For the "mleBb" method, the values of m and s are obtained from mo, and size is extracted from the attributes of mo.

The variance of a beta binomial distribution is readily calculable "by hand". These functions are provided for convenience and to preserve parallelism with the db distribution.

Value

Numeric scalar equal to the variance of a beta binomial distributed random variable with the given parameters.

Author(s)

Rolf Turner <r.turner@auckland.ac.nz>

See Also

varDb() expValDb() expValBb()

Examples

```
varBb(0.7,0.1,15)
varBb(0.7,400,15)
X   <- hmm.discnp::Downloads
fit <- mleBb(X,size=15)
varBb(fit)
```

---

varDb *Variance of a db distribution.*

Description

Calculate the variance of a random variable having a db distribution.

Usage

```
varDb(ao,...)
## S3 method for class 'mleDb'
varDb(ao,...)
## Default S3 method:
varDb(ao, beta, ntop, zeta=FALSE,...)
```

Arguments

ao	For the "mleDb" method this argument is an object of class "mleDb" as returned by mleDb(). For the default method it is a numeric scalar playing the role of alpha (see ddb()).
beta	See ddb().
ntop	See ddb().
zeta	See ddb().
...	Not used.

Details

For the "mleDb" method, the single argument should really be called (something like) "object" and for the default method the first argument should be called alpha. However the argument lists must satisfy the restrictions that "*A method must have all the arguments of the generic, including ... if the generic does.*" and "*A method must have arguments in exactly the same order as the generic.*"

For the "mleDb" method, the values of alpha and beta are obtained from ao, and ntop, and zeta are extracted from the attributes of ao.

The variance of a db distribution is theoretically intractable but is readily calculable numerically as

$$\sum (x-\mu)^2 \times \Pr(X=x)$$

, where μ is the expected value of the given distribution.

Value

Numeric scalar equal to the variance of a db distributed random variable with the given parameters.

Author(s)

Rolf Turner <r.turner@auckland.ac.nz>

See Also

ddb() expValDb()

Examples

```
varDb(3,4,15)
varDb(3,4,15,TRUE)
X   <- hmm.discnp::Downloads
fit <- mleDb(X,ntop=15,zeta=TRUE)
varDb(fit)
```

`vcov.mleBb` *Retrieve the covariance matrix from an* `"mleBb"` *object.*

Description

Extract the covariance matrix attribute an object of class `"mleBb"`. I.e. obtain the estimated covariance matrix of the maximum likelihood estimates of the parameters of a beta binomial distribution.

Usage

```
   ## S3 method for class 'mleBb'
vcov(object, ...)
```

Arguments

`object`	An object of class `"mleBb"` as returned by `mleBb()`.
`...`	Not used.

Details

The estimated covariance matrix is the inverse of the hessian of the negative log likelihood. (This may also be referred to as the observed Fisher information — the Fisher information evaluated at the maximum likelihood estimates of the parameters).

Value

A two-by-two positive definite (with any luck!) numeric matrix. It is an estimate of the covariance matrix of the parameter estimates.

Author(s)

Rolf Turner `<r.turner@auckland.ac.nz>`

See Also

`vcov.mleDb)` `mleBb()`

Examples

```
X     <- hrsRcePred
top1e <- X[X$sbjType=="Expert","top1"]
fit   <- mleBb(top1e,size=10)
vcov(fit)
```

`vcov.mleDb` *Retrieve the covariance matrix from an* `"mleDb"` *object.*

Description

Extract the covariance matrix attribute an object of class `"mleDb"`. I.e. obtain the estimated covariance matrix of the maximum likelihood estimates of the parameters of a db distribution.

Usage

```
   ## S3 method for class 'mleDb'
vcov(object, ...)
```

Arguments

`object`	An object of class `"mleDb"` as returned by `mleDb()`.
`...`	Not used.

Details

The estimated covariance matrix is the inverse of the hessian of the negative log likelihood. (This may also be referred to as the observed Fisher information — the Fisher information evaluated at the maximum likelihood estimates of the parameters).

Value

A two-by-two positive definite (with any luck!) numeric matrix. It is an estimate of the covariance matrix of the parameter estimates.

Author(s)

Rolf Turner `<r.turner@auckland.ac.nz>`

See Also

`vcov.mleBb)` `mleDb()`

Examples

```
X <- hmm.discnp::SydColDisc
X$y <- as.numeric(X$y)
X <- split(X,f=with(X,interaction(locn,depth)))
fitz <- lapply(X,function(x){mleDb(x$y,ntop=5)})
lapply(fitz,vcov)
```

`visRecog` *Visual recognition data.*

Description

Counts of successes in visual recognition memory for large and small binary pictures.

Usage

```
data("visRecog")
```

Format

A data frame with 30 observations on the following 4 variables.

`deck` An integer vector indicating which of two decks of cards, bearing graphic images, was used in the given experiment.

`subject` An integer vector indexing the (human) subjects in the experiments.

`tot5` An integer vector whose entries are counts of successes when the cards used consist of a 5×5 grid of "facets".

`tot10` An integer vector whose entries are counts of successes when the cards used consist of a 10×10 grid of "facets".

Details

Adult subjects were shown a series of cards, each bearing a simple graphic image. Each image resembled one face of a Rubik's cube, formed of either a 5x5 or a 10x10 grid of facets, each facet being either black or white. Later, each subject was shown a series of 20 similar cards, exactly 10 of which had been shown to the subject previously. The subject's task was to identify each image as a new one, or as a previously seen one. The response variable `tot5` is the number of correct identifications, out of 20, for the 5×5 cards. Similarly the variable `tot10` is the number of correct identifications for the 10×10 cards.

Subjects 21–30 were (deliberately) tested with a different set of cards than subjects 1–20, to ensure that results were not a function of the original deck of cards. (This seems to have no actual relevance.)

Source

The data are taken from the paper sited in **References** below. They were provided by a generous email correspondent who prefers to remain anonymous.

References

Green, D. M., and Purohit, A. K. (1976). Visual recognition memory for large and small binary pictures. *Journal of Experimental Psycholology: Human Learning and Memory* **2**, pp. 32–37.

Examples

```
dbfit5  <- with(visRecog,mleDb(tot5,20,TRUE))
dbfit10 <- with(visRecog,mleDb(tot10,20,TRUE))
set.seed(42) # To get repeatable Monte Carlo p-values.
print(gof(dbfit5,obsd=visRecog[["tot5"]],MC=TRUE)$pval)   # 0.86
print(gof(dbfit10,obsd=visRecog[["tot10"]],MC=TRUE)$pval) # 0.68
bbfit5  <- with(visRecog,mleBb(tot5,20))
bbfit10 <- with(visRecog,mleBb(tot10,20))
set.seed(42) # To get repeatable Monte Carlo p-values.
print(gof(bbfit5,obsd=visRecog[["tot5"]],MC=TRUE)$pval)   # 0.94
print(gof(bbfit10,obsd=visRecog[["tot10"]],MC=TRUE)$pval) # 0.70
```

Index