Package 'dbd'

April 30, 2021

Version 0.0-8

Date 2021-04-30

Title Versatile Discrete Distributions

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

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

Description Tools for working with a new family of versatile discrete distributions, the db (``discretised Beta'') family. 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.

Depends R (>= 3.2.2)

Suggests hmm.discnp, MASS

LazyData true

License GPL (>= 2)

NeedsCompilation no

Repository CRAN

Date/Publication 2021-04-30 12:50:03 UTC

R topics documented:

aHess 2
db 3
eow 5
expValDb 6
finfo 7
gof 9

hrsRcePred 11
llPlot 12
logLikDbd 14
makeDbdpars 15
mcCovMat 16
mleDb 18
ndata 21
nHess 22
plot.mleDb 24
plotDb 25
simulateDbd 27
varDb 28
vcov.mleDb 29
visRecog 30

Index **32**

`aHess` *Analytic hessian.*

Description

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

Usage

```
aHess(object)
```

Arguments

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

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"` 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

```
if(requireNamespace("hmm.discnp")) {
    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   <- mleDb(top1e,ntop=10,zeta=TRUE)
H     <- aHess(fit)
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, as it should.
```

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

---

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 given the parameters of that distribution. A specified number of observations ndata must be supplied. The inverse of the Fisher information is an estimate of the covariance matrix of the parameter estimates.

Usage

```
finfo(alpha, beta, ntop, zeta = FALSE, ndata)
```

Arguments

alpha	See ddb().
beta	See ddb().
ntop	See ddb().
zeta	See ddb().
ndata	The number of observations for which the Fisher information is being determined.

Details

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

Note that the number of observations must be supplied explicitly as ndata whereas for aHess() this number is extracted from the object argument.

This function in effect calculates the *expected* information, since the information matrix does not depend on the parameters.

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() nHess() mleDb()

Examples

```
print(finfo(0.6,0.3,5,FALSE,54))
if(requireNamespace("hmm.discnp")) {
    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 <- unname(fit["alpha"])
    beta  <- unname(fit["beta"])
    ntop  <- attr(fit,"ntop")
    zeta  <- attr(fit,"zeta")
    ndat  <- ndata(fit)
    print(finfo(alpha,beta,ntop,zeta,ndat))
    print(aHess(fit)) # Same as above.
}
```

```
X     <- hrsRcePred
top1e <- X[X$sbjType=="Expert","top1"]
fit <- mleDb(top1e,ntop=10,zeta=TRUE)
print(finfo(alpha=unname(fit["alpha"]),beta=unname(fit["beta"]),
            ntop=10,zeta=TRUE,ndata=ndata(fit)))
# Gives the expected Fisher info evaluated at the maximum
# likelihood parameter estimates.
print(aHess(fit)) # Same as above.
```

---

gof *Goodness of fit test for db distributions.*

Description

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

Usage

```
gof(object,obsd,...,test=TRUE,MC=FALSE,nsim=99,maxit=1000)
```

Arguments

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

Details

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 are 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.

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!

Author(s)

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

See Also

`mleDb()`

Examples

```
X <- hmm.discnp::Downloads
f <- mleDb(X,15,TRUE)
tst <- gof(f,X) # Gives warning that the chi-squared test is invalid.
set.seed(42)
tst2 <- gof(f,X,MC=TRUE)
# 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.
```

---

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)
## Not run:  # Takes too long for CRAN
    set.seed(42) # To get repeatable Monte Carlo p-values.
    print(gof(dbfit1e,obsd=top1e,MC=TRUE,maxit=10000)$pval) # 0.41
    print(gof(dbfit1n,obsd=top1n,MC=TRUE)$pval) # 0.75
    print(gof(dbfit3e,obsd=top3e,MC=TRUE)$pval) # 0.37
    print(gof(dbfit3n,obsd=top3n,MC=TRUE,maxit=8000)$pval) # 0.36

## End(Not run)
```

`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, ntop, zeta, 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 distribution.
`ntop`	See `mleDb()` and `ddb()`.
`zeta`	See `mleDb()` and `ddb()`.
`alim`	Numeric vector of length 2; the range of `alpha` values over which the surface is to be plotted. Defaults to `c(0,10)`.

blim	Numeric vector of length 2; the range of beta values over which the surface is to be plotted. Defaults to c(0,10).
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.

The default values for alim and blim may well be inappropriate. (To say the least!) Experimentation will probably be needed.

Value

A list with entries

x	The vector of values of the first parameter (alpha) 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) 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". 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

mleDb() persp() contour()

Examples

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

## End(Not run)

# Negative (!) parameters.
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")

X     <- hrsRcePred
top1e <- X[X$sbjType=="Expert","top1"]
srf   <- llPlot(top1e,ntop=10,zeta=TRUE,plotType="c")
fit   <- mleDb(top1e,ntop=10,zeta=TRUE)
points(fit[1],fit[2],col="red")
```

`logLikDbd` *Retrieve the (maximised) log likelihood from an* `"mleDb"` *object.*

Description

Extract the log likelihood attribute an object of class `"mleDb"`. I.e. obtain the maximum log likelihood in respect of the estimation of the parameters of a db distribution.

Usage

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

Arguments

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

Value

An object of class "logLik". Such an object consists of a numeric scalar equal to the maximum of the log likelihood of the model being fitted. In this instance the maximum is taken over the parameters of a db distribution which was fitted to a specified data set. The object has an attribute "df" ("degrees of freedom") equal to the number of estimated parameters, in this instance 2.

Author(s)

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

See Also

mleDb()

Examples

```
if(requireNamespace("hmm.discnp")) {
    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   <- mleDb(top1e,ntop=10,zeta=TRUE)
    logLik(fit)
}
```

---

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. May be abbreviated.
beta	The second "shape" parameter of the db distribution. May be abbreviated.
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

simulateDbd

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 distribution via simulation.

Usage

```
mcCovMat(object, nsim = 100, seed=NULL, maxit=1000)
```

Arguments

object	An object of class "mleDb" as returned by the function mleDb() or an object of class "Dbdpars", possibly as created by makeDbdpars().
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. If object is of class "mleDb" then this size is the same as that of the data set to which object was fitted). If object is of class "Dbdpars" then this size is the "ndata" component of object. The simulation is based on the parameter estimates, or specified values, contained in object.

From each such simulated data set, parameter estimates are obtained. The covariance matrix of these latter parameter estimates is taken to be the required covariance matrix estimated.

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

aHess() nHess() vcov.mleDb() makeDbdpars()

Examples

```
## Not run:  # Takes too long for CRAN.
    if(requireNamespace("hmm.discnp")) {
        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   <- mleDb(top1e,ntop=10,zeta=TRUE)
CM.m  <- mcCovMat(fit,nsim=500,maxit=5000)
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)
obj <- makeDbdpars(alpha=2.5,beta=4,ntop=10,zeta=TRUE,ndata=30)
CM.m <- mcCovMat(obj,nsim=500)
CM.f <- solve(do.call(finfo,obj))
cat("Monte Carlo:\n\n")
print(CM.m)
cat("From Fisher info:\n\n")
print(CM.f)

## End(Not run)
```

`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

```
if(requireNamespace("hmm.discnp")) {
    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,covmat=FALSE)})
    sapply(fitz,ndata)
    ## Not run:  # Takes too long for CRAN.
        Y    <- hmm.discnp::ccprSim
        Ypr  <- lapply(Y,function(x){x[,2]})
        OK   <- sapply(Ypr,function(x){sum(!is.na(x)) > 0})
        Ypr  <- Ypr[OK]
    # Need the try() in the following since mleDb() falls over on some of
    # the data sets; not clear why.
        fitz <- lapply(Ypr,function(x){try(mleDb(x,ntop=4,zeta=TRUE,
                       covmat=FALSE),silent=TRUE)})
        ok   <- sapply(fitz,function(x){!inherits(x,"try-error")})
        fitz <- fitz[ok]
        sapply(fitz,ndata)

## End(Not run)
}
set.seed(42)
obj <- makeDbdpars(alpha=0.2,beta=0.25,ntop=20,zeta=TRUE,
                   ndata=sample(200:600,100))
Y   <- simulate(obj,nsim=100)
fitz <- lapply(Y,mleDb,ntop=20,zeta=TRUE,covmat=FALSE)
sapply(fitz,ndata)
```

---

nHess *Numerical hessian calculation.*

Description

Calculate an approximation to the hessian of the **negative** log likelihood of a db 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" as returned by the function mleDb().
x	Numeric vector of non-negative integer data, presumably the data set on the basis of which object was calculated.
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 `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 a db distribution, 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()` `optim()` `optimHess()`

Examples

```
if(requireNamespace("hmm.discnp")) {
    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))
    x <- hmm.discnp::Downloads
    fit <- mleDb(x,ntop=15,zeta=TRUE)
    H   <- nHess(fit,x)
    print(solve(H)) # Compare with ...
    print(vcov(fit))
}
```

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().

Usage

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

Arguments

x	An object of class "mleDb" as returned by the function mleDb()
...	Not used.
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.
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.)
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

None. A plot is produced as a side effect.

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()`

Examples

```
if(requireNamespace("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")
```

---

plotDb *Plot a db distribution.*

Description

Plots the probabilities of a specified db distributon.

Usage

```
plotDb(alpha, beta, ntop, zeta, ..., 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.
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

None. A plot is produced as a side-effect.

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))
OP <- 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")
par(OP)
```

simulateDbd *Simulate data from a fitted db distribution.*

Description

Simulate one or more data sets from a db distribution. The parameters may be equal to those obtained from fitting such a distibution to data using `mleDb()`. Alternatively they may be specified in an object of class `"Dbdpars"`.

Usage

```
   ## S3 method for class 'mleDb'
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 `"Dbdpars"`. Objects of this latter class may be conveniently constructed using the function `makeDbdpars()`.
`nsim`	The number of data sets to simulate.
`seed`	Integer scalar specifying the seed for random number generation. If `seed` it is not supplied it is created by sampling 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()`.

Value

A list, of length `nsim`, whose entries are integer vectors, the length of of the `i`th entry being equal to `ndata[i]`. If `nsim==1` and if `drop` is `TRUE`, then the value is simply an integer vector (of length `ndata[1]`).

The returned value has an attribute "seed" equal to the value of the seed that was set before the generation of the samples was initiated. This attribute is equal to the value of the seed argument if this was supplied

Author(s)

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

See Also

simulate() rdb() makeDbdpars()

Examples

```
if(requireNamespace("hmm.discnp")) {
    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))
}
```

---

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 (passed to ddb()) 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)
if(requireNamespace("hmm.discnp")) {
    X   <- hmm.discnp::Downloads
    fit <- mleDb(X,ntop=15,zeta=TRUE)
    varDb(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 covariance matrix attribute may not exist (may be NULL) if the argument covmat of mleDb() was set to FALSE. In this case the value returned by this function is NA.

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

Value

A two-by-two positive definite (with any luck!) numeric matrix, or NA (see **Details**. When a matrix is returned it is an estimate of the covariance matrix of the parameter estimates.

Author(s)

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

See Also

mleDb()

Examples

```
if(requireNamespace("hmm.discnp")) {
    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
```

Index

∗ **Fisher information**
finfo, 7
∗ **covariance estimation**
aHess, 2
finfo, 7
mcCovMat, 16
nHess, 22
∗ **datagen**
simulateDbd, 27
∗ **datasets**
hrsRcePred, 11
visRecog, 30
∗ **distribution**
db, 3
∗ **expected value**
expValDb, 6
∗ **hessian**
aHess, 2
nHess, 22
∗ **hplot**
llPlot, 12
plot.mleDb, 24
plotDb, 25
∗ **htest**
gof, 9
∗ **inference**
aHess, 2
finfo, 7
mcCovMat, 16
nHess, 22
∗ **math**
expValDb, 6
varDb, 28
∗ **univar**
expValDb, 6
varDb, 28
∗ **utilities**
eow, 5
logLikDbd, 14
makeDbdpars, 15
ndata, 21
vcov.mleDb, 29
∗ **variance**
varDb, 28

aHess, 2, *8*, *17*, *20*, *23*

contour, *13*

db, 3
dbd *(db)*, 3
ddb, *6–8*, *12*, *19*, *20*, *26*, *28*, *29*
ddb *(db)*, 3

eow, 5
expValDb, 6, *29*

finfo, *2*, *3*, 7

get.eow *(eow)*, 5
gof, 9

hrsRcePred, 11

legend, *24*
llPlot, 12
logLik.mleDb *(logLikDbd)*, 14
logLikDbd, 14

makeDbdpars, 15, *17*, *27*, *28*
mcCovMat, 16
meDb, *5*, *19*, *20*
mleDb, *2*, *3*, *5*, *6*, *8–10*, *12–15*, *17*, 18, *21–25*, *27*, *28*, *30*

ndata, 21
nHess, *3*, *8*, *17*, 22

optim, *9*, *17–20*, *23*
optimHess, *22*, *23*
options, *6*

pdb *(db)*, 3
persp, *13*
plot.default, *26*
plot.mleDb, 24, *26*
plotDb, *25*, 25

qdb *(db)*, 3

rdb, *27, 28*
rdb *(db)*, 3

set.eow, *19*
set.eow *(eow)*, 5
simulate, *28*
simulate.Dbdpars *(simulateDbd)*, 27
simulate.mleDb *(simulateDbd)*, 27
simulateDbd, *16*, 27

title, *26*

varDb, *7*, 28
vcov.mleDb, *17*, *20*, 29
visRecog, 30