

# Package ‘hyperdirichlet’

January 2, 2012

**Type** Package

**Title** A generalization of the Dirichlet distribution

**Version** 1.4-6

**Date** 2008-04-28

**Depends** R (>= 2.7.0), methods, aylmer, abind, mvtnorm, cubature

**Author** Robin K.S. Hankin

**Maintainer** <hankin.robin@gmail.com>

**Description** A suite of routines for the hyperdirichlet distribution

**License** GPL-2

**LazyLoad** yes

**Repository** CRAN

**Date/Publication** 2011-12-27 10:06:11

## R topics documented:

hyperdirichlet-package . . . . .	2
Arith-methods . . . . .	3
B . . . . .	4
bernoulli . . . . .	6
binmat . . . . .	8
dhyperdirichlet . . . . .	9
diri_norm . . . . .	11
doubles . . . . .	12
Extract.hyperdirichlet . . . . .	13
e_to_p . . . . .	14
fitz . . . . .	16
gd . . . . .	17
hd_add . . . . .	19

head . . . . .	20
hyperdirichlet . . . . .	21
hyperdirichlet-class . . . . .	22
justpairs . . . . .	24
matrix_to_HD . . . . .	25
maximum_likelihood . . . . .	26
maxmult . . . . .	27
paircomp . . . . .	28
params . . . . .	29
paulino . . . . .	30
pnames . . . . .	31
pollen . . . . .	31
print . . . . .	32
serum . . . . .	33
triplot . . . . .	34
uniform . . . . .	36
volleyball . . . . .	37

**Index** **39**

hyperdirichlet-package

*The Hyperdirichlet package*

**Description**

A generalization of the Dirichlet distribution

**Details**

Package: hyperdirichlet  
 Type: Package  
 Version: 1.1-8  
 Date: 2008-03-26  
 License: GPL

This package provides a generalization of the Dirichlet distribution that is useful for analyzing multinomial trials with *a priori* restrictions.

As an example, consider six people (“players”), numbered 1 to 6. These players are members of a running club and regularly race one another.

Each player has an associated number  $p_1$  to  $p_6$ , with  $0 \leq p_i \leq 1$  for  $i = 1, \dots, 6$  and  $\sum_{i=1}^6 p_i = 1$ . If all six take part in a race, then the probability that player  $i$  wins is simply  $p_i$ .

We wish to make inferences about the  $p_i$  from their performances.

If all six race and  $p_i$  wins  $n_i$ , then the likelihood function is just

$$p_1^{n_1} \cdot p_2^{n_2} \cdot p_3^{n_3} \cdot p_4^{n_4} \cdot p_5^{n_5} \cdot p_6^{n_6}.$$

With a uniform prior, the posterior is Dirichlet.

The players now have a race but only  $p_1$ ,  $p_2$  and  $p_3$  take place, winning  $r_1$ ,  $r_2$  and  $r_3$  respectively. The likelihood function is then

$$\frac{p_1^{n_1+r_1} \cdot p_2^{n_2+r_2} \cdot p_3^{n_3+r_3} \cdot p_4^{n_4} \cdot p_5^{n_5} \cdot p_6^{n_6}}{(p_1 + p_2 + p_3)^{r_1+r_2+r_3}}$$

This distribution is not a Dirichlet distribution but is representable in this package; the R idiom would be

```
jj <- dirichlet(powers = c(5,4,3,5,3,2))  jj <- jj + mult_restricted_obs(6,
1:3, c(4,5,2))
```

where the first line specifies a Dirichlet distribution for the all-play data and the second line augments the likelihood with the observations from the restricted race.

#### Author(s)

Robin K. S. Hankin

#### References

Robin K. S. Hankin (2010). "A Generalization of the Dirichlet Distribution", *Journal of Statistical Software*, 33(11), 1-18, <http://www.jstatsoft.org/v33/i11/>

#### Examples

```
jj <- dirichlet(powers = c(5,4,3,5,3,2))
jj <- jj + mult_restricted_obs(6, 1:3, c(4,5,2))

data(icons)
maximum_likelihood(as.hyperdirichlet(icons))
```

---

Arith-methods

*Arith methods for hyperdirichlet objects*

---

#### Description

Methods for Arithmetic functions on hyperdirichlet objects. There are only two operations:

- Add two of the same dimensions, as in 'a+b'
- Multiply a hyperdirichlet object by a length-one scalar, as in 'a\*5'

Both these operations result in the normalization constant becoming unknown.

No unary operations are defined.

**Value**

Arithmetic functions on hyperdirichlet objects return hyperdirichlet objects. Multiplication operates on the powers (not parameters), thus it is defined so that  $a+a = 2*a = a*2$ ; associativity means that  $a+\dots+a$  ( $n$  copies) equals  $n*a$ .

**Note**

The package delegates ‘a+b’ to `hd_add()`, which the user should use if more control is desired.

**Author(s)**

Robin K. S. Hankin

**See Also**

[hd\\_add](#)

**Examples**

```
gd(1:3,1:3) + dirichlet(1:4)
```

```
dirichlet(1:4) * 10
```

---

B

*Normalizing constant for the hyperdirichlet distribution*

---

**Description**

Uses numerical techniques for calculating the normalizing constant for the hyperdirichlet distribution

**Usage**

```
B(x, ...)
NC(x)
calculate_B(x, disallowed=NULL, give=FALSE, ...)
probability(x, disallowed, ...)
mgf(x, powers, ...)
mean(x, ...)
is.proper(x,irregardless)
validated(x)
```

### Arguments

<code>x</code>	Object of class “hyperdirichlet” (or coerced thereto)
<code>powers</code>	Vector of length $\dim(x)$ whose elements are the powers of the expectation; see details section
<code>irregardless</code>	Boolean; see details section
<code>disallowed</code>	Function specifying a subset of the simplex over which to integrate; default NULL means to integrate over the whole simplex. The integration proceeds over $p$ with <code>disallowed(p)</code> evaluating to FALSE
<code>give</code>	Boolean, with default FALSE meaning to return the value of the integral and TRUE meaning to return the full output of <code>adaptIntegrate()</code>
<code>...</code>	Further arguments passed to <code>adaptIntegrate()</code>

### Details

- Function `B()` is the user-friendly version. It accesses the `NC` slot. If not NA, the value is returned; if NA, the normalizing constant is calculated using `adaptIntegrate()` of the `cubature()` package, via `calculate_B()`.
- Function `NC()` is not intended for the user. It is used internally as an accessor method for the `NC` slot, and this value is returned indiscriminately.
- Function `calculate_B()` is the engine which actually does the work. Observe how  $p$  is converted to  $e$  (by `e_to_p()`) and the integral proceeds over a hypercube. Function `dirichlet()` and `gd()` do not use this as the normalizing constant has an analytical expression and this is used instead.
- Function `probability()` gives the probability of an observation from a hyperdirichlet distribution satisfying `!disallowed(p)`.
- Function `mgf()` is the moment generating function, taking an argument that specifies the powers of  $p$  needed: the expectation of  $\prod_{i=1}^n p_i^{\text{powers}[i]}$  is returned.
- Function `mean()` returns the mean value of the hyperdirichlet distribution. This is computationally slow (consider `maximum_likelihood()` for a measure of central tendency). The function takes a `normalize` argument, not passed to `adaptIntegrate()`: this is Boolean with FALSE meaning to return the value found by integration directly, and default TRUE meaning to normalize so the sum is exactly 1
- Function `is.proper()` checks a hyperdirichlet distribution for being normalizable: a “proper” hyperdirichlet object has a finite integral and therefore can be normalized. This function is quite time-consuming for hyperdirichlet distributions of large dimension.  
The `irregardless` argument to function `is.proper()` is Boolean, with TRUE meaning to carry out the checks whatever the value of slot `@validated` [that is, `validated(x)`]. Default FALSE means that function `is.proper()` returns TRUE if `@validated` is TRUE and to carry out the check otherwise. Use this argument to force `is.proper()` to carry out a check even if not strictly necessary.
- Function `validated()` is an accessor method for the `@validated` slot of hyperdirichlet object. It returns a Boolean variable with TRUE meaning that the object is **known** to be “proper” (ie `is.proper(x)` returns TRUE), so it is normalizable, even if the normalization constant is not known. This flag is present because many hyperdirichlet objects of interest are known *a priori* to be proper, so executing `is.proper()` would be unnecessary.

**Value**

Functions `B()`, `NC()`, `calculate_NC()` notionally return a scalar: the normalization constant

Functions `mean()` and `mgf()` return a  $k$ -tuple

Functions `is.proper()` and `validated()` return a Boolean

Function `probability()` returns a scalar, a probability.

**Note**

The `adapt` package is no longer available on CRAN; from 1.4-3, the package uses `adaptIntegrate` of the `cubature` package.

**Author(s)**

Robin K. S. Hankin

**See Also**

[hyperdirichlet](#)

**Examples**

```
a <- hyperdirichlet(c(4,3,6,5,4,3,2,1))
## Not run:
B(a)                                # Not recommended
a <- as.hyperdirichlet(a,TRUE)       # Recommended

is.proper(a)

mgf(a,powers=1:3)    # expectation of p1^1 * p2^2 * p3^3

## End(Not run)
```

---

bernoulli

*Hyperdirichlet distributions for various types of informative trials*

---

**Description**

Hyperdirichlet distributions for various types of informative trials including Bernoulli and multinomial

**Usage**

```

single_obs(d,n)
obs(x)
single_multi_restricted_obs(d,n,x)
mult_restricted_obs(d, a, nobs)
mult_bernoulli_obs(d, team1, team2, wins1, wins2)
single_bernoulli_obs(d, win, lose)
bernoulli_obs(d, winners, losers)

```

**Arguments**

d	Dimension of the distribution
n	Number of the winner
x	Summary statistic
a, win, lose, winners, losers, nobs, team1, team2, wins1, wins2	Arguments as detailed below

**Details**

These functions give likelihood functions for various observations. In the following, the paradigm is  $d$  players and the object of inference is  $p = (p_1, \dots, p_d)$  (the “skills”) with  $\sum p_i = 1$ . Different types of observation are possible.

The most informative is the unrestricted, uncensored case in which all  $d$  players play and the winner is identified unambiguously (`single_obs()`). However, other observations are possible, as detailed below:

- `single_obs(d,n)`. Single multinomial trial:  $d$  players, and player  $n$  wins.
- `obs(x)`. Repeated multinomial trials: `sum(x)` trials, each amongst `length(x)` players, with player  $i$  winning `x[i]` games (which might be zero)
- `single_multi_restricted_obs(d,n,x)`. Single restricted multinomial trial:  $d$  players, player  $n$  wins, conditional on the winner being one of `x[1]`, `x[2]`, etc
- `mult_restricted_obs(d,a,nobs)`. Multiple restricted multinomial trials:  $d$  players, conditional on winners being `a[1]`, `a[2]`, etc. Player `a[i]` wins `nobs[i]` times for  $1 \leq i \leq d$
- `mult_bernoulli_obs(d, team1, team2, wins1, wins2)`. Multiple Bernoulli trials between `team1` and `team2` with `team1` winning `wins1` and `team2` winning `wins2`
- `single_bernoulli_obs(d, win, lose)`. Single Bernoulli trial:  $d$  players, with two teams (`win` and `lose`). The winning team comprises `win[1]`, `win[2]`, etc and the losing team comprises `lose[1]`, `lose[2]`, etc.
- `bernoulli_obs(d, winners, losers)` Repeated Bernoulli trials:  $d$  players. Here `winners` and `losers` are lists of the same length; the elements are a team as in `single_bernoulli_obs()` above. Thus game  $i$  was between `winners[[i]]` and `losers[[i]]` and, of course, `winners[[i]]` won.

See examples section.

**Value**

All functions documented here return a hyperdirichlet object.

**Note**

The hyperdirichlet distributions returned by the functions documented here may be added (using “+”) to concatenate independent observations.

**Author(s)**

Robin K. S. Hankin

**Examples**

```
# Five players, some results:

jj1 <- obs(1:5) # five players, player 'i' wins 'i' games.
jj2 <- single_obs(5,2) # open game, p2 wins
jj3 <- single_multi_restricted_obs(5,2,1:3) # match: 1,2,3; p2 wins
jj4 <- mult_restricted_obs(5,1:2,c(0,4)) # match: 1,2, p1 wins 2 games, p2 wins 3
jj5 <- single_bernoulli_obs(5,1:2,3:5) # match: 1&2 vs 3&4&5; 1&2 win
jj6 <- mult_bernoulli_obs(6, 1:2,c(3,5), 7,8) # match: 1&2 vs 3&5; 1&2 win 7, 3&5 win 8
jj6 <- bernoulli_obs(5,list(1:2,1:2), list(3,3:5)) # 1&2 beat 3; 1&2 beat 3&4&5

# Now imagine that jj1-jj6 are independent observations:

ans <- jj1 + jj2 + jj3 + jj4 + jj5 + jj6 #posterior PDF with uniform prior likelihood
```

---

binmat

*Create a matrix of all binary combinations*

---

**Description**

Create a matrix of all binary combinations for use with the hyperdirichlet distribution

**Usage**

```
binmat(n, alternatives = NULL, pnames=NULL)
```

**Arguments**

n	Number of binary digits
alternatives	The alternatives; prototypically TRUE and FALSE, but default NULL, taken to mean 0 and 1, is easier on the eye
pnames	Optional vector specifying the column names; default of NULL means to use p1,p2 etc

**Value**

Function `binmat(n)` returns an integer matrix of zeros and ones with  $2^n$  rows and  $n$  columns. In binary, the rows count from 0 to  $2^n - 1$ .

**Author(s)**

Robin K. S. Hankin

**See Also**

[print.hyperdirichlet](#)

**Examples**

```
binmat(4)
```

```
binmat(3,alternatives=c(TRUE,FALSE))
```

---

dhyperdirichlet	<i>Probability density function for, and random sampling from, the hyperdirichlet distribution</i>
-----------------	--

---

**Description**

Probability density function for the hyperdirichlet distribution in terms of either  $p$  or  $e$ ; and random sampling using Metropolis-Hastings

**Usage**

```
dhyperdirichlet_e(e, HD, include.Jacobian = TRUE)
dhyperdirichlet(p, HD, include.NC = FALSE, TINY = 1e-10, log = FALSE)
rhyperdirichlet(n, HD, start=NULL, sigma=NULL)
```

**Arguments**

HD	Object of class <code>hyperdirichlet</code> , or coerced thereto
p	Vector of length <code>dim(HD)</code> , notionally summing to one
e	Vector of length <code>dim(HD)</code> giving the point in e-space
include.Jacobian	In function <code>dhyperdirichlet_e()</code> , Boolean with default TRUE meaning to include the Jacobian of the transform from e to p
include.NC	In function <code>dhyperdirichlet_e()</code> , Boolean with TRUE meaning to include the normalization factor and default FALSE meaning not to include it (it is expensive to calculate). Note that if the normalizing factor is not known, the function will return NA

TINY	In function dhyperdirichlet_p(), numeric, specifying minimum size for elements of p via <code>p &lt;- pmax(p , TINY)</code>
log	In function dhyperdirichlet_p(), Boolean with default FALSE meaning to return the probability density and TRUE meaning to return its logarithm
n, start, sigma	In function rhyperdirichlet(), n is the number of observations to take, start is the start-point for the random walk (with default NULL meaning to use the neutral point), and sigma is the standard deviation for the (Gaussian) kernel, with default NULL meaning to use $1/d$

### Details

Function dhyperdirichlet() gives the density as a function of the  $p_1, p_2, \dots, p_d$ .

Function dhyperdirichlet\_e() gives the density as a function of the  $e_i$ . This is useful when integrating as the simplex (in p-space) transforms to a hypercube in e-space.

### Value

Functions dhyperdirichlet() and dhyperdirichlet\_e() return a scalar; function rhyperdirichlet() returns a matrix whose rows are  $k$ -tuples

### Note

Function rhyperdirichlet() uses a Metropolis-Hastings algorithm to construct a Markov chain. Note that successive observations are not independent of one another. The details of this nonindependence are poorly understood (by me, at any rate). The examples section below shows how to generate a single random observation from the hyperdirichlet distribution.

Function dhyperdirichlet() silently normalizes p by `p <- p/sum(p)`.

The relationship between  $e$  and  $p$  is given in `e_to_p.Rd`.

### Author(s)

Robin K. S. Hankin

### See Also

[maximum.likelihood.e\\_to\\_p](#)

### Examples

```
dhyperdirichlet(c(1,4,3,2)/10, dirichlet(1:4))

rhyperdirichlet(20, dirichlet(1:3))

diff(c(0,sort(runif(9)),1)) # random sample drawn from dirichlet(rep(1,10))

# how to draw a single observation from the hyperdirichlet:
f <- function(HD,n,...){rhyperdirichlet(n,HD=HD,...)[n,,drop=TRUE]}
f(dirichlet(1:3),n=100)
```

```
f(dirichlet(1:3),n=100)
f(dirichlet(1:3),n=100)
# Note, n=100 might not be enough burn-in.
```

---

diri_norm	<i>Normalization constants for the Dirichlet and generalized Dirichlet distributions</i>
-----------	--

---

**Description**

Normalization constants for the Dirichlet and generalized Dirichlet distributions using computationally efficient methods

**Usage**

```
diri_norm(x)
gd_norm(a,b)
```

**Arguments**

x	Vector of parameters in the Dirichlet distribution
a,b	Vectors of parameters in the generalized Dirichlet distribution

**Value**

Returns the normalization constant:

$$\frac{\prod_{i=1}^k \Gamma(\alpha_i)}{\Gamma\left(\sum_{i=1}^k \alpha_i\right)}$$

for the Dirichlet and

$$\frac{\prod_{i=1}^k \Gamma(a_i) \Gamma(b_i)}{\prod_{i=1}^k \Gamma(a_i + b_i)}$$

for the generalized Dirichlet.

**Note**

The functions make use of the logarithmic form of the gamma and beta functions to avoid overflow

**Author(s)**

Robin K. S. Hankin

**References**

R. J. Connor and J. E. Mosimann 1969. "Concepts of independence for proportions with a generalization of the Dirichlet distribution". *Journal of the American Statistical Association*, volume 64, number 325, pp194-206

**See Also**

[dhyperdirichlet](#)

**Examples**

```
diri_norm(runif(9))
gd_norm(1:3 , 3:1)
```

---

doubles

*Match outcomes from repeated doubles tennis matches*

---

**Description**

Match outcomes from repeated doubles tennis matches

**Usage**

```
data(doubles)
```

**Format**

A hyperdirichlet object corresponding to the match outcomes listed below.

**Details**

There are four players,  $p_1$  to  $p_4$ . These players play doubles tennis matches with the following results:

match	score
$\{p_1, p_2\}$ vs $\{p_3, p_4\}$	9-2
$\{p_1, p_3\}$ vs $\{p_2, p_4\}$	4-4
$\{p_1, p_4\}$ vs $\{p_2, p_3\}$	6-7
$\{p_1\}$ vs $\{p_3\}$	10-14
$\{p_2\}$ vs $\{p_3\}$	12-14
$\{p_1\}$ vs $\{p_4\}$	10-14
$\{p_2\}$ vs $\{p_4\}$	11-10
$\{p_3\}$ vs $\{p_4\}$	13-13

It is suspected that  $p_1$  and  $p_2$  have some form of team cohesion and play better when paired than when either solo or with other players. As the scores show, each player and, apart from p1-p2, each

doubles partnership, is of approximately the same strength.

Dataset `doubles_noghost` gives the appropriate likelihood function for the players' strengths; and dataset `doubles` gives the appropriate likelihood function if the extra strength due to team cohesion of  $\{p_1, p_2\}$  is represented by a ghost player.

### Source

Doubles tennis matches at NOCS, Jan-May 2008

### References

Robin K. S. Hankin (2010). "A Generalization of the Dirichlet Distribution", *Journal of Statistical Software*, 33(11), 1-18, <http://www.jstatsoft.org/v33/i11/>

### Examples

```
data(doubles)
```

---

```
Extract.hyperdirichlet
```

*Extract or Replace parameters of a hyperdirichlet object*

---

### Description

Methods for "[" and "[<-", i.e., extraction or subsetting of hyperdirichlet objects.

### Arguments

<code>x</code>	Object of class <code>hyperdirichlet</code>
<code>i</code>	elements to extract or replace
<code>value</code>	replacement value

### Value

Always returns an object of class `hyperdirichlet`.

### Methods

- `x[i]`
- `x[i] <- value`

### Note

The replacement method "[<-" invokes the validity checking method `.hd.valid()` by default; this may be time-consuming for hyperdirichlet objects of high dimension.

Users may use the functional form for replacement to override this behaviour; the method takes `recalculate` and a validated arguments, defaulting to `FALSE`, that may be set if desired.

**Author(s)**

Robin K. S. Hankin

**Examples**

```

a <- uniform(4)
a[5] <- 1.2

## Not run:
a <- as.hyperdirichlet(a, TRUE) # recommended way to calculate NC

## End(Not run)

```

---

e\_to\_p

*Transform from a simplex to a hypercube*


---

**Description**

Transform from a simplex (either regular or right-angled) to a hypercube.

**Usage**

```

e_to_p(e)
p_to_e(p)
Jacobian(e)

```

**Arguments**

e                    Vector with all elements between 0 and 1 (hypercube)  
p                    Vector of positive elements whose sum is either equal to, or less than, one (simplex)

**Details**

Function e\_to\_p() takes one from e-space to p-space.

Function p\_to\_e() takes one from p-space to e-space. This is useful when integrating over a simplex; use Jacobian() to evaluate the Jacobian of the transform.

Forward transformation:

$$e_1 = \sum_{i=1}^d p_i$$

$$e_i = \frac{p_{i-1}}{\sum_{j=i-1}^d p_j}, \quad 2 \leq i \leq d$$

Backward transformation:

$$p_1 = e_1 e_2$$

$$p_i = e_1 e_{i+1} \prod_{j=2}^i (1 - e_j), \quad 2 \leq i \leq d$$

Jacobian:

$$J = \prod_{i=2}^{d-1} (1 - e_i)^{d-i}$$

### Value

The functions documented here return a scalar.

### Note

To do a regular simplex, use the “di” of the right-angled simplex; see the examples.

### Author(s)

Robin K. S. Hankin

### References

- M. Evans and T. Swartz 2000. *Approximating Integrals via Monte Carlo and Deterministic Methods*, Oxford University Press; page 28
- Robin K. S. Hankin (2010). “A Generalization of the Dirichlet Distribution”, *Journal of Statistical Software*, 33(11), 1-18, <http://www.jstatsoft.org/v33/i11/>

### See Also

[dhyperdirichlet](#)

### Examples

```
## Not run:
# First, try to calculate the volume of a regular 4-simplex:
adapt(5,rep(0,5),rep(1,5),functn=function(x){Jacobian(c(1,x))})
# Should be close to 1/5! = 1/120 ~= 0.008333
# (that was the 'di trick')

# Now, try to calculate the volume of a triangular-based pyramid:
adapt(3,rep(0,3),rep(1,3),functn=Jacobian)
# Should be close to 1/8=0.125

## End(Not run)
```

---

 fitz

*Fitzmaurice*


---

**Description**

Hyperdirichlet distribution corresponding to a dataset of Fitzmaurice et al

**Usage**

```
fitz(dat , include.missing=TRUE , validated=NULL)
```

**Arguments**

dat	A vector corresponding to either male or female data
include.missing	Boolean, with default TRUE meaning to return the likelihood function for the data including the missing cases, and FALSE meaning to include only the data corresponding to complete cases
validated	Boolean, with TRUE meaning to omit the checks (OK if all elements of dat are non-negative) and FALSE meaning to check them all (time-consuming)

**Details**

Fitzmaurice considered childhood obesity. See the reference for further details.

**Value**

Returns a hyperdirichlet distribution (without normalizing factor) corresponding to the observations, either male or female, made by Fitzmaurice et al.

**Note**

Pat Altham originally spotted that this dataset could be represented using the hyperdirichlet distribution.

The functional form for replacement “[<-()” is used because it is possible to set the validated argument to TRUE: this suppresses the computationally intensive checking that the distribution is proper.

**warning.** Setting validated to TRUE is not recommended in general. It is OK here because the function knows that no elements of dat is negative.

**Author(s)**

Robin K. S. Hankin

**References**

G. M. Fitzmaurice, N. M. Laird, and S. R. Lipsitz 1994. “Analysing incomplete longitudinal binary responses: a likelihood-based approach”. *Biometrics*, volume 50, pp601-612.

**Examples**

```

boys <- c(20,7,9,8, 8, 8,15,150,13,3,2,42,3,1, 6,16,11,1,3,38,14,55,4,33,7,45)
girls <- c(21,6,6,2,19,13,14,154, 8,1,4,47,4,0,16, 3,11,1,3,25,13,39,5,23,7,47)

male <- fitz(boys)
maximum_likelihood(male)

```

---

gd

*The Dirichlet and generalized Dirichlet distribution*


---

**Description**

Specify a Dirichlet or generalized Dirichlet distribution as a special case of the hyperdirichlet distribution

**Usage**

```

dirichlet(params, powers, pnames)
is.dirichlet(x)
dirichlet_params(x)
dirichlet_params(x) <- value
gd(a, b, b0 = 0, pnames = NULL)

```

**Arguments**

params,powers	Numeric vectors (supply exactly one) specifying the parameters or the powers respectively of the Dirichlet distribution
x	Object of class hyperdirichlet
value	Numeric vector
a,b	Numeric vectors of the same length specifying the parameters of the generalized Dirichlet distribution
b0	Arbitrary constant for the generalized Dirichlet distribution
pnames	Character vector for name of the hyperdirichlet object

**Details**

Function `dirichlet()` returns the hyperdirichlet distribution corresponding to the classical Dirichlet distribution. If the vector `params|powers` is a named vector, then the hyperdirichlet object inherits the names (but the names are ignored if argument `pnames` is supplied).

Function `is.dirichlet(x)` returns TRUE or FALSE according to whether the hyperdirichlet object `x` is a Dirichlet distribution.

Function `dirichlet_params()` returns the Dirichlet parameters of a hyperdirichlet object.

Function `gd()` returns the hyperdirichlet distribution corresponding to the generalized Dirichlet distribution of Connor and Mosimann.

For convenience, the generalized Dirichlet distribution described here. Connor and Mosimann 1969 give the PDF as

$$\left[ \prod_{i=1}^{k-1} B(a_i, b_i) \right]^{-1} p_k^{b_k-1} \prod_{i=1}^{k-1} \left[ p_i^{a_i-1} \left( \sum_{j=i}^k p_j \right)^{b_{i-1}-(a_i+b_i)} \right].$$

where  $\sum_{i=1}^k p_i = 1$  and  $b_0$  is arbitrary. If  $b_{i-1} = a_i + b_i$  for  $i = 2, \dots, k-1$  then the PDF reduces to a standard Dirichlet distribution with  $\alpha_i = a_i$  for  $i = 1, \dots, k-1$  and  $\alpha_k = b_{k-1}$ .

Wong 1998 gives the algebraically equivalent form

$$\prod_{i=1}^k \frac{1}{B(\alpha_i, \beta_i)} x_i^{\alpha_i-1} (1 - x_1 - \dots - x_i)^{\gamma_i}$$

for  $x_1 + x_2 + \dots + x_k \leq 1$  and  $x_j \geq 0$  for  $j = 1, 2, \dots, k$  and  $\gamma_j = \beta_j - \beta_{j+1}$  for  $j = 1, 2, \dots, k-1$  and  $\gamma_k = \beta_k - 1$ .

Here,  $B(x, y) = \Gamma(x)\Gamma(y)/\Gamma(x+y)$  is the beta function.

### Value

Functions `dirichlet()` and `gd()` return a hyperdirichlet object; function `is.dirichlet()` returns a logical; and function `dirichlet_params()` returns a numeric vector.

### Note

These functions have cheaply evaluated analytic expressions for the normalizing constant.

If a hyperdirichlet object corresponds to a Dirichlet distribution, this is relatively easy to detect [using `is.dirichlet()`]. However, the corresponding case for the generalized Dirichlet distribution is not yet coded up, owing to the non-neutrality of the GD.

### Author(s)

Robin K. S. Hankin

### References

- R. J. Connor and J. E. Mosimann 1969. *Concepts of independence for proportions with a generalization of the Dirichlet distribution*. Journal of the American Statistical Association, volume 64, number 325, pp194-206
- T-T Wong 1998. *Generalized Dirichlet distribution in Bayesian analysis*. Applied Mathematics and Computation, volume 97, pp165-181

### See Also

[justpairs](#)

## Examples

```
a <- dirichlet(1:4 , pnames=letters[1:4])
is.dirichlet(a) # should be TRUE
dirichlet(dirichlet_params(a)) # should be 'a'

gd(1:5,5:1)
```

---

hd\_add

*Add two hyperdirichlet distributions*

---

## Description

Given two hyperdirichlet distributions, “add” them, in the sense of concatenating their information, assumed to be independent.

## Usage

```
hd_add(e1, e2, assume_validated = FALSE)
```

## Arguments

`e1, e2` Hyperdirichlet distributions of the same dimension

`assume_validated` Boolean, with default FALSE meaning that the returned sum cannot be assumed to be proper; so the returned hyperdirichlet object is tested with `is.proper()` (which is time-consuming). Set to TRUE only when you *know* that the sum is proper

## Details

Think of this function as a computerized embodiment of Bayes’s theorem with `e1` representing the prior and `e2` representing one or more informative trials.

The basic guts of the function is `hyperdirichlet(powers(e1) + params(e2))`. Note that this is equivalent to `hyperdirichlet(params(e1) + powers(e2))`.

The functional form is not really intended for the end user; use `e1 + e2` instead (but observe that the sum will be validated using `is.proper()`, which may take a long time).

## Value

Returns a hyperdirichlet distribution

## Author(s)

Robin K. S. Hankin

**See Also**

[hyperdirichlet](#), [is.proper](#), [Arith](#)

**Examples**

```
dirichlet(1:4) + gd(c(0.1 , 0.3 , 0.5), c(0.2 , 0.4 , 0.9))

uniform(4) + single_bernoulli_obs(4,1,2)

data(chess)
ch <- as.hyperdirichlet(chess)
stopifnot(all(params(ch+ch+ch) == params(ch*3)))
```

---

head

*Head and tail*

---

**Description**

Print the first few, or last few, lines of a hyperdirichlet object

**Usage**

```
## S4 method for signature 'hyperdirichlet'
head(x, n = 6, ...)
## S4 method for signature 'hyperdirichlet'
tail(x, n = 6, ...)
```

**Arguments**

x	object of class hyperdirichlet
n	number of lines to print as per same argument in head() and tail()
...	Further arguments passed to head() or tail()

**Details**

Prints the head or tail of the `binmat()` matrix and then prints the normalizing constant.

**Value**

The functions documented here coerce to a matrix, then return the appropriate rows (as a matrix).

**Author(s)**

Robin K. S. Hankin

**See Also**[binmat](#)**Examples**

```
head(dirichlet(1:6))
```

---

hyperdirichlet	<i>The Hyperdirichlet distribution</i>
----------------	--

---

**Description**

Create, coerce to, or test for an object of class hyperdirichlet

**Usage**

```
hyperdirichlet(x, NC, pnames = character(), validated=FALSE)
is.hyperdirichlet(x)
as.hyperdirichlet(x, calculate_NC = FALSE , ...)
```

**Arguments**

x	Object to be coerced or tested for
NC	Normalizing constant
pnames	names of the columns with length-0 default resulting in the print method using column names p1, p2, etc
validated	<p>Boolean. Setting to TRUE is taken to mean that x is <b>known</b> to be ‘proper’ (<i>i.e.</i> x is normalizable), but the normalizing constant is not necessarily known.</p> <p>Default FALSE is taken to mean that x is not known to be proper: it is possible that x is not normalizable so cannot correspond to a PDF.</p> <p>Setting to FALSE means that the object will be passed to <code>is.proper()</code> for checking; this can be time-consuming. The flag is set to TRUE <i>ab initio</i> for <code>dirichlet()</code> and <code>gd()</code> because these distributions have an analytical expression for the normalizing constant</p>
calculate_NC	Boolean, with default FALSE meaning not to calculate the normalizing constant and TRUE meaning to calculate it
...	Further arguments passed to <code>adaptIntegrate()</code>

**Details**

To determine the normalization constant, use something like `a <- as.hyperdirichlet(a, calculate_NC=TRUE)`.

Matrices may be coerced to a hyperdichlet object using `as.hyperdirichlet()`: the call is dispatched to `matrix_to_HD()` (`qv`).

If x is a matrix, be sure to specify the `bernoulli` argument, which is passed on to `matrix_to_HD()`

**Value**

Functions `hyperdirichlet()` and `as.hyperdirichlet()` return a `hyperdirichlet` object; function `is.hyperdirichlet()` returns a Boolean.

**Author(s)**

Robin K. S. Hankin

**See Also**

[B,extract,matrix\\_to\\_HD](#)

**Examples**

```
(a <- uniform(3))
a[c(TRUE,TRUE,FALSE)] <- 0.3
## Not run:
(a <- as.hyperdirichlet(a, calculate_NC = TRUE)) # recommended way to calculate NC

## End(Not run)

(b <- dirichlet(1:3))
as.hyperdirichlet(b)          # "forgets" the normalizing constant

## Not run:
as.hyperdirichlet(b, TRUE)    # recalculates NC; accuracy tolerable
                              # (analytic answer = 1/60)

## End(Not run)

## Not run: # takes a long time
op <- options()
options(warn = -1)
x <- dirichlet(rep(2,4)) + justpairs(matrix(1,4,4))
f <- function(p){p[1]>p[2]}
probability(x,f) # should be 0.5: distribution is symmetric

## End(Not run)
```

---

`hyperdirichlet-class` *Class "hyperdirichlet"*

---

**Description**

A generalization of the Dirichlet distribution

## Objects from the Class

Objects can be created by calls of the form `hyperdirichlet(...)`.

## Slots

**.Data:** Numeric vector of length a power of two, holding the parameters (sic; not the powers) of the distribution

**NC:** The normalizing constant, if known

**pnames:** Character vector holding the names of the columns

**validated:** Boolean, indicating whether the distribution is known to be normalizable

## Methods

**NC** signature(object = "hyperdirichlet"): normalizing constant

**params** signature(object = "hyperdirichlet"): accessor method for parameters

**pnames** signature(object = "hyperdirichlet"): accessor method for pnames

**validated** signature(x = "hyperdirichlet"): returns TRUE if the distribution is known to be normalized

**pnames<-** signature(object = "hyperdirichlet"): set method for pnames

**dim** signature(x = "hyperdirichlet"): Dimensionality of the distribution

**mean** signature(object = "hyperdirichlet"): returns the mean of the distribution

**as.hyperdirichlet** signature(object = "hyperdirichlet"): Coerce to hyperdirichlet form

**head** signature(object = "hyperdirichlet"): head method; prints the first few lines of the parameters

**tail** signature(object = "hyperdirichlet"): head method; prints the last few lines of the parameters

**show** signature(object = "hyperdirichlet"): prints the object

## Author(s)

Robin K. S. Hankin; based on `vglm-class.Rd` from the VGAM package

## See Also

[hyperdirichlet](#)

## Examples

```
dput(uniform(3,pnames=letters[1:3]))
```

---

 justpairs

*Hyperdirichlet distribution with nonzero coefficients for just the pairs*


---

**Description**

Hyperdirichlet distribution with nonzero coefficients for terms consisting of a pair of  $ps$ .

**Usage**

justpairs(x)

**Arguments**

x                    A square matrix. Upper triangular elements correspond to pairs of included  $ps$ , and lower triangular elements correspond to pairs of excluded  $ps$

**Details**

The PDF is

$$\prod_{i < j} (p_i + p_j)^{M[i,j]} (1 - p_i - p_j)^{M[j,i]}$$

**Value**

Returns a hyperdirichlet object (without normalizing factor) of dimension `nrow(x)`.

**Note**

The case where `nrow(x)==4` is treated specially. In this case, the lower triangular elements are discarded, with a warning.

**Author(s)**

Robin K. S. Hankin

**See Also**

[paircomp](#)

**Examples**

```
justpairs(matrix(1:25,5,5))
```

matrix\_to\_HD

*Coerce matrices to hyperdirichlet objects***Description**

Coerce matrices to hyperdirichlet objects. These functions are not intended for the user (use `as.hyperdirichlet()` instead).

**Usage**

```
matrix_to_HD(x, calculate_NC = FALSE, bernoulli = NULL, ...)
bernoulli_matrix_to_HD(x, calculate_NC = FALSE, ...)
multinomial_matrix_to_HD(x, calculate_NC = FALSE, ...)
```

**Arguments**

<code>x</code>	Matrix to be coerced
<code>bernoulli</code>	In function <code>matrix_to_HD()</code> , Boolean with TRUE meaning that the matrix rows are to be interpreted as repeated Bernoulli trials and FALSE meaning that they are interpreted as multinomial trials. Default NULL means to use a simple heuristic to infer the desired behaviour
<code>calculate_NC</code>	Boolean, with default FALSE meaning that the normalization constant is not to be calculated
<code>...</code>	Further arguments passed to <code>as.hyperdirichlet()</code> (thence to <code>adapt()</code> )

**Details**

These functions are not intended for the user; use `as.hyperdirichlet()` directly if at all possible.

Function `bernoulli_matrix_to_HD()` operates on rows. Each row has entries corresponding to the columns (the “players”). Each is a Bernoulli trial with three types of entry: NA for not playing, 1 for ‘on the winning side’ and 0 for ‘on the losing side’. Thus the Bernoulli trial is between `which(x==0)` and `which(x==1)`, with the latter winning. A warning is given unless there is at least one 1 and at least one 0 on each row.

Function `multinomial_matrix_to_HD()` also operates on rows. Each row corresponds to a series of restricted multinomial observations with likelihood given by `mult_restricted_obs()` (qv).

**Value**

Returns a hyperdirichlet object

**Author(s)**

Robin K. S. Hankin

**See Also**

[mult\\_restricted\\_obs](#)

**Examples**

```
data.icons)
matrix_to_HD(icons, bern=FALSE)
```

---

maximum_likelihood	<i>Maximum likelihood point for the hyperdirichlet distribution</i>
--------------------	---

---

**Description**

Maximum likelihood point for the hyperdirichlet distribution as estimated using numerical maximization.

**Usage**

```
maximum_likelihood(HD, start_p = NULL, give = FALSE, disallowed = NULL, zero=NULL, ...)
mle(HD, start_p = NULL, give = FALSE, disallowed = NULL, ...)
mle_restricted(HD, start_p = NULL, give = FALSE, disallowed = NULL, zero=NULL, ...)
```

**Arguments**

HD	Object of class hyperdirichlet
start_p	Start value for the ps. See details section
give	Boolean with default FALSE meaning to return just the point estimate and TRUE meaning to return all the output from optim()
disallowed	A function of p returning a Boolean to restrict the search for the MLE. See examples
zero	In function maxlike_restricted(), a Boolean vector with TRUE elements corresponding to components that are constrained to be zero. See details section
...	Further arguments sent to optim()

**Details**

The user should use function maximum\_likelihood(), which is a user-friendly wrapper for one of the two functions (mle() or mle\_rst()) depending on whether argument zero is or is not NULL.

Argument start\_p specifies the start point for the optimization; default NULL is interpreted as rep(1/n, n) where n is dim(HD) (ie neutral position).

It is not necessary to normalize start\_p: this is done by dhyperdirichlet().

Non-default values for this argument are interpreted by dhyperdirichlet().

Argument zero, if not default NULL, is Boolean in the standard case; but if it is not Boolean, it is interpreted as a numeric vector of integers indicating which components of the distribution are restricted to zero. An example is given below.

**Value**

Returns a  $k$ -tuple.

**Note**

The functions minimize `-dhyperdirichlet(..., log=TRUE)`; so there is no need to set `fnscale`.  
Be aware that the **aylmer** package includes a function `maxlike()`, which does something different.

**Author(s)**

Robin K. S. Hankin

**See Also**

[dhyperdirichlet\\_p\\_optim](#)

**Examples**

```
maximum_likelihood(dirichlet(1:4))      # Should be 0:3

jj.numerical <- maximum_likelihood(dirichlet(3:8), zero=2:3)$MLE

jj <- c(2,0,0,5,6,7)
jj.analytical <- jj/sum(jj)

jj.numerical - jj.analytical # should be small
```

---

maxmult

*Estimation of the multivariate beta distribution*


---

**Description**

Gives a maximum likelihood estimate for the parameters of a Dirichlet distribution, on the basis of datapoints drawn from a multivariate beta distribution

**Usage**

```
maxmult(M, start_a = NULL, give = FALSE, method = "nlm", ...)
```

**Arguments**

M	Integer matrix whose rows give multinomial observations
start_a	Start point for optimization, with default NULL being interpreted as Mosimann's formula 29
give	Boolean, with default FALSE meaning to return just the point estimate and TRUE meaning to return all the output from the optimization routine
method	Text string specifying the optimization routine to use. Two values coded: default <code>nlm</code> means to use <code>nlm()</code> and <code>optim</code> meaning to use <code>optim()</code> ; anything else means to return Mosimann's estimate (equation 29)
...	Further arguments passed to <code>nlm()</code> or <code>optim()</code>

**Details**

Finds the maximum likelihood estimate from the equation 7 of Mosimann 1962.

**Note**

The `nlm()` function appears to be better suited to this problem than `optim()`.

**Author(s)**

Robin K. S. Hankin

**References**

J. E. Mosimann 1962. "On the compound multinomial distribution, the multivariate  $\beta$ -distribution, and correlations among proportions". *Biometrika*, volume 49, numbers 1 and 2, pp65-82.

**See Also**

[pollen](#)

**Examples**

```
data(pollen)
maxmult(pollen, start_a=c(51.81, 0.987, 5.332, 1.961))
```

---

paircomp

*Hyperdirichlet distribution for repeated Bernoulli trials*

---

**Description**

Hyperdirichlet distribution with nonzero coefficients for terms consisting of a pair of  $ps$ .

**Usage**

```
paircomp(x)
```

**Arguments**

`x` A square matrix. If players  $i$  and  $j$  play, with  $i < j$ , then player  $j$  wins  $x[i, j]$  trials and player  $i$  wins  $x[j, i]$  trials

**Details**

The PDF is

$$\propto \prod_{i < j} \frac{p_i^{M[j,i]} p_j^{M[i,j]}}{(p_i + p_j)^{M[i,j] + M[j,i]}}$$

**Value**

Returns a hyperdirichlet object (without normalizing factor) of dimension `nrow(x)`.

**Note**

Elements on the diagonal of `x` are silently ignored.

**Author(s)**

Robin K. S. Hankin

**See Also**

[justpairs](#)

**Examples**

```
paircomp(matrix(1:9,6,6))
```

---

params

*Parameters of the hyperdirichlet distribution*

---

**Description**

Parameters of the hyperdirichlet distribution

**Usage**

```
params(x)  
powers(x)
```

**Arguments**

`x`                    Object of class hyperdirichlet

**Details**

Function `powers()` lists the powers of the various combinations; function `params()` lists the parameters.

The two functions differ only in single-p combinations (by one).

**Value**

These functions return a vector of length  $2^{\dim(x)}$ .

**Note**

The default print method gives powers and parameters.

**Examples**

```
a <- gd(1:3,3:1)
params(a)
powers(a)
```

---

paulino

*Dataset used by Paulino*

---

**Description**

A hyperdirichlet distribution for the dataset considered by Paulino and de Bragança Pereira (1995).

**Usage**

```
data(paulino)
```

**Details**

Paulino and de Bragança Pereira considered 97 subjects who each were one of three risk types: “low”, “medium”, and “high” (or 1, 2, 3).

Of the 97 subjects, 51 were fully categorized: 14 were type 1, 17 type 2, and 20 type 3; the remaining 46 were partly categorized: 28 were of types 1 or 2, and 18 were of types 2 or 3.

The likelihood function is thus

$$p_1^{14} p_2^{17} p_3^{20} (p_1 + p_2)^{28} (p_2 + p_3)^{18}$$

and object `paulino` gives the PDF.

**Note** the object gives the *likelihood*, not the posterior PDF. Paulino and de Bragança Pereira use prior information of a type not readily representable in the hyperdirichlet paradigm.

**Source**

C. D. M. Paulino and C. A. de Bragança Pereira 1995. *Bayesian Methods for Categorical Data Under Informative General Censoring*, *Biometrika*, volume 82, number 2, pages 439-446

**Examples**

```
data(paulino)
maximum_likelihood(paulino)
```

---

pnames	<i>Names for the columns</i>
--------	------------------------------

---

**Description**

Get and set column names

**Usage**

```
pnames(x)
pnames(x) <- value
```

**Arguments**

x	Object of class hyperdirichlet
value	Names of the columns

**Value**

Function pnames() returns a  $k$ -tuple of names and function pnames<-() returns a hyperdirichlet object.

**Note**

The default is to leave the columns unnamed; the print method puts them there for printing if null.

**Author(s)**

Robin K. S. Hankin

**Examples**

```
a <- dirichlet(1:3)
pnames(a) <- letters[1:3]
a
```

---

pollen	<i>Pollen data from Mosimann 1962</i>
--------	---------------------------------------

---

**Description**

Data from Mosimann 1962 detailing forest pollen counts

**Usage**

```
data(pollen)
```

**Format**

A matrix with four columns and 76 rows.

**Details**

The rows each sum to 100; the values are counts of four different types of pollen. Each row corresponds to a different level in the core; the levels are in sequence with the first row being most recent and the last row being the oldest.

**References**

J. E. Mosimann 1962. "On the compound multinomial distribution, the multivariate  $\beta$ -distribution, and correlations among proportions". *Biometrika*, volume 49, numbers 1 and 2, pp65-82.

**See Also**

[serum](#)

**Examples**

```
data(pollen)

func <- function(x,l){
  ifelse(any(l<0),Inf,
    lfactorial(sum(x)) -sum(lfactorial(x))
    +lgamma(sum(l)) +sum(lgamma (x+1))
    -sum(lgamma(l)) -lgamma(sum (x+1))
  )
}

start_vec <- c(51.6, 1, 5.3 , 2)

optim(start_vec , function(l){ -sum(apply(pollen , 1, FUN=func,l=1))})
```

---

 print

---

*Print and various S4 methods for hyperdirichlet objects*


---

**Description**

Print method for hyperdirichlet objects

**Usage**

```
print.hyperdirichlet(x, n = 0, do_head = TRUE, ...)
## S4 method for signature 'hyperdirichlet'
dim(x)
## S4 method for signature 'hyperdirichlet'
show(object)
```

**Arguments**

x, object	Object of class hyperdirichlet
n	Integer passed to base head() or tail()
do_head	Boolean argument ignored if n takes its default value of 0. If n<>0, argument do_head taking default TRUE means to pass control to head(x,n,...) and FALSE means to pass control to tail(x,n,...)
...	Further arguments passed to head() or tail() as necessary

**Details**

The matrix itself is printed using internal functions .print\_hyperdirichlet() and .print\_hyperdirichlet\_worker(), which are not intended for the end user.

**Author(s)**

Robin K. S. Hankin

**See Also**

[binmat](#)

**Examples**

```
gd(1:3, 3:1)
```

---

serum

*Serum data from Mosimann 1962*

---

**Description**

Data from Mosimann 1962 detailing serum proteins in white Pekin ducklings.

**Usage**

```
data(serum)
```

**Format**

A matrix with three columns and 23 rows.

**Details**

The rows each sum to 1; the values are the relative frequencies of serum proteins in white Pekin ducklings as determined by electrophores.

**References**

J. E. Mosimann 1962. "On the compound multinomial distribution, the multivariate  $\beta$ -distribution, and correlations among proportions". *Biometrika*, volume 49, numbers 1 and 2, pp65-82.

**Examples**

```
data(serum)
dhyperdirichlet(serum , HD=dirichlet(4:6),log=TRUE)

f <- function(x){
  if(any(x<0)){
    return(Inf)
  } else {
    return(-sum(dhyperdirichlet(serum , HD=dirichlet(x),include.NC=TRUE,log=TRUE)))
  }
}

fish <- optim(c(3.21,20.38,21.68),f)
triplot(HD=dirichlet(fish$par),l=60,main="increase 'l' for better plot quality")
points(serum[,1]+serum[,2]/2,serum[,2]*sqrt(3)/2,pch=16)
```

---

triplot

*Plot density of a 3D hyperdirichlet distribution*


---

**Description**

Plot (log) density of a 3D hyperdirichlet distribution

**Usage**

```
triplot(HD, l = 100, do_image=TRUE, do_contour=TRUE, discard = 0.05, labels=NULL, ...)
```

**Arguments**

HD	Either a function of a three-element vector, or a hyperdirichlet object
l	Size of plot; larger values look much better but take longer
do_image, do_contour	Boolean, indicating whether to plot the image and contour respectively
discard	Numeric; default 0.05 means to disregard densities less than the fifth percentile ( <i>i.e.</i> quantile 0.05). This makes the contour plot intervals prettier

labels	Length three character vector for corner labels with default NULL meaning to use pnames(HD)
...	Further arguments passed to <code>image()</code> and <code>contour()</code>

### Details

If argument HD is not a function it is interpreted as a hyperdirichlet object. If HD is a function, `triplot()` will pass it a three-element vector.

If argument HD is a hyperdirichlet object, then `triplot()` plots contours of the log of the density (ie support). If you want likelihood instead, see examples section for how to do this.

Labels will be printed at the appropriate corners; use `par(xpd=TRUE)` to suppress clipping.

### Value

The function returns (invisibly) an 1-by-1 matrix holding the support as a function of the two independent variables, with NA outside the domain.

### Author(s)

Robin K. S. Hankin

### Examples

```
a <- dirichlet(1:3)
triplot(a,l=20)

old.par <- par(no.readonly=TRUE);par(xpd=TRUE) # (needed for labels)

pnames(a) <- letters[1:3]
triplot(a,l=33)

f <- function(p){1-p[1]+p[2]+8*p[2]^2*p[3]}
triplot(f,labels=LETTERS[1:3])

g <- function(p){dhyperdirichlet(p, a, log=FALSE)}
triplot(g,l=20)

par(old.par)
```

---

uniform

*A uniform Dirichlet distribution*

---

### Description

A uniform Dirichlet distribution

### Usage

```
uniform(n, pnames=NULL)
```

### Arguments

n	The dimension of the distribution
pnames	A character vector giving the names of the columns

### Value

Returns a hyperdirichlet object corresponding to a uniform distribution

### Author(s)

Robin K. S. Hankin

### See Also

[dirichlet](#)

### Examples

```
uniform(4)

uniform(4) + dirichlet(1:4)  # identical (but the normalizing constant is lost)

# Two ways to do the same thing:
rhyperdirichlet(n=11, uniform(4))
t(replicate(11,diff(c(0,sort(runif(3)),1))))
```

---

`volleyball`*Results from the NOCS volleyball league*

---

### Description

Results from the NOCS volleyball league. Object `volleyball_results` is a matrix in which each column corresponds to a player and each row corresponds to a volleyball set; `vb` is the corresponding likelihood function in the form of a hyperdirichlet distribution.

Object `vb_synthetic` is a hyperdirichlet object corresponding to a synthetic dataset obtained from 4000 simulated volleyball sets.

### Usage

```
data(volleyball)
```

### Details

A volleyball set is a Bernoulli trial between two disjoint subsets of the players. The two subsets are denoted (after the game) as the “winners” and the “losers”: these are denoted by 1 and 0 respectively.

Thus the first line reads of `volleyball_results` reads:

```
p1 p2 p3 p4 p5 p6 p7 p8 p9
1  0 NA  1  0  0 NA  1 NA
```

showing that the teams were p1, p4 and p8 against p2, p5 and p6; players p3, p7 and p9 did not play.

Dataset `vb_synthetic` is the likelihood function of 4000 simulated trials in which the skills are distributed according to Zipf’s law:  $1/(1:9)/\text{sum}(1/(1:9))$ .

These datasets illustrate the fact that such Bernoulli trials are only weakly informative. The synthetic dataset involves 4000 observations because this was about the minimum number for which one could estimate the probabilities reasonably reliably. Even then, the lowest probabilities are poorly identified.

### Source

Volleyball games at NOCS, 2006-2008

### References

Robin K. S. Hankin (2010). “A Generalization of the Dirichlet Distribution”, *Journal of Statistical Software*, 33(11), 1-18, <http://www.jstatsoft.org/v33/i11/>

**Examples**

```
data(volleyball)
maximum_likelihood(vb , start_p = c(0.407, 0.091, 0.432, 1.73e-05,
2.24e-08, 1.9e-05, 1.8e-07, 0.03, 0.039) , control=list(maxit=100))

zipf <- 1/seq_len(9)
maximum_likelihood(vb_synthetic , start_p=zipf, control=list(maxit=100))
```

# Index

- \*Topic **classes**
  - hyperdirichlet-class, [22](#)
- \*Topic **datasets**
  - doubles, [12](#)
  - paulino, [30](#)
  - pollen, [31](#)
  - serum, [33](#)
  - volleyball, [37](#)
- \*Topic **math**
  - Arith-methods, [3](#)
  - binmat, [8](#)
  - Extract.hyperdirichlet, [13](#)
  - gd, [17](#)
  - justpairs, [24](#)
  - paircomp, [28](#)
- \*Topic **methods**
  - Arith-methods, [3](#)
- \*Topic **package**
  - hyperdirichlet-package, [2](#)
- [,hyperdirichlet-method
  - (Extract.hyperdirichlet), [13](#)
- [.hyperdirichlet
  - (Extract.hyperdirichlet), [13](#)
- [<-,hyperdirichlet-method
  - (Extract.hyperdirichlet), [13](#)
- [<-.hyperdirichlet
  - (Extract.hyperdirichlet), [13](#)
- Arith, [20](#)
- Arith,ANY,hyperdirichlet-method
  - (Arith-methods), [3](#)
- Arith,hyperdirichlet,ANY-method
  - (Arith-methods), [3](#)
- Arith,hyperdirichlet,hyperdirichlet-method
  - (Arith-methods), [3](#)
- Arith,hyperdirichlet,missing-method
  - (Arith-methods), [3](#)
- Arith,hyperdirichlet,numeric-method
  - (Arith-methods), [3](#)
- Arith,numeric,hyperdirichlet-method
  - (Arith-methods), [3](#)
- Arith-methods, [3](#)
- as.hyperdirichlet (hyperdirichlet), [21](#)
- as.hyperdirichlet,hyperdirichlet-method
  - (hyperdirichlet), [21](#)
- as.hyperdirichlet,matrix-method
  - (hyperdirichlet), [21](#)
- as.hyperdirichlet,numeric-method
  - (hyperdirichlet), [21](#)
- B, [4](#), [22](#)
- bernoulli, [6](#)
- bernoulli\_matrix\_to\_HD (matrix\_to\_HD),  
[25](#)
- bernoulli\_obs (bernoulli), [6](#)
- binmat, [8](#), [21](#), [33](#)
- calculate\_B (B), [4](#)
- dhyperdirichlet, [9](#), [12](#), [15](#)
- dhyperdirichlet\_e (dhyperdirichlet), [9](#)
- dhyperdirichlet\_p, [27](#)
- dhyperdirichlet\_p (dhyperdirichlet), [9](#)
- dim (print), [32](#)
- dim,hyperdirichlet-method (print), [32](#)
- diri\_norm, [11](#)
- Dirichlet (gd), [17](#)
- dirichlet, [36](#)
- dirichlet (gd), [17](#)
- dirichlet\_params (gd), [17](#)
- dirichlet\_params<- (gd), [17](#)
- doubles, [12](#)
- doubles\_noghost (doubles), [12](#)
- e\_to\_p, [10](#), [14](#)
- extract, [22](#)
- extract (Extract.hyperdirichlet), [13](#)
- Extract.hyperdirichlet, [13](#)
- fitz, [16](#)

- gd, 17
- gd\_norm (diri\_norm), 11
- Generalized Dirichlet (gd), 17
- hd\_add, 4, 19
- head, 20
- head, hyperdirichlet-method (head), 20
- hyperdirichlet, 6, 20, 21, 23
- hyperdirichlet-class, 22
- hyperdirichlet-package, 2
- is.dirichlet (gd), 17
- is.hyperdirichlet (hyperdirichlet), 21
- is.hyperdirichlet, hyperdirichlet-method (hyperdirichlet), 21
- is.proper, 20
- is.proper (B), 4
- isvalidated (B), 4
- Jacobian (e\_to\_p), 14
- justpairs, 18, 24, 29
- matrix\_to\_HD, 22, 25
- maximum.likelihood, 10
- maximum.likelihood (maximum\_likelihood), 26
- maximum\_likelihood, 26
- maxmult, 27
- mean (B), 4
- mean, hyperdirichlet-method (B), 4
- mgf (B), 4
- mle (maximum\_likelihood), 26
- mle\_restricted (maximum\_likelihood), 26
- mult\_bernoulli\_obs (bernoulli), 6
- mult\_restricted\_obs, 25
- mult\_restricted\_obs (bernoulli), 6
- multinomial\_matrix\_to\_HD (matrix\_to\_HD), 25
- NC (B), 4
- NC, hyperdirichlet-method (B), 4
- obs (bernoulli), 6
- optim, 27
- p\_to\_e (e\_to\_p), 14
- paircomp, 24, 28
- params, 29
- params, hyperdirichlet-method (params), 29
- Paulino (paulino), 30
- paulino, 30
- pnames, 31
- pnames, hyperdirichlet-method (pnames), 31
- pnames<- (pnames), 31
- pnames<- , hyperdirichlet-method (pnames), 31
- pollen, 28, 31
- powers (params), 29
- print, 32
- print.hyperdirichlet, 9
- print.hyperdirichlet (print), 32
- probability (B), 4
- rhyperdirichlet (dhyperdirichlet), 9
- serum, 32, 33
- show (print), 32
- show, hyperdirichlet-method (print), 32
- single\_bernoulli\_obs (bernoulli), 6
- single\_multi\_restricted\_obs (bernoulli), 6
- single\_obs (bernoulli), 6
- tail, hyperdirichlet-method (head), 20
- triplet, 34
- uniform, 36
- validated (B), 4
- validated, hyperdirichlet-method (B), 4
- vb (volleyball), 37
- vb\_synthetic (volleyball), 37
- volleyball, 37
- volleyball\_results (volleyball), 37