Package ‘hyper2’

May 16, 2019

Type Package
Title The Hyperdirichlet Distribution, Mark 2
Version 1.0-7
Maintainer Robin K. S. Hankin <hankin.robin@gmail.com>
Description A suite of routines for the hyperdirichlet distribution; supersedes the 'hyperdirichlet' package.
License GPL (>= 2)
LazyData yes
Depends R (>= 2.10), methods, magrittr, cubature
Suggests knitr
VignetteBuilder knitr
Imports Rcpp (>= 0.12.5), partitions
LinkingTo Rcpp
URL https://github.com/RobinHankin/hyper2.git
BugReports https://github.com/RobinHankin/hyper2/issues
NeedsCompilation yes
Author Robin K. S. Hankin [aut, cre] (<https://orcid.org/0000-0001-5982-0415>)
Repository CRAN
Date/Publication 2019-05-16 07:30:03 UTC

R topics documented:

hyper2-package ......................................................... 2
B ................................................................................. 4
change_pnames .......................................................... 6
character_to_number .................................................... 8
chess ........................................................................ 9
counterstrike ............................................................... 10
cplusplus ................................................................. 11
The Hyperdirichlet Distribution, Mark 2

Description

A suite of routines for the hyperdirichlet distribution; supersedes the 'hyperdirichlet' package.

Details

The DESCRIPTION file:

Package: hyper2
Type: Package
Title: The Hyperdirichlet Distribution, Mark 2
Version: 1.0-7
Authors@R: person(given=c("Robin", "K. S."), family="Hankin", role = c("aut","cre"), email="hankin.robin@gmail.com")
Maintainer: Robin K. S. Hankin <hankin.robin@gmail.com>
Description: A suite of routines for the hyperdirichlet distribution; supersedes the 'hyperdirichlet' package.
License: GPL (>= 2)
LazyData: yes
Depends: methods, magrittr, cubature
Suggests: knitr
VignetteBuilder: knitr
Imports: Rcpp (>= 0.12.5), partitions
LinkingTo: Rcpp
URL: https://github.com/RobinHankin/hyper2.git
BugReports: https://github.com/RobinHankin/hyper2/issues
Author: Robin K. S. Hankin [aut, cre] (<https://orcid.org/0000-0001-5982-0415>)

Index of help topics:

B Normalizing constant for the hyperdirichlet distribution
Extract.hyper2 Extract or replace parts of a hyper2 object
NBA Basketball dataset
Ops.hyper2 Arithmetic Ops Group Methods for hyper2 objects
Print Print methods
change_pnames Change pnames of a hyper2 object
character_to_number Convert a character vector to a numeric vector
chess Chess playing dataset
counterstrike Counterstrike
cplusplus Wrappers to c calls
dirichlet Dirichlet distribution and generalizations
euro Eurovision Song contest dataset
fillup Fillup function
formula1 Formula 1 dataset
ggol Order statistics
gradient Differential calculus
handover Dataset on communication breakdown in handover between physicians
head.hyper2 First few terms of a distribution
hyper2 Basic functions in the hyper2 package
hyper2-package The Hyperdirichlet Distribution, Mark 2
icons Dataset on climate change due to O'Neill
increment Increment and decrement operators
interzonal 1963 World Chess Championships
karpov_kasparov_anand Karpov, Kasparov, Anand
length.hyper2 Length method for hyper2 objects
loglik Log likelihood functions
masterchef Masterchef series 6
maxp Maximum likelihood estimation
mult_grid  Kronecker matrix functionality
rowing  Rowing dataset, sculling
rrank  Random ranks
rugby  Super Rugby 2016 data
saffy  Choice matrix functionality
skating  Figure skating at the 2002 Winter Olympics
table_tennis  Match outcomes from repeated table tennis matches
tennis  Match outcomes from repeated doubles tennis matches
tidy  Keep or discard players
volley  Team sports

A generalization of the Dirichlet distribution, using a more computationally efficient method than the hyperdirichlet package. The software is designed for the analysis of order statistics and team games.

Author(s)

NA
Maintainer: Robin K. S. Hankin <hankin.robin@gmail.com>

References


Examples

H <- hyper2(list(1,2,3,1:2,2:3),1:5)
data(chess)
maxp(chess)

B  Normalizing constant for the hyperdirichlet distribution

Description

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

\[ B(H, \text{disallowed=\texttt{NULL}}, \text{give=\texttt{FALSE}}, \ldots) \]
\[ \text{probability}(H, \text{disallowed=\texttt{NULL}}, \ldots) \]
\[ \text{mgf}(H, \text{powers}, \ldots) \]
\[ \text{dhyper2}(P,H,\ldots) \]
\[ \text{dhyper2.e}(e,H,\text{include.Jacobian=\texttt{TRUE}}) \]
\[ \text{mean_hyper2}(H, \text{normalize=\texttt{TRUE}}, \ldots) \]
\[ \text{Jacobian}(e) \]
\[ \text{e_to_p}(e) \]
\[ \text{p_to_e}(p) \]

Arguments

\[ H \quad \text{Object of class \texttt{hyper2}} \]
\[ \text{powers} \quad \text{Vector of length } \dim(x) \text{ whose elements are the powers of the expectation; see details section} \]
\[ \text{disallowed} \quad \text{Function specifying a subset of the simplex over which to integrate; default } \texttt{NULL} \text{ means to integrate over the whole simplex. The integration proceeds over } p \text{ with disallowed}(p) \text{ evaluating to } \texttt{FALSE} \]
\[ e,p \quad \text{Vectors; see details} \]
\[ P \quad \text{A vector of probabilities with unit rowsums} \]
\[ \text{include.Jacobian} \quad \text{Boolean, indicated whether or not to include the Jacobian transformation in the evaluation} \]
\[ \text{give} \quad \text{Boolean, with default } \texttt{FALSE} \text{ meaning to return the value of the integral and } \texttt{TRUE} \text{ meaning to return the full output of adaptIntegrate()} \]
\[ \text{normalize} \quad \text{Boolean, indicates whether return value of mean_hyper2() is normalized to have unit sum} \]
\[ \ldots \quad \text{Further arguments passed to adaptIntegrate()} \]

Details

- Function \( B() \) returns the normalizing constant; \( \text{dhyper2}(p,H)/B(H) \) is a probability density function. Internally, \( p \) is converted to \( e \) (by \( \text{e_to_p}(e) \)) and the integral proceeds over a hypercube.
- Function \( \text{probability}() \) gives the probability of an observation from a hyperdirichlet distribution satisfying \( !\text{disallowed}(p) \).
- Function \( \text{mgf}() \) is the moment generating function, taking an argument that specifies the powers of \( p \) needed: the expectation of \( \prod_{i=1}^{d} p[i]^{\text{powers}[i]} \) is returned.
- Function \( \text{mean_hyper2()} \) returns the mean value of the hyperdirichlet distribution. This is computationally slow (consider \( \text{maxp()} \) for a measure of central tendency). The function takes a normalize argument, not passed to adaptIntegrate(): this is Boolean with \( \texttt{FALSE} \) meaning to return the value found by integration directly, and default \( \texttt{TRUE} \) meaning to normalize so the sum is exactly 1.
Value

- Function \( B() \) returns a scalar: the normalization constant
- Function \( \text{mean}() \) returns a \( k \)-tuple with unit sum
- Function \( \text{mgf}() \) returns a scalar equal to the expectation of \( p^\text{power} \)
- Functions \( \text{is.proper}() \) and \( \text{validated}() \) return a Boolean
- Function \( \text{probability}() \) returns a scalar, a (Bayesian) probability

Note

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

Author(s)

Robin K. S. Hankin

See Also

- \texttt{loglik}

Examples

```r
data(chess)
mean_hyper(chess)
maxp(chess)

# disallowed argument typically means slow run times; use high tol for speed
probability(chess, disallowed=function(p){p[1]>p[2]}, tol=0.1)
probability(chess, disallowed=function(p){p[1]<p[2]}, tol=0.1)
```

---

**change_pnames**

*Change pnames of a hyper2 object*

Description

Change the pnames attribute of a hyper2 object

Usage

```r
change_pnames(H, new_pnames)
all_pnames(L)
```
change_pnames

Arguments

H Object of class hyper2
L List of hyper2 objects
new_pnames Character vector of the new pnames

Note

Function change_pnames() is not a straightforward function. The intent is to enable one to add hyper2 objects with non-identical pnames attributes in a sensible way.

The standard use for change_names() is to add new players to a hyper2 object: so the new names must be a superset of the old, although the order may differ.

Function all_pnames() returns the union of the pnames of the hyper2 objects sent to it

Author(s)

Robin K. S. Hankin

Examples

x1 <- hyper2(list(1,1:2,1:3),1:3,pnames=letters[1:3])
x2 <- x1
pnames(x2) <- letters[3:1]

x1
x2
# At this point, note that x1 and x2 are algebraically identical; but:
x1==x2

## Variables x1 and x2 are different! This is by design. The pnames
## attribute is just syntatic sugar, mathematically meaningless.

## Also note that you cannot add x1 and x2 because to do so is bad weirdness:
## Not run:
x1 + x2

## End(Not run)

## Here is a sensible use-case:

x1 <- hyper2(list(1,1:2,1:3),1:3,pnames=letters[1:3])
x2 <- hyper2(list(1,2),1:11,pnames=letters[4:5])
pnames(x1) <- letters[1:5]
pnames(x2) <- letters[1:5]
x1+x2

## NB:
character_to_number

Convert a character vector to a numeric vector

Description

Convert string descriptions of competitors into their number

Usage

character_to_number(char, pnames)
char2num(char, pnames)

Arguments

char Character vector to be converted
pnames Names vector (usually pnames(H))

Details

The internal mechanism of functions such as ggol(), and all the C++ code, operates with the competitors labelled with a non-negative integer; it is then natural to refer to the competitors as p1, p2, etc.

However, sometimes the competitors have names (as in, for example, the rowing dataset). If so, it is more natural to refer to the competitors using their names rather than an arbitrary integer.

Function character_to_number() converts the names to numbers. If an element of char is not present in pnames, an error is returned.

Function char2num() is an easy-to-type synonym.

Author(s)

Robin K. S. Hankin

See Also

order_likelihood
Examples

```r
H <- hyper2(pnames=letters[1:9])

# Use numbers:
H + order_liklihood(sample(9))

# Use letters:
H + order_liklihood(character_to_number(sample(letters[1:9]), pnames(H)))
```

---

**chess**

*Chess playing dataset*

**Description**

A tally of wins and losses for games between three chess players: Topalov, Anand, Karpov.

**Usage**

data(chess)

**Details**

This is a very simple dataset that can be used for illustration of hyper2 idiom.

The players are:

- Grandmaster Veselin Topalov. FIDE world champion 2005-2006; peak rating 2813
- Grandmaster Anatoly Karpov. FIDE world champion 1993-1999; peak rating 2780

Observe that Topalov beats Anand, Anand beats Karpov, and Karpov beats Topalov (where “beats” means “wins more games than”).

The games thus resemble a noisy version of “rock paper scissors”.

The likelihood function does not record who played white; see karpov_kasparov_anand for such a dataset.

**References**


**See Also**

karpov_kasparov_anand
Examples

```r
data(chess)
maxp(chess)
```

counterstrike  Counterstrike

Description

A kill-by-kill analysis of a counterstrike game.

Usage

```r
data(counterstrike)
```

Details

E-sports are a form of competition using video games. E-sports are becoming increasingly popular, with high-profile tournaments attracting over 400 million viewers, and prize pools exceeding US$20m.

Counter Strike: Global Offensive (CS-GO) is a multiplayer first-person shooter game in which two teams of five compete in an immersive virtual reality combat environment.

CS-GO is distinguished by the ability to download detailed game files in which every aspect of an entire match is recorded, it being possible to replay the match at will.

Statistical analysis of such game files is extremely difficult, primarily due to complex gameplay features such as cooperative teamwork, within-team communication, and real-time strategic fluidity.

It is the task of the statistician to make robust inferences from such complex datasets, and here I analyze an influential match between “FaZe Clan” and “Cloud9”, two of the most successful E-sports syndicates of all time, when they competed at Boston 2018.

Dataset `counterstrike` is a loglikelihood function for the strengths of ten counterstrike players. It can be created by executing the code in `inst/counterstrike.R`, which also shows how to create synthetic datasets for the purposes of comparison.

The probability model is identical to that of `nba`: when a player kills (scores), this is taken to be a success of the whole team rather than the shooter.

The `counterstrike` object is created in `inst/counterstrike.R` (but is called “H” there, to avoid terrible confusion). There is quite a lot of extra documentation in that file.

Counterstrike dataset kindly supplied by Zachary Hankin.

References

- [https://www.youtube.com/watch?v=XKWzlG4jDnI](https://www.youtube.com/watch?v=XKWzlG4jDnI)
Examples

data(counterstrike)
dotchart(maxp(counterstrike))

Description

Various low-level wrappers to C calls, courtesy of Rcpp

Usage

overwrite(L1, powers1, L2, powers2)
accessor(L, powers, Lwanted)
assigner(L, p, L2, value)
add(L1, p1, L2, p2)
identityL(L, p)
evaluate(L, powers, probs)
differentiate(L, powers, probs, n)

Arguments

L, L1, L2, Lwanted
   Lists with integer vector elements, used to specify the brackets of the hyper-
dirichlet distribution
p, p1, p2, powers, powers1, powers2
   A numeric vector specifying the powers to which the brackets are raised
value
   RHS in assignment, a numeric vector
probs
   Vector of probabilities for evaluation of log-likelihood
n
Details

These functions are not really intended for the end-user, as out-of-scope calls may cause crashes.

Value

These functions return a named List

Author(s)

Robin K. S. Hankin
**dirichlet**

*Dirichlet distribution and generalizations*

**Description**

The Dirichlet distribution in likelihood (for p) form, including the generalized Dirichlet distribution due to Connor and Mosimann

**Usage**

```r
dirichlet(powers, alpha, pnames=NA)
GD(alpha, beta, beta0=0, pnames=NA)
GD_wong(alpha, beta, pnames=NA)
```

**Arguments**

- `powers`: In function `dirichlet()` a (named) vector of powers
- `alpha, beta`: A vector of parameters for the Dirichlet or generalized Dirichlet distribution
- `beta0`: In function `GD()`, an arbitrary parameter
- `pnames`: Optional; a character vector specifying the names of $p_1$ through $p_n$

**Details**

These functions are really convenience functions.

**Author(s)**

Robin K. S. Hankin

**See Also**

`hyper2`

**Examples**

```r
x1 <- dirichlet(c(1, b=2, c=3))
x2 <- dirichlet(c(3, d=4))

# will not work: x1+x2  ## probably not what you wanted
```
Eurovision Song contest dataset

Description

Voting patterns from Eurovision 2009

Usage

data(euro)

Format

A hyper2 object that gives a likelihood function.

Details

Object euro2009 is a hyper2 object that gives a likelihood function for the skills of the 18 competitor countries in semi-final 1 of the 2009 Eurovision Song contest. It is created by the self-contained code in inst/euro.R, which is heavily documented.

The reason for choosing this particular dataset is that Pat Altham (Statistical Laboratory, Cambridge) considered it with a view to discover similarities between voters. In the current analysis, the likelihood function euro2009 assumes their independence.

References

• P. M. E. Altham, personal communication

Examples

data(euro)

dotchart(maxp(euro2009))

x1 <- loglik(euro2009, indep(maxp(euro2009)))
x2 <- loglik(euro2009, indep(equalp(euro2009)))

## if the competitors have equal strength, the asymptotic distribution
## of 2*(x1-x2) is chi-square with 17 degrees of freedom:
pchisq(2*(x1-x2), df=size(euro2009)-1, lower.tail=FALSE)
### Description

Extract or replace parts of a hyper2 object

### Usage

```r
## S3 method for class 'hyper2'
x[...]
## S3 replacement method for class 'hyper2'
x[index, ...] <- value
```

### Arguments

- **x**: An object of class hyper2
- **...**: Further arguments, currently ignored
- **index**: A list with integer vector elements corresponding to the brackets whose power is to be replaced
- **value**: Numeric vector of powers

### Details

These methods should work as expected, although the off-by-one issue might be a gotcha.

For the extract method, a numeric vector is returned but the elements may be in any order (the brackets and indeed the powers are not stored in any particular order).

The replace method, \( H[L] \) \( \leftarrow \) \( value \), the index \( L \) is a list specifying the brackets to be overwritten; argument \( value \) is not recycled unless it is of length 1.

If the index argument is missing, viz \( H[] \) \( \leftarrow \) \( H2 \), this is a special case. Argument \( H1 \) must be a hyper2 object, and the idiom effectively executes \( H1[\text{brackets}(H2)] \) \( \leftarrow \) \( \text{powers}(H2) \), but more efficiently (note that this operation is well-defined even though the order of the brackets is arbitrary). This special case is included in the package because it has a very natural C++ expression [function \( \text{overwrite}() \) in the src/ directory] that was too neat to omit.

Altering (incrementing or decrementing) the power of a single bracket is possible using idiom like \( H[x] \) \( \leftarrow \) \( H[x] + 1 \); this is documented at \( \text{Ops.hyper2} \), specifically \( \text{hyper2_sum_numeric()} \).

### Value

The extractor method returns a hyper2 object, restricted to the elements specified
**Note**

Use `powers()` and `brackets()` to extract a numeric vector of powers or a list of integer vectors respectively.

Replacement idiom `h[x] <- val` cannot use non-trivial recycling. This is because the elements of `h` are stored in an arbitrary order, but the elements of `val` are stored in a particular order. Also see function `hyper2_sum_numeric()`.

**Author(s)**

Robin K. S. Hankin

**See Also**

hyper2,ops.hyper2

**Examples**

data(chess)

chess[1]
chess[c(1,2)]
chess[List(1,1:2)]
chess[1] <- 5

a <- hyper2(pnames=letters[1:8])
a[sapply(seq_len(8),seq_len)] <- 1
a

b <- a
b[1] <- 1000
b[] <- a # b[1] overwritten; b[2] preserved

---

**fillup**  
*Fillup function*

**Description**

Function `fillup()` concatenates a vector with a ‘fillup’ value to ensure a unit sum; if given a matrix, attaches a column so the rowsums are 1.

Function `indep()` is the inverse: it removes the final element of a vector, leaving only an independent set.

**Usage**

`fillup(x)`
`indep(x)`
Arguments

x Numeric vector

Author(s)

Robin K. S. Hankin

See Also

equalp

Examples

fillup(c(1/2,1/3))

indep(c(1/2,1/3,1/6))

---

**formula1**

*Formula 1 dataset*

Description

Race results from 2017 Formula One World Championship

Usage

data(formula1)

Format

A hyper2 object that gives a likelihood function

Details

Object formula1 is a hyper2 object that gives a likelihood function for the strengths of the competitors of the 2017 Formula One World Championship. It is created from inst/formula1.txt by the code in inst/formula1.R, which is heavily documented.

References


Examples

data(formula1)
dotchart(maxp(formula1))
ggol

Order statistics

Description

Various functions for calculating the likelihood function for order statistics

Usage

ggol(H, ...)
general_grouped_orderlikelihood(H, ...)
choose_losers(H, all_players, losers)
choose_winners(H, all_players, winners)
order_likelihood(M, times=1)

Arguments

H
Object of class hyper2

... Numeric or character vectors specifying groups of players with equal rank, with higher-ranking groups coming earlier in the argument list

all_players, winners, losers Numeric or character vectors specifying (all, winning, losing) competitors in randomly chosen groups

M In function order_likelihood(), a matrix with each row corresponding to a race. The columns correspond to the finishing order; thus a=M[i, j] means that competitor a finished in place j in race i

times Vector specifying the number of times each row is observed

Details

These functions are designed to return likelihood functions, in the form of lists of hyper2() objects, for typical order statistics such as the results of rowing heats or MasterChef tournaments.

Function ggol() is an easily-typed alias for general_grouped_orderlikelihood().

Function addrank() is now defunct.

Author(s)

Robin K. S. Hankin

See Also

rrank
Examples

\begin{verbatim}
W <- hyper2(pnames=letters[1:7])
W1 <- ggol(W, 'a', letters[2:6], 'g')  # 24-element list
W2 <- ggol(W, 'b', letters[3:7], 'a')  # 24-element list
like_single_list(rep(1/7,6),W1)
like_series(rep(1/7,6),list(W1,W2))

# Run 10 races:
r1 <- rrank(10,p=(7:1)/28)
colnames(r1) <- letters[1:7]

# Likelihood function for r1:
W <- order_likelihood(r1)

# Incorporate information from another (independent) observation:
W <- W + order_likelihood(sample(7))

# Consistency check:
H <- hyper2(pnames=letters[1:5])
H <- H + order_likelihood(rrank(100,5:1))  # probs = 5/15,4/15,...,1/15
maxp(H)  # Should be close to (5:1)/15
\end{verbatim}

---

gradient

Differential calculus

Description

The gradient of the log-likelihood of a hyper2 object, at a specific point in probability space

Usage

\[ \text{gradient}(H, \text{probs}) \]

Arguments

- **H**: A hyper2 object
- **probs**: A vector of probabilities

Details

Function `gradient()` returns the gradient of the log-likelihood function. If the hyper2 object is of size \( n \), then argument `probs` must be a vector of length \( n - 1 \).
Note

This functionality is peculiarly susceptible to off-by-one errors.

Author(s)

Robin K. S. Hankin

See Also

differentiate

Examples

data(chess)
p <- c(1/2,1/3)
delta <- rnorm(2)/1e5  # delta needs to be quite small

delta <- loglik(chess,p+delta) - loglik(chess,p)
deltaLn <- sum(delta+gradient(chess,p + delta/2))  # numeric

deltaL - deltaLn  # should be small

handover

Dataset on communication breakdown in handover between physicians

Description

Object handover is a likelihood function corresponding to a dataset arising from 69 medical malpractice claims and concerns handover (or hand-off) between physicians. This dataset was analysed by Lin et al. (2009), and further analysed by Altham and Hankin (2010). The computational methods are presented in the ‘hyperdirichlet’ and ‘aylmer’ packages and a further discussion is given in the “integration” vignette of the ‘hyper2’ package.

Usage

data(handover)

Details

Details are provided in the integration vignette, and the cited papers.
References


Examples

```r
data(handover)
maxp(handover)
```

---

**head.hyper2**  
*First few terms of a distribution*

Description

First few terms in a hyperdirichlet distribution

Usage

```r
## S3 method for class 'hyper2'
head(x, ...)
```

Arguments

- `x` Object of class `hyper2`
- `...` Further arguments, passed to `head()`

Details

Function is `x[head(brackets(x), ...)]`

Value

Returns a `hyper2` object

Author(s)

Robin K. S. Hankin
Examples

```r
H <- order_likelihood(rrank(100, p=(10:1)/55))
head(H)
```

---

**Description**

Basic functions in the hyper2 package

**Usage**

```r
hyper2(L, d, pnames = NA)
## S3 method for class 'hyper2'
brackets(H)
## S3 method for class 'hyper2'
powers(H)
## S3 method for class 'hyper2'
pnames(H)
size(H)
as.hyper2(L, d, pnames)
is_valid_hyper2(L, d, pnames)
is_constant(H)
```

**Arguments**

- `H` A hyper2 object
- `L` A list whose elements specify the brackets of a hyper2 object
- `d` A vector of powers; `hyper2()` recycles only if `d` is of length 1
- `pnames` A character vector specifying the names of \( p_1 \) through \( p_n \).

**Details**

These are the basic functions of the hyper2 package. Function `hyper()` is the low-level creator function; `as.hyper2()` is a bit more user-friendly and attempts to coerce its arguments into a suitable form; for example, a matrix is interpreted as rows of brackets.

Function `is_valid_hyper2()` tests for valid input, returning a Boolean.

Function `size()` returns the (nominal) length \( n \) of nonnegative vector \( p = (p_1, \ldots, p_n) \) where \( p_1 + \cdots + p_n = 1 \). Note that function `lhyper2()` takes the vector \( p = (p_1, \ldots, p_{n-1}) \).

**Author(s)**

Robin K. S. Hankin
See Also

Ops.hyper2, Extract.loglik

Examples

hyper2(list(1,2,3,1:2,2:3),1:5)

icons

Dataset on climate change due to O’Neill

Description

Object icons_matrix is a matrix of nine rows and six columns, one column for each of six icons relevant to climate change. The matrix entries show the number of respondents who indicated which icon they found most concerning. The nine rows show different classes of respondents who were exposed to different subsets (of size four) of the six icons. The columns correspond to the different stimulus icons used, detailed below.

Object icons is the corresponding likelihood function, which can be created with saffy/icons_matrix).

Usage

data(oneill)

Details

The six icons were used in this study were:

PB polar bears, which face extinction through loss of ice floe hunting grounds
NB The Norfolk Broads, which flood due to intense rainfall events
LF London flooding, as a result of sea level rise
THC The Thermo-haline circulation, which may slow or stop as a result of anthropogenic modification of the hydrological cycle
OA Oceanic acidification as a result of anthropogenic emissions of carbon dioxide
WAIS The West Antarctic Ice Sheet, which is calving into the sea as a result of climate change

Source

Data kindly supplied by Saffron O’Neill of the University of East Anglia

References

• S. O’Neill 2007. An Iconic Approach to Communicating Climate Change, University of East Anglia, School of Environmental Science (in prep)
Examples

data(oneill)
maxp(icons)

icons == saffy(icons_matrix)  # should be TRUE

Description

Increment and decrement operators

Syntactic sugar for incrementing and decrementing likelihood functions

Usage

inc(H, val = 1)
dec(H, val = 1)
trial(H, winners, players, val=1)

Arguments

H
A hyper2 object

winners, players
Numeric or character vectors specifying the winning team and the losing team

val
Numeric

Details

A very frequent operation is to increment a single term in a hyper2 object. If

R> H <- hyper2(list(1,2,3,1:2,2:3),1:5)
R> pnames(H) <- letters[1:3]

Then we would have

R> H
a * (a + b)^4 * b^2 * (b + c)^5 * c^3

Suppose we wish to increment the power of a+b. We could do:

H[c("a","b")] <- H[c("a","b")] + 1

Or use magrittr pipes:

H[c("a","b")] %<% `*\`(1)
But inc and dec furnish convenient idiom to accomplish the same thing:

\[ H[c("a","b")]%\rangle%inc \]

Functions inc and dec default to adding or subtracting 1, but other values can be supplied:

\[ H[c("a","b")]%\rangle%inc(3) \]

Or even

\[ H[c("a","b")]%\rangle%inc(H["a"]) \]

The convenience function trial() takes this one step further and increments the ‘winning team’ and decrements the ‘losing team’. This function is intended to be used with magrittr pipes:

\[
\begin{align*}
> H & \leftarrow \text{hyper2}(\text{pnames}=\text{letters}[1:6]) \\
> H & \leftarrow (a + b + c + d + e + f)^0 \\
> H & \%\rangle\% \text{trial}(c("a","b"),c("a","b","c")) \\
> H & \leftarrow (a + b) \times (a + b + c)^{-1}
\end{align*}
\]

Using trial() in this way ensures that the powers sum to zero.

The inc and dec operators are used in inst/rowing_analysis.R; and the trial() function is used in inst/kka_draws.R.

Author(s)

Robin K. S. Hankin

Examples

data(chess)

## Now suppose Topalov beats Anand. To incorporate this into the LF:
chess["Topalov"] %\rangle% inc
chess[c("Topalov","Anand")] %\rangle% dec
chess
**interzonal**  

**1963 World Chess Championships**

**Description**

Likelihood functions for players’ strengths in the fifth Interzonal tournament which occurred as part of the 1963 Chess world Championships in Stockholm, 1962.

**Details**

The 1963 World Chess Championships was notable for allegations of Soviet collusion. Specifically, Fischer publicly alleged that certain Soviet players had agreed in advance to draw all their games.

Likelihood functions interzonal and interzonal_collusion are created by files inst/collusion_stockholm.R, which are heavily documented and include some analysis. Object interzonal includes a term for drawing, (“draw”), assumed to be the same for all players; object collusion includes in addition to draw, a term for the drawing in Soviet-Soviet matches, “coll”.

**See Also**

chess,karpov_kasparov_anand

**Examples**

```r
data(interzonal)
dotchart(maxp(interzonal))
maxp(collusion)
```

**karpov_kasparov_anand  Karpov, Kasparov, Anand**

**Description**

Data of three chess players: Karpov, Kasparov, and Anand. Includes two likelihood functions for the strengths of the players, and matrices of game results

**Details**

The strengths of chess players may be assessed using the generalized Bradley-Terry model. The karpov_kasparov_anand likelihood function allows one to estimate the players’ strengths, propensity to draw, and also the additional strength conferred by playing white.

Likelihood functions karpov_kasparov_karpov, kka_3draws and kka_3whites are created by files inst/karpov_kasparov_anand.R, inst/kka_3draws and inst/kka_3whites, which are heavily documented and include some analysis. Object karpov_kasparov_anand assumes that the draw
potential is the same for all three players; likelihood function \texttt{kka\_draws} allows the propensity to draw to differ between the three players.

The reason that the players are different from those in the \texttt{chess} dataset is that the original data does not seem to be available any more.

Dataset \texttt{kka} refers to scorelines of matches between three chess players (Kasparov, Karpov, Anand). It is a list with names such as \texttt{‘karpov\_plays\_white\_beats\_kasparov’} which has value 18: we have a total of 18 games between Karpov and Kasparov in which Karpov played white and beat Kasparov.

The three matrices \texttt{plays\_white\_wins}, \texttt{plays\_white\_draws}, and \texttt{plays\_white\_loses} tabulate this information in a coherent way; and array \texttt{kka\_array} presents the same information in a 3D array (but the names of the dimnames are lost).

All data drawn from chessgames.com, specifically \url{http://www.chessgames.com/perl/ezsearch.pl?search=karpov+vs+kasparov}

Note that the database allows one to sort by white wins or black wins (there is a ‘refine search’ tab at the bottom). Some searches have more than one page of results.

Numbers here downloaded 17 February 2019. Note that only ‘classical games’ are considered here (rapid and exhibition games being ignored).

\textbf{See Also}

\texttt{chess}

\textbf{Examples}

\begin{verbatim}
 data(karpov_kasparov_anand)
 maxp(karpov_kasparov_anand)
 pie(maxp(karpov_kasparov_anand))
\end{verbatim}

---

\textbf{length.hyper2}

\textit{Length method for hyper2 objects}

\textbf{Description}

Length method for hyper2 objects, being the number of different brackets in the expression

\textbf{Usage}

\begin{verbatim}
 ## S3 method for class ‘hyper2’
 length(x)
\end{verbatim}

\textbf{Arguments}

\begin{itemize}
  \item \texttt{x} hyper2 object
\end{itemize}
loglik

Author(s)
Robin K. S. Hankin

Examples

data("oneill")
length(icons)
seq_along(icons)

loglik Log likelihood functions

Description
Returns a log-likelihood for a given hyper2 object at a specific point

Usage
loglik(H, p, log = TRUE)
like_single_list(p,Lsub)
like_series(p,L,log=TRUE)

Arguments
H An object of class hyper2
p A probability point. See details
log Boolean with default TRUE meaning to return the log-likelihood and FALSE meaning to return the likelihood
L,Lsub A list of hyper2 objects, or a list of list of loglik objects

Details
Function loglik() is a straightforward likelihood function. It takes the vector \( p = (p_1, \ldots, p_{n-1}) \) and returns the (log) likelihood. Note that the ‘fillup’ value is automatically appended; the elements of \( p \) are (linearly!) independent.

Function size() returns the (nominal) length \( n \) of nonnegative vector \( p = (p_1, \ldots, p_n) \) where \( p_1 + \cdots + p_n = 1 \).

If \( p \) is a matrix, the rows are interpreted as probability points.

Functions like_single_list() and like_series() are intended for use with gglol().

Note
Likelihood is defined up to an arbitrary multiplicative constant.
Log-likelihood (also known as support) is defined up to an arbitrary additive constant.
Author(s)
Robin K. S. Hankin

See Also
maxp

Examples

data(chess)
loglik(chess,c(1/3,1/3))

data(masterchef)

a1 <- rep(1/13,12)  # equal strengths
a2 <- indep(pmax_masterchef6)  # MLE

like_series(a1,masterchef_series6)
like_series(a2,masterchef_series6)

W <- hyper2(pnames=letters[1:6])
W1 <- ggol(W, 'a', letters[2:5],'f')  # 24-element list
W2 <- ggol(W, c('a','b'), c('c','d'),c('e','f'))  # 2^3=8 element list

like_single_list(rep(1/6,5),W1)  # information from first observation
like_series(rep(1/6,5),list(W1,W2))  # information from both observations

---

table

masterchef  
Masterchef series 6

Description
Data from Australian Masterchef Series 6

Usage
data(masterchef)

Format
Object masterchef_series6 is a list of hyper2 objects; pmax_masterchef_series6 and pmax_masterchef_series6_constrained are named vectors with unit sum.
Details

The object is created using the code in inst/masterchef_series6_allrounds.R, which is heavily documented. Not all the information available is included in the likelihood function as some of the early rounds result in an unmanageably large list. Inclusion is controlled by Boolean vector doo. The definitive source is the coloured table on the wiki page.

References


See Also

ggol

Examples

data(masterchef)

a1 <- rep(1/13,12)                      # equal strengths
a2 <- indep(pmax_masterchef6)          # MLE
a3 <- indep(pmax_masterchef6_constrained) # constrained MLE

like_series(a1, masterchef_series6)
like_series(a2, masterchef_series6)
like_series(a3, masterchef_series6)

maxp

Maximum likelihood estimation

Description

Find the maximum likelihood estimate for p, also equal probabilities

Usage

maxp(H, startp=NULL, give=FALSE, fcm=NULL, fcv=NULL, ...)  
equalp(H)

Arguments

H                      A hyper2 object
startp                 A vector of probabilities
give                   Boolean, with default FALSE meaning to return just the evaluate (including fillup), and TRUE meaning to return the entire formal output of the optimization routine
fcm, fcv               Further problem-specific constraints
...                    Further arguments which maxp() passes to constrOptim()
Details

Function `maxp()` returns the maximum likelihood estimate for \( p \), which has the unit sum constraint implemented. The function does not work for the output of `ggol()` nor the `masterchef_series6` dataset. These require a bespoke optimization as shown in the vignette.

Function `equalp()` returns the value of \( p \) for which all elements are the same.

In function `maxp()`, arguments `fcm` and `fcv` implement linear constraints to be passed to `constrOptim()`. These constraints are in addition to the usual nonnegativity constraints and unit-sum constraint. Examples of their use are given in the "icons" vignette.

Note

This functionality is peculiarly susceptible to off-by-one errors.

Author(s)

Robin K. S. Hankin

See Also

```
gradient.fillup
```

Examples

```r
data(chess)
maxp(chess)

data(rowing)

(x1 <- loglik(sculls2016, indep(maxp(sculls2016))))
(x2 <- loglik(sculls2016, indep(equalp(sculls2016))))

## get a p-value for the null of equal player strengths:
pchisq(2*(x1-x2), df=size(sculls2016)-1, lower.tail=FALSE)

## Note that Wilks's theorem is only asymptotic.
```

Description

Peculiar version of `expand.grid()` for matrices
Usage

\[
\text{mult\_grid}(L) \\
\text{pair\_grid}(a, b)
\]

Arguments

\[
L \quad \text{List of matrices} \\
a, b \quad \text{Matrices}
\]

Details

Function \text{pair\_grid}(a, b) returns a matrix with each column of a \text{cbind()}-ed to each column of b.

Function \text{mult\_grid()} takes a list of matrices; it is designed for use by \text{ggol()}.

Author(s)

Robin K. S. Hankin

See Also

\text{ggol}

Examples

\[
\text{pair\_grid(diag(2), diag(3))} \\
\text{mult\_grid(lapply(1:4, diag))}
\]

---

**NBA**  
*Basketball dataset*

Description

A point-by-point analysis of a basketball game

Usage

\text{data(NBA)}

Details

Dataset \text{NBA} is a dataframe contains a point-by-point analysis of a basketball match. Each row corresponds to a point scored. The first column is the time of the score, the second is the number of points scored, the third shows which team had possession at the start of play, and the fourth shows which team scored.

The other columns show the players. Table entries show whether or not that particular player was on the pitch when the point was scored.
Likelihood function NBA_likelihood is a hyper2 object that gives the log-likelihood function for this dataset. It may be created by executing basketball.R, in the inst/ directory.
Note that the two “ghost” players represent the effect of having possession.

References

- https://www.nbafullhd.com/24828-2/
- http://www.NBAbase.com/

Examples

data(NBA)
dotchart(maxp(NBA_likelihood))

Ops.hyper2 Arithmetic Ops Group Methods for hyper2 objects

Description

Allows arithmetic operators “+”, “*” and comparison operators “==” and “!=”, to be used for hyper2 objects.

Specifically, H1 + H2 implements addition of two log-likelihood functions, corresponding to incorporation of new observational data; and n*H1 implements H1+H1+...+H1, corresponding to repeated observations of the same data.

There are no unary operations for this class.

Usage

## S3 method for class 'hyper2'
Ops(e1, e2 = NULL)
## S3 method for class 'hyper2'
sum(x,...,na.rm=FALSE)
hyper2_add(e1,e2)
hyper2_equality(e1,e2)
hyper2_sum_numeric(H,r)

Arguments

e1,e2 Objects of class hyper2, here interpreted as hyperdirichlet distributions
x,...,na.rm In the sum() method, objects to be summed; na.rm is currently ignored
H,r In function hyper2_sum_numeric(), object H is a hyper2 object and r is a length-one real vector (a number)
Details

Testing for equality is not straightforward for two implementation reasons. Firstly, the object itself is stored internally as a stl map, which does not store keys in any particular order; and secondly, the stl set class is used for the brackets. A set does not include information about the order of its elements; neither does it admit repeated elements. See examples.

Function `hyper2_sum_numeric()` is defined so that idiom like

\[
H[p] <- H[p] + 3
\]

works (without this, one has to type \(H[p] \leftarrow \text{powers}(H[p]) + 3\), which sucks).

Value

Returns a hyper2 object or a Boolean.

Author(s)

Robin K. S. Hankin

Examples

```r
stopifnot(hyper2(list(1,1:2),1:2)==hyper2(list(1:2,1),2:1)) # stl map class
stopifnot(hyper2(list(1,1:2),1:2)==hyper2(list(c(1,1,2),1:2)) # stl set class

n <- 4
H <- hyper2(d=n)
for(i in seq_len(30)){
  jj <- sample(seq_len(n),sample(n,1),replace=FALSE)
  H[jj] <- H[jj] + 1
}
```

```r
# without pnames:
x1 <- hyper2(list(1,1:2,1:3),1:3)
x2 <- hyper2(list(4,5),10:11)
x1+x2
```

```r
# Same as above, but with pnames:
y1 <- hyper2(list(1,1:2,1:3),1:3,pnames=letters[1:3])
y2 <- hyper2(list(4,5),10:11,pnames=letters[1:5])
# Not run: y1+y2 # fails, pnames not identical
```

```r
pnames(y1) <- letters[1:5]
y1+y2
```
Print methods

Description
Print methods for hyper2 objects

Usage

```r
## S3 method for class 'hyper2'
print(x, ...)
```

Arguments

- `x`: An object of class hyper2
- `...`: Further arguments, currently ignored

Value
Returns the hyper2 object it was sent, invisibly. Used mainly for its side-effect

Note
This is the only place that the pnames attribute is used

Author(s)
Robin K. S. Hankin

Examples
```r
data(chess)
chess
```

rowing

Rowing dataset, sculling

Description
Data from Men’s single sculls, 2016 Summer Olympics

Usage
```r
data(rowing)
```
**Format**

sculls2016 is a hyper2 object that gives a likelihood function for the 2016 men’s sculls.

**Details**

Object sculls2016 is created by the code in inst/rowing_analysis.R. This reads file inst/rowing.txt, each line of which is a heat showing the finishing order. Character vector allrowers holds the names of all the rowers. File inst/rowing_minimal.txt has the same data but with dominated players (that is, any group of players none of whom have beaten any player not in the group) have been removed. This is because dominated players have a ML strength of zero.

**References**


**See Also**

ggol

**Examples**

data(rowing)
dotchart(maxp(sculls2016))

---

**rrank**  
*Random ranks*

**Description**

A function for producing ranks randomly, consistent with a specified strength vector

**Usage**

rrank(n = 1, p, pnames=NULL, fill = FALSE)
## S3 method for class 'rrank'
print(x, ...)

Arguments

- **n**: Number of observations
- **p**: Strength vector
- **pnames**: Character vector specifying names of the columns
- **fill**: Boolean, with default FALSE meaning to interpret the elements of p as strengths, notionally summing to one; and TRUE meaning to augment p with a fillup value
- **x, ...**: Arguments passed to the print method

Value

If n=1, return a vector; if n>1 return a matrix with n rows, each corresponding to a ranking. The canonical example is a race in which the probability of competitor i coming first is \( p_i / \sum p_j \), where the summation is over the competitors who have not already finished.

If, say, the first row of `rrank()` is c(2, 5, 1, 3, 4), then competitor 2 came first, competitor 5 came second, and so on.

Note that function `rrank()` returns an object of class `rrank`, which has its own special print method. The column names appear as “c1, c2, ...” which is intended to be read “came first”, “came second”, and so on. The difference between `rank` and `order` can be confusing.

```r
R> x <- c(3, 1, 2, 4.1)
R> rank(x)
[1] 3 1 2 4
R> order(x)
[1] 2 3 1 4
R>
```

In the current context, we have:

```r
R> rrank(2, ptrue)
   c1  c2  c3  c4
[1,] 3 1 2 4
[2,] 2 3 4 1
R>
```

In the above, each row is a race; we have four runners. In the first race, runner number 3 came first, runner 1 came second, runner 2 came third, and so on. In the second race, runner 2 came first, etc.

Now consider the following R idiom which uses the first race:

```r
R> p <- c(3,1,2,4)
R> order(p)  # slicker to say p[p] <- 1:4 as in inverse_word_single()
[1] 2 3 1 4
R>
```

This shows that runner 1 came second, runner 2 came third, runner 3 came first, and runner 4 came fourth.
Take the first race as an example.


**Author(s)**

Robin K. S. Hankin

**See Also**

ggol, order\_likelihood, skating

**Examples**

```r
ptue <- (4:1)/10
rrank(100,p=ptue)

r1 <- rrank(100,p=ptue)
H <- order\_likelihood(r1)
maxp(H)  # should be close to ptue

H <- H + order\_likelihood(rrank(100,maxp(H)))
maxp(H)
```

---

**rugby**

Super Rugby 2016 data

**Description**

Data from 2016 Super Rugby season. Two hyper2 objects, super2016 and super2016g. The second includes a “ghost” team whose strength helps the home team.

**Usage**

data(super\_rugby)
References


See Also

loglik

Examples

```r
data(super_rugby)
loglik(super2016, indep(maxp(super2016)))
loglik(super2016g, indep(maxp(super2016g)))
```

# differ by more than 2, ghost is significant!

---

saffy  

Choice matrix functionality

Description

Converts a matrix of restricted choices into a likelihood function. It is named for Saffron O’Neill.

Usage

```r
saffy(M)
```

Arguments

- **M**  
  
  Matrix

Details

The canonical example is Saffron’s climate change dataset, documented at icons. Function saffy() returns the appropriate likelihood function for the dataset.

Author(s)

Robin K. S. Hankin

See Also

icons
Examples

```r
data("oneill")
icons == saffy(icons_matrix)  # should be TRUE
```

**skating**

*Figure skating at the 2002 Winter Olympics*

Description

A likelihood function for the competitors at the Ladies’ Free Skate at the 2002 Winter Olympics

Usage

```r
data(skating)
```

Details

Object skating can be generated by running script inst/skating_analysis.R, which includes some further technical documentation.

The dataset is interesting because it has been analysed by many workers, including Lock and Lock, for consistency between the judges.

Note that file is structured so that each competitor is a row, and each judge is a column. Function order_likelihood() requires a transpose of this to operate.

References

- [https://en.wikipedia.org/wiki/Figure_skating_at_the_2002_Winter_Olympics#Full_results_2](https://en.wikipedia.org/wiki/Figure_skating_at_the_2002_Winter_Olympics#Full_results_2)
- Robin Lock and Kari Frazer Lock, Winter 2003. "Judging Figure Skating Judges". *STATS 36*, ASA

Examples

```r
data(skating)
dotchart(maxp(skating))

par(pty='s')
lik <- order(maxp(skating),decreasing=TRUE)
plot(1:23,lik,asp=1,pch=16,xlab='official order',ylab='likelihood order')
text(1:23,lik,pos=c(rep(4,19),rep(2,4)),pnames(skating))
abline(0,1)
```
### Description

Match outcomes from repeated singles table tennis matches

### Usage

```r
data(table_tennis)
```

### Format

A likelihood function corresponding to the match outcomes listed below.

### Details

There are four players, A, B, and C, who play singles table tennis matches with the following results:

- A vs B, A serves, 5-1
- A vs B, B serves, 1-3
- A vs C, A serves, 4-1
- A vs C, C serves, 1-2

As discussed in vignette `table_tennis_serve`, we wish to assess the importance of the serve. The vignette presents a number of analyses including a profile likelihood plot.

See vignette `table_tennis_serve` for an account of how to create `table_tennis`.

### Examples

```r
data(table_tennis)
dotchart(maxp(table_tennis))
```

---

### Description

Match outcomes from repeated doubles tennis matches

### Usage

```r
data(tennis)
```
Format

A hyper2 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:

<table>
<thead>
<tr>
<th>match</th>
<th>score</th>
</tr>
</thead>
<tbody>
<tr>
<td>${p_1, p_2}$ vs ${p_3, p_4}$</td>
<td>9-2</td>
</tr>
<tr>
<td>${p_1, p_3}$ vs ${p_2, p_4}$</td>
<td>4-4</td>
</tr>
<tr>
<td>${p_1, p_4}$ vs ${p_2, p_3}$</td>
<td>6-7</td>
</tr>
<tr>
<td>${p_1}$ vs ${p_3}$</td>
<td>10-14</td>
</tr>
<tr>
<td>${p_2}$ vs ${p_3}$</td>
<td>12-14</td>
</tr>
<tr>
<td>${p_1}$ vs ${p_4}$</td>
<td>10-14</td>
</tr>
<tr>
<td>${p_2}$ vs ${p_4}$</td>
<td>11-10</td>
</tr>
<tr>
<td>${p_3}$ vs ${p_4}$</td>
<td>13-13</td>
</tr>
</tbody>
</table>

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 $p_1$-$p_2$, each doubles partnership, is of approximately the same strength.

Dataset tennis gives the appropriate likelihood function for the players’ strengths; and dataset tennisGhost 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


Examples

data(tennis)
dotchart(maxp(tennis))
eigen(maxp(tennis,give=TRUE,hessian=TRUE)$hessian,TRUE,TRUE)$values

maxp(tennis_ghost)
tidy

Keep or discard players

Description

Functionality to keep or discard subsets of the players in a hyper2 object

Usage

tidy(H)
keep(H, keep, tidy=TRUE)
discard(H, discard, tidy=TRUE)

Arguments

H  A hyper2 object
tidy  Boolean, with default TRUE meaning to return a mathematically identical, but tidied, likelihood function
keep, discard  Players to keep or discard. May be character or integer

Details

Function tidy(H) returns a hyper2 object mathematically identical to H but with unused players (that is, players that do not appear in any bracket) removed.

Functions keep() and discard() will either keep or discard players specified in the second argument.

Note

Function tidy() is very, very, inelegant

Author(s)

Robin K. S. Hankin

Examples

data("oneill")
maxp(icons)
discard(icons,c("OA","WAIS"))

data("skating")
maxp(skating)[1:4]  # numbers work, keep the first four skaters
maxp(keep(skating,1:4))
Description

Converts a matrix of games into a likelihood function.

Usage

volley(M)

Arguments

M     Matrix

Details

The canonical example is the volleyball dataset. Each row is a volleyball game; each column is a player. An entry of 0 means “on the losing side”, an entry of 1 means “on the winning side”, and an entry of NA means did not play.

Author(s)

Robin K. S. Hankin

See Also

saffy

Examples

data(volleyball)
jj <- volley(volleyball_matrix)
stopifnot(jj == volleyball)
Index

*Topic datasets
  chess, 9
  counterstrike, 10
  handover, 19
  icons, 22
  interzonal, 25
  karpov_kasparov_anand, 25
  masterchef, 28
  NBA, 31
  rowing, 34
  rugby, 37
  skating, 39
  table_tennis, 40
  tennis, 40

*Topic package
  hyperRMpackage, 2

*Topic symbolmath
  Ops.hyper2, 32
  [.hyper2 (Extract), 14
  [<-.hyper2 (Extract), 14

accessor (cplusplus), 11
addL (cplusplus), 11
all_pnames (change_pnames), 6
allequal (maxp), 29
allrowers (rowing), 34
as.hyper2 (hyper2), 21
assigner (cplusplus), 11

B, 4
basketball (NBA), 31
black_wins (karpov_kasparov_anand), 25
brackets (hyper2), 21

change_pnames, 6
char2num (character_to_number), 8
character_to_number, 8
chess, 9, 25, 26
choose_losers (ggol), 17
choose_winners (ggol), 17
collusion (interzonal), 25
Connor (dirichlet), 12
counterstrike, 10
counterstrike_likelihood (counterstrike), 10
cplusplus, 11
curacao (interzonal), 25
dec (increment), 23
decrement (increment), 23
dhyper2 (B), 4
dhyper2_e (B), 4
differentiate, 19
differentiate (cplusplus), 11
Dirichlet (dirichlet), 12
dirichlet, 12
discard (tidy), 42
doubles (tennis), 40
doubles_ghost (tennis), 40
doubles_noghost (tennis), 40
drawn_games (karpov_kasparov_anand), 25
e_to_p (B), 4
equal (cplusplus), 11
equality (cplusplus), 11
equalp, 16
equalp (maxp), 29
equalprobs (maxp), 29
euro, 13
euro2009 (euro), 13
Eurovision (euro), 13
eurovision (euro), 13
Eurovision2009 (euro), 13
Eurovision_song_contest (euro), 13
evaluate (cplusplus), 11
Extract, 14, 22
extract (Extract), 14
Extract.skeleton (Extract), 14
extractor (Extract), 14

F1 (formula1), 16
F1_2014 (formula1), 16
F1_2015 (formula1), 16
F1_2016 (formula1), 16
F1_2017 (formula1), 16
fillup, 15, 30
formula1, 16
formula_1 (formula1), 16
formula_one (formula1), 16
GD (dirichlet), 12
gd (dirichlet), 12
GD_wong (dirichlet), 12
general_grouped_order_likelihood (ggol), 17
ggol, 17, 29, 31, 35, 37
gradient, 18, 30
handoff (handover), 19
handover, 19
head (hyper2), 20
hyper2, 12, 15, 21
hyper2_package, 2
hyper2_accessor (cplusplus), 11
hyper2_add (Ops.hyper2), 32
hyper2_addl (cplusplus), 11
hyper2_assigner (cplusplus), 11
hyper2_differentiate (cplusplus), 11
hyper2_equal (cplusplus), 11
hyper2_equality (Ops.hyper2), 32
hyper2_evaluate (cplusplus), 11
hyper2_identityl (cplusplus), 11
hyper2_overwrite (cplusplus), 11
hyper2_prod (Ops.hyper2), 32
hyper2_sum_numeric (Ops.hyper2), 32
icons, 22, 38
icons_matrix (icons), 22
identity (cplusplus), 11
inc (increment), 23
increment, 23
indep (fillup), 15
interzonal, 25
is_constant (hyper2), 21
is_valid_hyper2 (hyper2), 21
Jacobian (B), 4
karpov_kasparov_anand, 9, 25, 25
keep (tidy), 42
kka (karpov_kasparov_anand), 25
kka_3draws (karpov_kasparov_anand), 25
kka_3whites (karpov_kasparov_anand), 25
kka_array (karpov_kasparov_anand), 25
length (length.hyper2), 26
length.hyper2, 26
like_series (loglik), 27
like_single_list (loglik), 27
loglik, 6, 22, 27, 38
malpractice (handover), 19
MasterChef (masterchef), 28
masterchef, 28
masterchef_series6 (masterchef), 28
maxp, 28, 29
mean (B), 4
mean_hyper2 (B), 4
mgf (B), 4
Mosimann (dirichlet), 12
mult_grid, 30
NBA, 31
NBA_likelihood (NBA), 31
oneill (icons), 22
Ops (Ops.hyper2), 32
Ops.hyper2, 15, 22, 32
order_likelihood, 8, 37
order_likelihood (ggol), 17
overwrite (cplusplus), 11
p_to_e (B), 4
pair_grid (mult_grid), 30
ping_pong (table_tennis), 40
plays_white_draws (karpov_kasparov_anand), 25
plays_white_loses (karpov_kasparov_anand), 25
plays_white_wins (karpov_kasparov_anand), 25
pmax_masterchef6 (masterchef), 28
pmax_masterchef6_constrained (masterchef), 28
pnames (hyper2), 21
pnames (hyper2), 21
powers (hyper2), 21
Print, 34
print (Print), 34
print.rrank (rrank), 35
probability (B), 4
profile (table_tennis), 40
profile_likelihood (table_tennis), 40

rowing, 34
rrank, 17, 35
rugby, 37
rugby2016 (rugby), 37

saffy, 38, 43
sculling (rowing), 34
sculls2016 (rowing), 34
size (hyper2), 21
skating, 37, 39
skating_table (skating), 39
stockholm1962 (interzonal), 25
sum.hyper2 (Ops.hyper2), 32
super2016 (rugby), 37
super2016g (rugby), 37
super_rugby (rugby), 37

table_tennis, 40
table_tennis_serve (table_tennis), 40
tennis, 40
tennis_ghost (tennis), 40
tennis_noghost (tennis), 40
tidy, 42
trial (increment), 23

volley, 43
volleyball (volley), 43
volleyball_matrix (volley), 43

white_wins (karpov_kasparov_anand), 25