Package ‘MM’

May 16, 2018

Type Package

Title The Multiplicative Multinomial Distribution

Description Various utilities for the Multiplicative Multinomial distribution.

Version 1.6-5

Depends R (>= 2.10.0), Oarray (>= 1.4-6), partitions (>= 1.9-14)

Imports magic (>= 1.5-6), abind, emulator (>= 1.2-13), methods

Date 2018-05-16

Author Robin K. S. Hankin and P. M. E. Altham

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

LazyLoad yes

NeedsCompilation no

Repository CRAN

Date/Publication 2018-05-16 09:21:57 UTC

R topics documented:

`MM-package` *The Multiplicative Multivariate distribution, and the Multivariate Multiplicative Binomial Distribution*

Description

Two generalizations of the Multiplicative Binomial distribution of Altham (1978).

Details

Package:	MM
Type:	Package
Version:	1.0
Date:	2012-02-29
License:	GPL
LazyLoad:	yes

Author(s)

Robin K. S. Hankin and P. M. E. Altham

Maintainer: `<hankin.robin@gmail.com>`

References

P. M. E. Altham 1978. "Two Generalizations of the Binomial Distribution". *Applied Statistics* 27:162–167

P. M. E. Altham and Robin K. S. Hankin 2012. "Multivariate Generalizations of the Multiplicative Binomial Distribution: Introducing the MM Package", *Journal of Statistical Software*, 46(12), 1-23. http://www.jstatsoft.org/v46/i12/

Examples

```
data(voting)
Lindsey(voting, voting_tally)

jj <- paras(3)
```

```
rMM(10,4,jj)
```

---

danaher *Dataset due to Danaher*

Description

Dataset due to Danaher; also an analysis *ab initio*

Usage

```
data(danaher)
```

Format

- `danaher` is a matrix (of class `Oarray`) that represents Danaher and Hardie's Table 1

Details

Since bacon is often *eaten* with eggs, it is reasonable to expect that it is *purchased* with eggs.

Danaher and Hardie use a dataset obtained from a sample of 548 households over four consecutive store trips. They considered only grocery shopping trips with a total basket value of at least five dollars. For each household, they counted the total number of bacon purchases in their four eligible shopping trips, and the total number of egg purchases for the same trips.

Object `danaher` is a five-by-five matrix of class `Oarray` with entry (i, j) indicating the number of shoppers buying bacon on i occasions and eggs on j occasions (note the zero offset). Thus `danaher[1,2]=16` indicates that 16 shoppers bought bacon on 1 occasion and eggs on 2 occasions.

References

P. J. Danaher and B. G. S. Hardie 2005. "Bacon with your eggs? Applications of a new bivariate beta-binomial distribution". *The American Statistician*, 59(4):282

See Also

`optimizer`

Examples

```
data(danaher)
Lindsey_MB(danaher)

# Dataset from table 3 follows; see also the example at Lindsey.Rd
mags <-
c(2463, 35, 44, 14, 16, 7, 262, 20, 2, 2, 0, 0, 0, 2, 17, 2,
0, 2, 0, 0, 3, 8, 0, 0, 1, 0, 0, 4, 8, 0, 1, 1, 0, 0, 3, 3,
0, 0, 0, 0, 0, 1, 52, 2, 1, 0, 2, 0, 22)
```

```
dim(mags) <- c(7,7)
mags <- as.Oarray(mags,offset=0)
dimnames(mags) <-
list(AA=as.character(0:6),Sig=as.character(0:6))  # messy kludge in Lindsey_MB()
summary(Lindsey_MB(mags))
```

Extract.paras *Extract or Replace parameters of a* `paras` *object*

Description

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

Arguments

`x`	Object of class `paras`
`i`	Elements to extract or replace
`value`	Replacement value

Value

Always returns an object of class `paras`.

Methods

- x[i]
- x[i] <- value
- x[i,j]
- x[i,j] <- value

Note

These methods are included for completeness; it's not clear to me that they are likely to be used by anyone. It might be better to always use constructions like `x <- paras(4) ; p(x)[2] <- 0.1` instead; YMMV.

Author(s)

Robin K. S. Hankin

Examples

```
x <- paras(4)
x[2] <- 0.1
x[1,2] <- 0.12
x
```

gunter *Convert from multiple multivariate observations to tabular form*

Description

Convert from a matrix with rows corresponding to multivariate observations, to a tabular form listing every possible combination together with the number of times that combination was observed.

Usage

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

Arguments

obs	Argument. If a matrix, interpret each row as a multivariate observation (so the rowsums are constant). If an object of class `MB`, interpret appropriately; if an `Oarray`, coerce to an `MB` object
x	Object of class `gunter` to be printed by the print method
...	Further arguments, currently ignored

Value

For matrices and data frames, function `gunter()` returns an object of class `gunter`: a list of two elments, the first being a matrix (‘`obs`’) with rows being possible observations, and the second (‘`d`’) a vector with one entry for each row of matrix `obs`.

For `MB` objects and `Oarray` objects, function `gunter()` returns an object of class `gunter_MB`.

The print method returns its argument, invisibly, after printing it coerced to a list.

Author(s)

Bert Gunter, with tiny alterations by Robin Hankin

Examples

```
data(wilson)
gunter(non_met)

data(danaher)
gunter(danaher)  # object of class gunter_MB
```

Lindsey *The Poisson device of Lindsey and Mersch (1992).*

Description

Function `Lindsey()` returns a maximum likelihood fit of the multiplicative multinomial using the Poisson device of Lindsey and Mersch (1992), and in the context of the multiplicative multinomial by Altham and Lindsey (1998).

Function `Lindsey_MB()` returns a maximum likelihood fit for the multivariate multiplicative binomial, for the special case of a bivariate distribution. An example of coercing a table to the correct form for use with `Lindsey_MB()` is given in the examples section below. Also, see `danaher` for another example.

Usage

```
Lindsey(obs, n = NULL, give_fit = FALSE)
Lindsey_MB(a)
## S3 method for class 'Lindsey_output'
print(x, ...)
```

Arguments

`obs`	In `Lindsey()`, an integer matrix with each row corresponding to an observation. All row sums must match
`n`	Vector with elements corresponding to the rows of `obs`; default of `NULL` corresponds to observing each row of `obs` once
`a`	In `Lindsey_MB()`, an object that is coerced to one of class `gunter_MB`. Typically, the user supplies an `Oarray` object or an `MB` object
`give_fit`	Boolean, with default `FALSE` meaning to return just the fit, coerced to an object of class `paras` and `TRUE` meaning to return a list with two elements, the first being a `paras` object and the second being the fit returned by `glm()`
`x`	In the print method, object of class `Lindsey_output`
`...`	In the print method, further arguments, currently ignored

Details

Uses the device first described by Lindsey in 1992; the 'meat' of which has R idiom

```
Off <- -rowSums(lfactorial(jj$tbl))
glm(jj$d ~ -1 + offset(Off) + (.)^2, data=data, family=poisson)
```

Function `Lindsey(..., give_fit=TRUE)` returns an object of class `Lindsey_output`, which has its own print method (which prints the summary of the fit rather than use the default method).

Function `Lindsey(..., give_fit=FALSE)` returns an object of class `paras`, which can then be passed on to functions such as `rMM()`, which take a `paras` object.

Function `Lindsey_MB()` returns an object of class `glm`.

Author(s)

P. M. E. Altham and Robin K. S. Hankin

References

- J. K. Lindsey and G. Mersch 1992. "Fitting and comparing probability distributions with log linear models", *Computational Statistics and Data Analysis*, 13(4):373–384
- P. M. E. Altham and J. K. Lindsey, 1998. "Analysis of the human sex ratio using overdispersion models", *Applied Statistics*, 47:149–157

See Also

gunter, danaher

Examples

```
data(voting)
(o <- Lindsey(voting, voting_tally))
rMM(10,5,o)

data(danaher)
Lindsey_MB(danaher)

## Not run:   #(takes a long time)
data(pollen)
Lindsey(pollen)

## End(Not run)

# Example of Lindsey_MB() in use follows.

a <- matrix(c(63,40,26,7,69,42,19,5,48,21,16,2,33,11,9,1,21,8,9,0,
    7,8,1,0,5,3,1,0,9,2,0,0),byrow=TRUE,ncol=4)

# Alternatively, you can get this from the pscl package as follows:
# library(pscl); data(bioChemists)
# a <- table(subset(bioChemists, fem == 'Men' & art < 8))

dimnames(a) <- list(papers=0:7,children=0:3)
require(Oarray)
a <- as.Oarray(a,offset=0)
# thus a[3,1]==11 means that 11 subjects had 3 papers and 1 child

summary(Lindsey_MB(a))
```

MB — *Multivariate multiplicative binomial distribution*

Description

Various utilities to coerce and manipulate MB objects

Usage

```
MB(dep, m, pnames=character(0))
## S3 method for class 'MB'
as.array(x, ...)
## S4 method for signature 'MB'
getM(x)
## S3 method for class 'gunter_MB'
print(x, ...)
```

Arguments

dep	Primary argument to `MB()`. Typically a matrix with each row being an observation (see 'details' section below for an example). If an object of class `Oarray`, function `MB()` coerces to an `MB` object
m	Vector containing the relative sizes of the various marginal binomial distributions
x	Object of class `MB` to be converted to an `Oarray` object
...	Further arguments to `as.array()`, currently ignored
pnames	In function `MB()`, a character vector of names for the entries

Details

Function `MB()` returns an object of class `MB`. This is essentially a matrix with one row corresponding to a single observation; repeated rows indicate identical observations as shown below. Observational data is typically in this form. The idea is that the user can coerce to a `gunter_MB` object, which is then analyzable by `Lindsey()`.

The multivariate multiplicative binomial distribution is defined by

$$\prod_{i=1}^{t}\binom{m_i}{x_i\,z_i}p_i^{x_i}q_i^{z_i}\theta_i^{x_i z_i}\prod_{i<j}\phi_{ij}^{x_i x_j}$$

Thus if $\theta = \phi = 1$ the system reduces to a product of independent binomial distributions with probability p_i and size m_i for $i = 1, \ldots, t$.

There follows a short R transcript showing the `MB` class in use, with annotation.

The first step is to define an `m` vector:

```
R> m <- c(2,3,1)
```

This means that $m_1 = 2, m_2 = 3, m_3 = 1$. So $m_1 = 2$ means that $i = 1$ corresponds to a binomial distribution with size 2 [that is, the observation is in the set $\{0, 1, 2\}$]; and $m_2 = 3$ means that $i = 2$ corresponds to a binomial with size 3 [ie the set $\{0, 1, 2, 3\}$].

Now we need some observations:

```
R> a <- matrix(c(1,0,0, 1,0,0, 1,1,1, 2,3,1, 2,0,1),5,3,byrow=T)
R> a
     [,1] [,2] [,3]
[1,]    1    0    0
[2,]    1    0    0
[3,]    1    1    1
[4,]    2    3    1
[5,]    2    0    1
```

In matrix `a`, the first observation, viz `c(1,0,0)` is interpreted as $x_1 = 1, x_2 = 0, x_3 = 0$. Thus, because $x_i + z_i = m_i$, we have $z_1 = 1, z_2 = 3, z_3 = 1$. Now we can create an object of class `MB`, using function `MB()`:

```
R>  mx <- MB(a, m, letters[1:3])
```

The third argument gives names to the observations corresponding to the columns of `a`. The values of m_1, m_2, m_3 may be extracted using `getM()`:

```
R> getM(mx)
a b c
2 3 1
R>
```

The `getM()` function returns a named vector, with names given as the third argument to `MB()`.

Now we illustrate the print method:

```
R> mx
     a na     b nb     c nc
[1,] 1 1      0 3      0 1
[2,] 1 1      0 3      0 1
[3,] 1 1      1 2      1 0
[4,] 2 0      3 0      1 0
[5,] 2 0      0 3      1 0
R>
```

See how the columns are in pairs: the first pair total 2 (because $m_1 = 2$), the second pair total 3 (because $m_2 = 3$), and the third pair total 1 (because $m_3 = 1$). Each pair of columns has only a single degree of freedom, because m_i is known.

Also observe how the column names are in pairs. The print method puts these in place. Take the first two columns. These are named '`a`' and '`na`': this is intented to mean '`a`' and 'not `a`'.

We can now coerce to a `gunter_MB`:

```
R> (gx <- gunter(mx))
$tbl
   a b c
1  0 0 0
2  1 0 0
3  2 0 0
[snip]
24 2 3 1

$d
 [1] 0 2 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 0 0 0 0 0 0 1

$m
a b c
2 3 1
```

Take the second line of the element `tbl` of `gx`, as an example. This reads `c(1,0,0)` corresponding to the observations of `a,b,c` respectively, and the second line of element `d` ["d" for "data"], viz 2, shows that this observation occurred twice (and in fact these were the first two lines of `a`).

Now we can coerce object `mx` to an array:

```
R> (ax <- as.array(mx))
, , c = 0

   b
a   0 1 2 3
  0 0 0 0 0
  1 0 0 2 0
  2 0 0 0 0

, , c = 1

   b
a   0 1 2 3
  0 0 1 0 0
  1 0 0 0 0
  2 1 1 0 0
>
```

(actually, `ax` is an `Oarray` object). The location of an element in `ax` corresponds to an observation of `abc`, and the entry corresponds to the number of times that observation was made. For example, `ax[1,2,0]=2` shows that `c(1,2,0)` occurred twice (the first two lines of `a`).

The Lindsey Poisson device is applicable: see `help(danaher)` for an application to the bivariate case and `help(Lindsey)` for an example where a table is created from scratch.

Author(s)

Robin K. S. Hankin

See Also

MM, Lindsey, danaher

Examples

```
a <- matrix(c(1,0,0, 1,0,0, 1,1,1, 2,3,1, 2,0,1),5,3,byrow=TRUE)
m <- c(2,3,1)
mx <- MB(a, m, letters[1:3])   # mx is of class 'MB'; column headings
                   #  mean "a" and "not a".
ax <- as.array(mx)
gx <- gunter(ax)
ax2 <- as.array(gx)

data(danaher)
summary(Lindsey_MB(danaher))
```

MM *Various multiplicative multinomial probability utilities*

Description

Various multiplicative multinomial probability utilities for different types of observation

Usage

```
MM(y,n=NULL,paras)
MM_allsamesum(y, n=NULL, paras)
MM_differsums(y, n=NULL, paras)
MM_allsamesum_A(y, paras)
MM_differsums_A(y, paras)
MM_single(yrow, paras, givelog=FALSE)
MM_support(paras, ss)
```

Arguments

y	Observations: a matrix, each row is a single observation
yrow	A single observation corresponding to one row of matrix y
n	Integer vector with one element for each row of y. Default value of NULL means to interpret each row of y as being observed once
ss	Sufficient statistics, as returned by suffstats()
givelog	Boolean in MM_single() with TRUE meaning to return the log likelihood and default FALSE meaning to return the likelihood
paras	Object of class paras

Details

Consider non-negative integers $y_1, \ldots, y_k$ with $\sum y_i = y$. Then suppose the frequency function of the distribution $Y_1, \ldots, Y_k$ is

$$C \cdot \binom{y}{y_1, \ldots, y_k} \prod_{i=1}^{k} p_i^{y_i} \prod_{1 \le i < j \le k} {\theta_{ij}}^{y_i y_j}$$

where $p_i, \ldots, p_k \ge 0$, $\sum p_i = 1$ correspond to probabilities; and $\theta_{ij} > 0$ for $1 \le i < j \le k$ are additional parameters.

Here C stands for a normalization constant:

$$C = C(p, \theta, Y) = \sum_{y_1 + \cdots + y_k = y} \prod_{i=1}^{k} p_i^{y_i} \prod_{1 \le i < j \le k} {\theta_{ij}}^{y_i y_j}$$

which is evaluated numerically. This is expensive.

The usual case is to use function `MM()`.

- Function `MM()` returns the log of the probability of a matrix of rows of independent multinomial observations. It is a wrapper for `MM_allsamesum()` and `MM_differsums()`. Recall that optional argument `n` specifies the number of times that each row is observed. Calls `NormC()`.
- Function `MM_allsamesum()` gives the log of the probability of observing a matrix where the rowsums are identical. Calls `NormC()`.
- Function `MM_differsums()` gives the log of the probability of observing a matrix where the rowsums are not necessarily identical. **Warning:** This function takes a long time to run. Calls `NormC()`, possibly many times.
- Functions `MM_allsamesum_A()` and `MM_differsums_A()` are analogous to functions `MM_allsamesum()` and `MM_differsums()` but interpret the matrix `y` as having rows corresponding to observations; each row is observed once, as in `data(pollen)`. Both call `NormC()`.
- Function `MM_single()` gives a likelihood function for a `paras` object with a single multinomial observation (that is, a single line of matrix `y`). Does not call `NormC()`.
- Function `MM_support()` gives the support (that is, the log-likelihood) of a `paras` object; argument `ss` is the sufficient statistic, as returned by `suffstats()`. Does not call `NormC()`.
- Function `dMM()` [documented more fully at `rMM.Rd`] gives the probability of a single multivariate observation (ie a single row of the matrix argument `y`). Calls `NormC()`.

Author(s)

Robin K. S. Hankin

Examples

```
data(voting)

data(voting)
p <- Lindsey(voting, voting_tally)
```

```
MM(voting,voting_tally,p)   #No other value of 'p' gives a bigger value
```

---

multinomial *Multinomial function*

Description

The multinomial function and its logarithm

Usage

```
multinomial(x)
lmultinomial(x)
```

Arguments

x	Numeric vector

Details

Function `multinomial()` returns

$$\binom{n}{n_1, n_2, \ldots, n_k} = \frac{n!}{n_1! n_2! \cdots n_k!}$$

where $\sum_i n_i = n$, and function `lmultinomial()` returns the natural logarithm of this.

Note

Uses logarithmic functions to avoid overflow.

Author(s)

Robin K. S. Hankin

Examples

```
x <- runif(10)

exp(lmultinomial(x)) - multinomial(x) #should be small
```

NormC — *Normalizing constant for the multiplicative multinomial*

Description

Calculates the normalizing constant for the multiplicative multinomial using direct numerical summation

Usage

```
NormC(Y, paras, log = FALSE)
```

Arguments

Y	Total number of observations
paras	Object of class paras
log	Boolean, with default FALSE meaning to return the value, and TRUE meaning to return the natural logarithm

Author(s)

Robin K. S. Hankin

Examples

```
jj <- paras(3)
theta(jj) <- 2
NormC(5,jj)
```

optimizer — *Maximum likelihood estimator for the MM*

Description

Maximum likelihood estimator for the MM

Usage

```
optimizer(y, n = NULL, start = NULL, method = "nlm",
            printing = FALSE, give_fit=FALSE, ...)
optimizer_allsamesum(y, n = NULL, start = NULL, method = "nlm",
            printing = FALSE, give_fit=FALSE, ...)
optimizer_differsums(y, n = NULL, start = NULL, method = "nlm",
            printing = FALSE, give_fit=FALSE, ...)
```

Arguments

`y`	Matrix with each row being a possible observation
`n`	Counts of observations corresponding to rows of `y`
`start`	Start value for optimization routine, taken to be an object of class `paras`. Default value of `NULL` means to start with `Lindsey(y,n)`, which theoretically should be the maximum likelihood estimate
`method`	String giving which optimization method to use. Default of `Nelder` means to use `optim()` with the Nelder-Mead method; the other supported option is `nlm`
`printing`	Boolean, with `TRUE` meaning to print information as the optimization progresses and default `FALSE` meaning to print nothing
`give_fit`	Boolean, with default `FALSE` meaning to return the maximum likelihood estimate in the form of a `paras` object, and `TRUE` meaning to return a two-element list, the first being the output of `nlm()` or `optim()` and the second being the MLE
`...`	Further arguments passed to the optimization routine. In particular, note that `hessian=TRUE` is useful in conjunction with `give_fit=TRUE`

Details

Function `optimizer()` is the user-friendly version: it is a wrapper for `optimizer_samesum()` and `optimizer_differsums()`; it dispatches according to whether the rowsums are identical or not.

These functions are slow because they need to evaluate `NormC()` repeatedly, which is expensive.

Function `optimizer_samesum()` nominally produces the same output as `Lindsey()`, but is more computationally intensive.

Author(s)

Robin K. S. Hankin

See Also

`Lindsey`

Examples

```
data(voting)
p1 <- Lindsey(voting,voting_tally)
p2 <- optimizer(voting,voting_tally,start=p1)

theta(p1) - theta(p2)   # Should be zero

## Not run:
data(pollen)
p1 <- optimizer(pollen)
p2 <- Lindsey(pollen)
theta(p1) - theta(p2)   #  Isn't zero...numerical scruff...
```

```
## End(Not run)
```

---

paras *Manipulate a paras object*

Description

Various utilities to manipulate `paras` objects. Functions `pnames()` and `pnames<-()` operate on `MB` objects as expected.

Usage

```
paras(x, p, theta, pnames = character(0))
p(x) <- value
theta(x) <- value
p(x)
theta(x)
pnames(x)
pnames(x) <- value
getVals(x)
## S4 method for signature 'paras'
length(x)
```

Arguments

`x`	Object of class `paras`
`p`	In function `paras()`, a vector of the first $k-1$ elements of the probabilities
`theta`	In function `paras()`, a k by k matrix with diagonal composed of ones
`pnames`	In function `paras()`, a character vector of names for the entries
`value`	Replacement value

Details

A `paras` object contains the parameters needed to specify a multiplicative multinomial distribution.

Suppose `p` is an object of class `paras` object. Then `p` is a list of two elements. The first element, `p`, is a vector of length `length(p)` and the second is an upper-diagonal matrix square matrix of size `length(p)`. The vignette gives further details.

The functions documented here allow the user to inspect and change `paras` objects.

Author(s)

Robin K. S. Hankin

See Also

`MM`, `MB`

Examples

```
jj <- paras(5)
pnames(jj) <- letters[1:5]
p(jj) <- c(0.1, 0.1, 0.3, 0.1)
theta(jj) <- matrix(1:25,5,5)
pnames(jj) <- letters[1:5]
jj

# OK, we've defined jj, now use it with some other functions:
dMM(rep(1,5),jj)
MM_single(1:5,jj)
rMM(2,9,jj)
```

---

pollen *Pollen data from Mosimann 1962*

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 β-distribution, and correlations among proportions”. *Biometrika*, volume 49, numbers 1 and 2, pp65-82.

Examples

```
## Not run:
data(pollen)
Lindsey(pollen)

## End(Not run)
```

powell — *Dataset due to Powell (1990)*

Description

Dataset due to Powell (1990)

Usage

```
data(powell)
```

Format

A frequency table of counts of association data.

Source

- W. Powell, M. Coleman and J. McNicol 1990 "The statistical analysis of potato culture data". *Plant Cell, Tissue and Organ Culture* 23:159-164

Examples

```
data(powell)
Lindsey(powell, powell_counts)
```

rMM — *Random samples from the multiplicative multinomial*

Description

Density, and random samples drawn from, the multiplicative multinomial

Usage

```
rMM(n, Y, paras, burnin = 4*Y, every = 4*Y, start = NULL)
dMM(Y, paras)
```

Arguments

`n`	Number of observations to make
`Y`	Sum of each observation (for example, 100 for the `pollen` dataset, 4 for `voting`)
`paras`	Parameters of the MM distribution; an object of class `paras`
`every`	Each row is recorded every `every` steps through the Markov chain. Thus `every=10` means every tenth row is written to the returned matrix during MH process (and the other nine values are discarded)
`burnin`	Number of initial observations to ignore
`start`	Observation to start simulation, with default `NULL` corresponding to using a random start vector

Details

Function `rMM()` uses standard Metropolis-Hastings simulation.

Function `dMM()` is documented here for convenience; see `help(MM)` for related functionality.

Value

Returns a matrix with `n` rows and `length(paras)` columns. Each row is an observation.

Author(s)

Robin K. S. Hankin

See Also

`MM`

Examples

```
data(voting)
rMM(10,4,Lindsey(voting,voting_tally))

p <- paras(3)
theta(p) <- 2
dMM(1:3,p)
```

`skellam` *Brassica Dataset due to Catcheside*

Description

Dataset due to Catcheside, used by Skellam (1948) and subsequently by Altham (1978).

Usage

```
data(skellam)
```

Format

A frequency table of counts of association data.

Source

- J. G. Skellam 1948. "A probability distribution derived from the binomial distribution by regarding the probability of success as variable between the sets of trials". *Journal of the Royal Statistical Society, series B (Methodological)*. Volume 10, number 2, pp257-248.
- D. Catcheside 1937. Cytologia, Fujii Jub. Vol.

Examples

```
data(skellam)
Lindsey(skellam, skellam_counts)
```

---

suffstats — *Sufficient statistics for the multiplicative multinomial*

Description

Calculate, manipulate, and display sufficient statistics of the multiplicative multinomial. Functionality for analysing datasets, and distributions specified by their parameters is given; summary and print methods are also documented here.

Usage

```
suffstats(y, n = NULL)
expected_suffstats(L,Y)
## S3 method for class 'suffstats'
print(x, ...)
## S3 method for class 'suffstats'
summary(object, ...)
## S3 method for class 'summary.suffstats'
print(x, ...)
```

Arguments

`y,n`	In function `suffstats()`, argument `y` is a matrix with each row being a possible observation and `n` is counts of observations corresponding to rows of `y` with default `NULL` interpreted as each row of `y` being observed once. If `y` is an object of class `gunter`, this is interpreted sensibly
`L,Y`	In function `expected_suffstats()`, argument `L` is an object of class `Lindsey` [typically returned by function `Lindsey()`], and `Y` is the known constant sum (ie the `rowSums()` of the observations)
`x, object`	An object of class `suffstats` or `summary.suffstats`, to be printed or summarized
`...`	Further arguments to the print or summary methods. Currently ignored

Details

Function `suffstats()` returns a list comprising a set of sufficient statistics for the observations `y,[n]`.

This function requires that the rowsums of `y` are all identical.

Value

Function `suffstats()` returns a list of four components:

Y Rowsums of `y`

nobs Number of observations

row_sums Column sums of `y`, counted with multiplicity

cross_prods Matrix of summed squares

Function `summary.suffstats()` provides a summary of a `suffstats` object that is a list with two elements: `row_sums` and `cross_prods`, normalized with `nobs` and `Y` so that the values are comparable with that returned by `expected_suffstats()`. In particular, the sum of `row_sums` is the known sum y.

Author(s)

Robin Hankin and P. M. E. Altham

Examples

```
data(voting)
suffstats(voting, voting_tally)

data(wilson)
wilson <- gunter(non_met)
suffstats(wilson)

L <- Lindsey(wilson)

expected_suffstats(L,5)
summary(suffstats(wilson))  ## matches.

summary(suffstats(rMM(10,5,L))) # should be close.
```

`sweets` *Synthetic dataset due to Hankin*

Description

Four objects:

- `sweets` is a $2 \times 3 \times 21$ array
- `sweets_tally` is a length 37 vector
- `sweets_array` is a $2 \times 3 \times 37$ vector
- `sweets_table` is a 37×6 matrix

Usage

```
data(sweets)
```

Details

Object sweets is the raw dataset; objects sweets_table and sweets_tally are processed versions which are easier to analyze.

The father of a certain family brings home nine sweets of type mm and nine sweets of type jb each day for 21 days to his children, AMH, ZJH, and AGH.

The children share the sweets amongst themselves in such a way that each child receives exactly 6 sweets.

- Array sweets has dimension c(2,3,21): 2 types of sweets, 3 children, and 21 days. Thus sweets[,,1] shows that on the first day, AMH chose 0 sweets of type mm and 6 sweets of type jb; child ZJH chose 3 of each, and child AGH chose 6 sweets of type mm and 0 sweets of type jb.

  Observe the constant marginal totals: the kids have the same overall number of sweets each, and there are a fixed number of each kind of sweet.
- Array sweets_array has dimension c(2,3,37): 2 sweets, 3 children, and 37 possible ways of arranging a matrix with the specified marginal totals. This can be produced by allboards() of the **aylmer** package.
- sweets_table is a dataframe with six columns, one for each combination of child and sweet, and 37 rows, each row showing a permissible arrangement. All possibilities are present. The six entries of sweets[,,1] correspond to the six elements of sweets_table[1,]; the column names are mnemonics.
- sweets_tally shows how often each of the arrangements in sweets_tally was observed (that is, it's a table of the 21 observations in sweets)

Source

The Hankin family

Examples

```
data(sweets)

# show correspondence between sweets_table and sweets_tally:
cbind(sweets_table, sweets_tally)

# Sum the data, by sweet and child and test:
fisher.test(apply(sweets,1:2,sum))
# Not significant!




# Now test for overdispersion.
# First set up the regressors:
```

```
jj1 <- apply(sweets_array,3,tcrossprod)
jj2 <- apply(sweets_array,3, crossprod)
dim(jj1) <- c(2,2,37)
dim(jj2) <- c(3,3,37)

theta_xy <- jj1[1,2,]
  phi_ab <- jj2[1,2,]
  phi_ac <- jj2[1,3,]
  phi_bc <- jj2[2,3,]

# Now the offset:
Off <- apply(sweets_array,3,function(x){-sum(lfactorial(x))})

# Now the formula:
f <- formula(sweets_tally~ -1 + theta_xy + phi_ab + phi_ac + phi_bc)

# Now the Lindsey Poisson device:
out <- glm(formula=f, offset=Off, family=poisson)

summary(out)
#  See how the residual deviance is comparable with the degrees of freedom
```

`voting` *Synthetic dataset of voting behaviour due to Altham*

Description

Synthetic dataset of voting behaviour due to Altham

Usage

```
data(voting)
```

Format

`voting` is a three-column matrix with each row being a configuration of voting in a household with four members, and three choices. Vector `voting_tally` is a list of how many households voted, and `Nvoting_tally` is a more extreme dataset of the same type, used to uncover bugs in `Lindsey()`.

Source

Supplied by P. M. E. Altham

Examples

```
data(voting)
Lindsey(voting,voting_tally)
```

`wilson` *Housing Dataset due to Wilson*

Description

Dataset due to Wilson

Usage

```
data(wilson)
```

Format

Two objects, `met_area` and `non_met`, which have three columns and either 17 or 18 rows. Each row corresponds to a neighborhood of five households, each of which votes for one of three choices: US, S, or VS. Each column corresponds to one of these choices. The rowsums are constant because there are exactly five households in each neighborhood.

Source

- J. R. Wilson 1989. “Chi-square tests for Overdispersion with Multiparameter Estimates”, *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 38(3):441–453
- S. S. Brier 1980. “Analysis of Contingency Tables Under Cluster Sampling”, *Biometrika* 67(3):591–596

Examples

```
data(wilson)
Lindsey(non_met)
```

Index