Package: contingency (via r-universe)

November 5, 2024

Title Discrete Multivariate Probability Distributions

Version 0.1.0

Description Provides an object class for dealing with many multivariate probability distributions at once, useful for simulation.

Depends R (>= 3.5.0), rje

License GPL-2

Suggests knitr, rmarkdown, testthat

VignetteBuilder knitr

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.1

URL https://github.com/rje42/contingency

BugReports https://github.com/rje42/contingency/issues

Repository https://rje42.r-universe.dev

RemoteUrl https://github.com/rje42/contingency

RemoteRef HEAD

RemoteSha a0ffd9221cd674e6d6a91cc2536e988dea34f7b6

Contents

`aperm.tables` *Permute dimensions of tables*

Description

Method for permuting indices of tables object.

Usage

```
## S3 method for class 'tables'
aperm(a, perm, ...)
```

Arguments

`a`	object of class `tables`
`perm`	permutation of 1,...,k, where each table has k dimensions
`...`	other arguments to methods

Value

A permuted `tables` object.

as.array.tables *Convert tables into array*

Description

Convert tables into array

Usage

```
## S3 method for class 'tables'
as.array(x, ...)
```

Arguments

x	tables object
...	other arguments

Value

An array object

as.matrix.tables *Convert tables into matrix*

Description

Convert tables into matrix

Usage

```
## S3 method for class 'tables'
as.matrix(x, ...)
```

Arguments

x	tables object
...	other arguments

Value

A matrix object

as_tables *As tables*

Description

As tables

Usage

```
as_tables(x, tdim, conditional, rev = FALSE, ...)
```

Arguments

x	array or matrix object
tdim	dimensions for each table
conditional	integer vector of indices that are conditional
rev	logical: should output move through each table fastest?
...	other arguments for methods

Details

Transforms a vector, matrix or array into a tables object with the specified dimensions. Note that if rev = TRUE then the final dimension is (by default) used to index the tables objects, and otherwise the first dimension..

Value

A tables object.

capply *Apply function over tables*

Description

Apply a function to each contingency table in a tables object.

Usage

```
capply(x, f, ...)
```

Arguments

x	object of class tables
f	function to apply to each table
...	additional arguments to f

Value

a vector, matrix or list of outputs from the function `f`.

`checkCI` *Check conditional independence*

Description

Gives a numerical check that a (conditional) independence holds in a probability distribution.

Usage

```
checkCI(x, A, B, C = integer(0), eps = .Machine$double.eps, ...)

## S3 method for class 'array'
checkCI(x, A, B, C = integer(0), eps = .Machine$double.eps, ...)

## S3 method for class 'tables'
checkCI(x, A, B, C = integer(0), eps = .Machine$double.eps, ...)
```

Arguments

`x`	an array or object of class `tables`
`A, B`	the sets of variables whose independence is to be tested
`C`	conditioning set (possibly empty)
`eps`	tolerance parameter
`...`	other arguments to methods

Details

just tests to an appropriate numerical precision that a conditional independence holds: this is *not* a statistical test for conditional independence. If `A` and `B` overlap with `C` then these vertices are ignored. If `A` and `B` intersect with one another (but not `C`) then the solution is always false.

Value

A logical, or a vector of logicals of the same length as the number of tables provided, indicating whether the conditional independence seems to hold numerically.

Methods (by class)

- `checkCI(array)`: method for `array` object
- `checkCI(tables)`: method for `tables` object

`entropy` *Calculate entropy of discrete distribution*

Description

Calculate entropy of discrete distribution

Usage

```
entropy(p, ...)

## Default S3 method:
entropy(p, ...)

## S3 method for class 'array'
entropy(p, margin, ...)

## S3 method for class 'tables'
entropy(p, margin, ...)
```

Arguments

`p`	non-negative numeric vector
`...`	other arguments to methods
`margin`	margin to consider

Value

A numeric value of the entopy, or vector of entropies.

Methods (by class)

- `entropy(default)`: Default method for vectors
- `entropy(array)`: Method for arrays
- `entropy(tables)`: Method for `tables` object

interactionInf *Interaction information*

Description

Interaction information

Usage

```
interactionInf(p, ...)

## Default S3 method:
interactionInf(p, ..., condition)
```

Arguments

p	object to find interaction information for
...	other arguments to methods
condition	variables on which to condition

Value

Numeric value for interaction information, or a vector of interaction information values.

Methods (by class)

- `interactionInf(default)`: Default method for vectors

is_rev *Test if tables object is reversed*

Description

Checks if the rows of the underlying matrix represent each table or each entry in a table

Usage

```
is_rev(x)
```

Arguments

x	object to be tested

is_tables *Test object is a collection of tables*

Description

Checks if "tables" is in the list of class attributes.

Usage

```
is_tables(x)
```

Arguments

x	object to be tested

kl *Kullback-Leibler Divergence*

Description

Get the KL Divergence between two discrete distributions

Usage

```
kl(x, y, ...)

## Default S3 method:
kl(x, y, ...)

## S3 method for class 'tables'
kl(x, y, ...)
```

Arguments

x, y	vectors (of probabilities)
...	other arguments to methods

Value

a numberic value, vector or matrix of KL-divergences.

Methods (by class)

- kl(default): Default method for vectors
- kl(tables): Method for tables object

`margin` *Get margin of a table or tables*

Description

Get margin of a table or tables

Usage

```
margin(x, ...)

margin2(x, ...)

conditional(x, ...)

conditional2(x, ...)

intervention(x, ...)
```

Arguments

`x`	a contingency table or `tables` object
`...`	other arguments, not currently used

Details

`margin2` keeps all dimensions, and hence results will sum to the original sum, times the number of cells summed over.

Value

an object of the same class as `x`. The resulting array, or collection of tables, will contain a marginal, conditional or interventional distribution.

Functions

- `margin2()`: keep all dimensions
- `conditional()`: conditional distributions
- `conditional2()`: conditional distributions with all dimensions kept
- `intervention()`: interventional distributions

`margin.tables` *Get the marginal distributions*

Description

Get the marginal distributions

Usage

```
## S3 method for class 'tables'
margin(x, margin = NULL, order = TRUE, ...)

## S3 method for class 'tables'
conditional(
  x,
  variables,
  condition = NULL,
  condition.value = NULL,
  force = FALSE,
  undef = NaN,
  ...
)

## S3 method for class 'tables'
conditional2(x, variables, condition = NULL, force = FALSE, undef = NaN, ...)

## S3 method for class 'tables'
intervention(x, variables, condition, force = FALSE, ...)
```

Arguments

`x`	an object of class `tables`
`margin`	integer vector giving margin to be calculated (1 for rows, etc.)
`order`	logical indicating whether resulting indices should be in the same order as stated in `margin`
`...`	other arguments to function
`condition`	variables to condition upon
`condition.value`	(optionally) values to condition upon
`undef`	value to return for undefined cells

Details

Calculates marginal distributions for each entry in a `probMat`.

Value

An object of class `tables` consisting of the required marginal distribution.

Functions

- `conditional(tables)`: condition in distributions
- `conditional2(tables)`: condition and keep all variables
- `intervention(tables)`: intervene on variables in distributions

`multiInf` *Multiinformation*

Description

Get the multiinformation for a discrete distribution

Usage

```
multiInf(x, ...)

## Default S3 method:
multiInf(x, margin = NULL, ...)

## S3 method for class 'tables'
multiInf(x, margin = NULL, ...)
```

Arguments

`x`	vectors (of probabilities)
`...`	other arguments to methods
`margin`	margin to find multiinformation for

Value

a numberic value, vector or matrix of required multiinformation.

Methods (by class)

- `multiInf(default)`: Default method for vectors and arrays
- `multiInf(tables)`: Method for `tables` object

mutualInf *(Conditional) mutual information*

Description

(Conditional) mutual information

Usage

```
mutualInf(p, m1, m2, condition, ...)

## Default S3 method:
mutualInf(p, m1, m2, condition, ...)

## S3 method for class 'tables'
mutualInf(p, m1, m2, condition, ...)
```

Arguments

p	numeric array or tables class
m1, m2	margins for mutual information
condition	conditional margin
...	other arguments to methods

Value

Numeric value for mutual information, or a vector of mutual information values.

Methods (by class)

- mutualInf(default): Default method for vectors
- mutualInf(tables): Method for tables object

ntables *Number of tables*

Description

Number of tables

Usage

```
ntables(x)
```

Arguments

x	an object of class tables

Details

Gives the number of tables in an object of class tables.

Value

An integer.

---

perm_dim *Permute indices for variable* k

Description

Currently only works for binary dimensions.

Usage

```
perm_dim(x, k, perm, ...)
```

Arguments

x	array or related object
k	index to permute
perm	permutation to perform
...	other arguments (not currently used)

Details

Permutes the levels of one variable according to the permutation given in perm. Can be applied to matrices, arrays or tables.

Value

A permuted array or tables object.

print.tables *Print tables*

Description

Print method for object of class `tables`.

Usage

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

Arguments

`x`	object of class `tables`
`...`	arguments to pass to print method for an array

Value

The input provided (invisibly).

repTables *Turn distributions into tables*

Description

Turn distributions into tables

Usage

```
repTables(n, f, ..., rev = FALSE)
```

Arguments

`n`	number of distributions to generate
`f`	function that generates a probability distribution
`...`	arguments to `f`
`rev`	logical: should output move through each table fastest?

Value

a tables object containing the outputs of `f`

rprobMat *Generate matrix of (conditional) probability distributions*

Description

Generates discrete probability distributions in a matrix.

Usage

```
rprobMat(n, dim, d, alpha = 1, rev = FALSE)

rcondProbMat(n, dim, d, alpha = 1, condition, rev = FALSE)
```

Arguments

n	number of distributions
dim	dimension of contingency table for distributions
d	number of dimensions of table
alpha	parameter to use in dirichlet distribution
condition	which dimensions should be conditioned upon

Details

Returns an object of class tables consisting of discrete probability distributions. Each distribution is assumed to be a contingency table of dimension dim, and the probabilities are generated using a Dirichlet distribution with parameters all equal to alpha.

Value

A tables object containing random distributions.

Functions

- rcondProbMat(): Random conditional distributions

Examples

```
dat <- rprobMat(10, c(2,2,2))
```

tables *Create blank tables*

Description

Create blank tables

Usage

```
tables(n, tdim, rev = FALSE)
```

Arguments

n	number of tables
tdim	dimension of each table
rev	logical: should output move through each table fastest?

tab_dir *Wrappers for Dirichlet distribution over a table*

Description

Wrappers for Dirichlet distribution over a table

Usage

```
dtab_dir(x, alpha, log = FALSE)

rtab_dir(n, alpha, rev = FALSE)
```

Arguments

x	tables object of observations
alpha	table containing parameters
n	number of samples
rev	logical: should output move through each table fastest?

Details

This function obtains the Dirichlet density over a contingency table structure. In other words, suppose that we have a matrix observation $x = (x_{ij})$ where $\sum_{i,j} x_{i,j} = 1$. Then we might choose to model the vector x as having a Dirichlet distribution, with weights α_{ij}.

If alpha is a scalar in dtab_dir then it is applied to every entry in x.

Functions

- `dtab_dir()`: density function
- `rtab_dir()`: sampling function

`tbind` *Bind tables of the same dimension*

Description

Bind tables of the same dimension

Usage

```
tbind(x, ..., rev = FALSE)
```

Arguments

`x`	a `tables object`
`...`	further `tables` objects with the same `tdim` attributes
`rev`	logical: should output move through each table fastest?

`tdim` *Dimension of distributions over contingency tables*

Description

Dimension of distributions over contingency tables

Usage

```
tdim(x)

tdim(x) <- value
```

Arguments

`x`	an object of class `tables`
`value`	value to set parameters to

Details

The class `tables` is used to represent a collection of multidimentional tables; this function returns the dimension of each table.

Value

an integer vector of the dimensions

the `tables` object inputted with the new dimensions

Functions

- `tdim(x) <- value`: assign tables dimension

`tdimnames` *Dimension names for distributions over contingency tables*

Description

Dimension names for distributions over contingency tables

Usage

```
tdimnames(x)

tdimnames(x) <- value
```

Arguments

`x`	`tables` object
`value`	value to set dimension names to

Value

the `tables` object inputted with the new dimension names

Functions

- `tdimnames(x) <- value`: assign dimension names

`[.tables` *Subset object of class tables*

Description

Take subset of tables class.

Usage

```
## S3 method for class 'tables'
x[i, j, ..., drop = TRUE, keep = FALSE]
```

Arguments

`x`	object of class `tables`
`i`	indices of which tables to retain
`j`	which rows of each table to retain (or if `...` not specified, entries)
`...`	additional indices up to the dimension of the table
`drop`	usual logical indicating whether to consolidate margins of the table (doesn't apply to `i`)
`keep`	if only one table is specified with `i`, should the object output be an object of class `tables`? If not becomes a suitable array.

Details

There are two main ways to subset these tables. In both cases the first index refers to the tables being selected; one of the methods is to additionally specify all the indices corresponding to the tables, the other is to only specify a single entry. For example, `x[,1,2,2]` specifies the (1,2,2)th entry of each table; `x[,7]` will have the same effect for 2x2x2 tables.

If only one index is specified, then the function behaves just as ordinary subsetting on an array.

Value

A tables object over the specific entries and values selected.

Examples

```
x <- rprobMat(n=10, rep(2,3))
x[1,]
x[,1,1:2,1]
x[,1,1:2,1,drop=FALSE]
```

Index