Package 'combinat'

February 19, 2015

Version 0.0-8

Title combinatorics utilities

Author Scott Chasalow

Maintainer Vince Carey <stvjc@channing.harvard.edu>

Description routines for combinatorics

License GPL-2

Repository CRAN

Date/Publication 2012-10-29 08:58:26

NeedsCompilation no

R topics documented:

`combn` *Generate all combinations of the elements of x taken m at a time.*

Description

Generate all combinations of the elements of x taken m at a time. If x is a positive integer, returns all combinations of the elements of seq(x) taken m at a time. If argument "fun" is not null, applies a function given by the argument to each point. If simplify is FALSE, returns a list; else returns a vector or an array. "..." are passed unchanged to function given by argument fun, if any.

combn2:Generate all combinations of the elements of x taken two at a time. If x is missing, generate all combinations of 1:n taken two at a time (that is, the indices of x that would give all combinations of the elements of x if x with length n had been given). Exactly one of arguments "x" and "n" should be given; no provisions for function evaluation.

nCm: Compute the binomial coefficient ("n choose m"), where n is any real number and m is any integer. Arguments n and m may be vectors; they will be replicated as necessary to have the same length. Argument tol controls rounding of results to integers. If the difference between a value and its nearest integer is less than tol, the value returned will be rounded to its nearest integer. To turn off rounding, use tol = 0. Values of tol greater than the default should be used only with great caution, unless you are certain only integer values should be returned.

Usage

```
combn(x, m, fun=NULL, simplify=TRUE, ...)
```

Arguments

x	vector source for combinations
m	number of elements
fun	function to be applied to each combination (may be null)
simplify	logical, if FALSE, returns a list, otherwise returns vector or array
...	args to fun

Details

Nijenhuis, A. and Wilf, H.S. (1978) Combinatorial Algorithms for Computers and Calculators. NY: Academic Press.

Value

see simplify argument

Author(s)

Code by Scott Chasalow, R package and doc prep by Vince Carey, stvjc@channing.harvard.edu

References

~put references to the literature/web site here ~

Examples

```
 combn(letters[1:4], 2)
 combn(10, 5, min)  # minimum value in each combination
# Different way of encoding points:
 combn(c(1,1,1,1,2,2,2,3,3,4), 3, tabulate, nbins = 4)
#Compute support points and (scaled) probabilities for a
#Multivariate-Hypergeometric(n = 3, N = c(4,3,2,1)) p.f.:
# table.mat(t(combn(c(1,1,1,1,2,2,2,3,3,4), 3, tabulate,nbins=4)))
```

---

dmnom *density of multinomial, and support functions*

Description

density of multinomial

Usage

```
dmnom(x, size=sum(x), prob=stop("no prob arg"))
```

Arguments

x	vector
size	total
prob	parameter vector (sums to 1)

Author(s)

code by Scott Chasalow, R pack and maint by VJ Carey <stvjc@channing.harvard.edu>

Examples

```
dmnom(c(1,1,4,4),10,c(.2,.2,.3,.3))
```

---

hcube *Generate all points on a hypercuboid lattice.*

Description

Generate all points on a hypercuboid lattice.

Usage

```
hcube(x, scale, translation)
```

Arguments

x	Argument x is an integer vector giving the extent of each dimension; the number of dimensions is length(x).
scale	Argument scale is a vector of real numbers giving an amount by which to multiply the points in each dimension; it will be replicated as necessary to have the same length as x.
translation	Argument translate is a vector of real numbers giving an amount to translate (from the "origin", rep(1,length(x))) the points in each dimension; it will be replicated as necessary to have the same length as x. To use rep(0,length(x)) as the origin, use translation = -1. Scaling, if any, is done BEFORE translation.

Value

A prod(x) by length(x) numeric matrix; element (i,j) gives the location of point i in the jth dimension. The first column (dimension) varies most rapidly.

Author(s)

code by Scott Chasalow, R pack and maint by VJ Carey <stvjc@channing.harvard.edu>

References

~put references to the literature/web site here ~

See Also

fac.design, expand.grid

---

nsimplex *Computes the number of points on a (p, n)-simplex lattice*

Description

Computes the number of points on a (p, n)-simplex lattice; that is, the number of p-part compositions of n. This gives the number of points in the support space of a Multinomial(n, q) distribution, where p == length(q).

Arguments p and n are replicated as necessary to have the length of the longer of them.

Usage

```
nsimplex(p, n)
```

Arguments

p	vector of integers
n	vector of integers

Value

integer

Examples

```
nsimplex(3,5)
```

---

permn *Generates all permutations of the elements of x*

Description

Generates all permutations of the elements of x, in a minimal- change order. If x is a positive integer, returns all permutations of the elements of seq(x). If argument "fun" is not null, applies a function given by the argument to each point. "..." are passed unchanged to the function given by argument fun, if any.

Usage

```
permn(x, fun=NULL, ...)
```

Arguments

x	vector
fun	if non.null, applied at each perm
...	args passed to fun

Value

list: each component is either a permutation, or the results of applying fun to a permutation

References

Reingold, E.M., Nievergelt, J., Deo, N. (1977) Combinatorial Algorithms: Theory and Practice. NJ: Prentice-Hall. pg. 170.

See Also

sample, fact, combn, hcube, xsimplex

Examples

```
# Convert output to a matrix of dim c(6, 720)
 t(array(unlist(permn(6)), dim = c(6, gamma(7))))
   # A check that every element occurs the same number of times in each
   # position
   apply(t(array(unlist(permn(6)), dim = c(6, gamma(7)))), 2, tabulate,
           nbins = 6)

   # Apply, on the fly, the diff function to every permutation
   t(array(unlist(permn(6, diff)), dim = c(5, gamma(7))))
```

`rmultinomial` *Generate random samples from multinomial distributions*

Description

rmultinomial: Generate random samples from multinomial distributions, where both n and p may vary among distributions

rmultz2: fixed p case

Usage

```
rmultinomial(n, p, rows=max(c(length(n), nrow(p))))
rmultz2(n, p, draws=length(n))
```

Arguments

`n`	vector of sizes
`p`	vector or probs
`rows`	numeric giving desired number rows to be output
`draws`	number samples required

Value

a matrix of `rows` rows delivering specified samples

Author(s)

John Wallace, 17 Feb 1997 S-news , mods by Chasalow

Examples

```
n <- c(100,20,10)
p <- matrix(c(.3,.1,.5,.1,.1,.2,.6,.8,.3),3)
rmultinomial(n,p)
```

x2u — *Convert an x-encoded simplex-lattice point to a u-encoded simplex-lattice point*

Description

Convert an x-encoded simplex-lattice point to a u-encoded simplex-lattice point (equivalently, "untabulate" bin counts)

Usage

```
x2u(x, labels=seq(along = x))
```

Arguments

x	x: A numeric vector. x[i] is interpreted as the count in bin i.
labels	A vector. Interpreted as the bin labels; default value is seq(along = x), which causes return of a u-encoded simplex-lattice point. Other values of labels cause return of the result of subscripting labels with the u-encoded simplex-lattice point that would have been obtained if the default value of labels were used.

Value

rep(labels, x), a vector of length sum(x). If labels = seq(along = x) (the default), value is the u-encoded translation of the simplex lattice point, x. Equivalently, value gives the bin numbers, in lexicographic order, for the objects represented by the counts in x. For other values of argument "labels", value gives the bin labels for the objects represented by the counts in x (equivalent to labels[x2u(x)]).

See Also

tabulate, rep

xsimplex — *Generates all points on a (p,n) simplex lattice (i.e. a p-part composition of n).*

Description

Generates all points on a p,n simplex lattice (i.e. a p-part composition of n). Each point is represented as x, a p-dimensional vector of nonnegative integers that sum to n. If argument "fun" is not null, applies a function given by the argument to each point. If simplify is FALSE, returns a list; else returns a vector or an array. "..." are passed unchanged to function given by argument fun, if any.

Usage

```
xsimplex(p, n, fun=NULL, simplify=TRUE, ...)
```

Arguments

p	first parameter of lattice description
n	second parameter of lattice description
fun	function to be applied pointwise
simplify	logical: if FALSE, value is a list, otherwise a vector or array
...	parameters to be passed to fun

Examples

```
#Compute Multinomial(n = 4, pi = rep(1/3, 3)) p.f.:
   xsimplex(3, 4, dmnom, prob=1/3)
```

Index