Package 'prob'

August 27, 2018

Version 1.0-1

Date 2018-08-27

Title Elementary Probability on Finite Sample Spaces

Depends combinat, fAsianOptions

Suggests VGAM, reshape, MASS, hypergeo

Description A framework for performing elementary probability
calculations on finite sample spaces, which may be represented by data frames
or lists. Functionality includes setting up sample spaces, counting tools,
defining probability spaces, performing set algebra, calculating probability
and conditional probability, tools for simulation and checking the law of
large numbers, adding random variables, and finding marginal distributions.
Characteristic functions for all base R distributions are included.

License GPL (>= 3)

URL http://prob.r-forge.r-project.org, http://gkerns.people.ysu.edu/

Author G. Jay Kerns [aut, cre, cph]

Maintainer G. Jay Kerns <gkerns@ysu.edu>

Repository CRAN

Repository/R-Forge/Project prob

Repository/R-Forge/Revision 51

Repository/R-Forge/DateTimeStamp 2018-08-27 16:33:32

Date/Publication 2018-08-27 19:20:03 UTC

NeedsCompilation no

R topics documented:

---

prob-package   *Elementary Probability on Finite Sample Spaces*

Description

A framework for performing elementary probability calculations on finite sample spaces. It is built around the concept of a *probability space*, which is an object of outcomes and an object probs of probabilities associated with the outcomes.

There are two ways to represent a probability space in the prob package. The first is with a data frame that has a probs column. Entries of probs should be nonnegative and sum to one. The second way is with a list having two components: outcomes and probs. The component outcomes is a list containing elements of the most arbitrary sort; they can be data frames, vectors, more lists, whatever. The probs component is a vector (of the same length as outcomes), which associates to each element of outcomes a nonnegative number. As before, the only additional requirement is that the sum of probs be one.

There are functions in the prob package to address many topics in a standard course in elementary probability. In particular, there are methods for setting up sample spaces, counting tools, defining probability spaces, performing set algebra, calculating probability and conditional probability,

tools for simulation and checking the law of large numbers, adding random variables, and finding marginal distributions. See `vignette("prob")` for details.

There are some functions included to set up some of the standard sample spaces usually encountered in an elementary probability course. Examples include tossing a coin, rolling a die, drawing from a 52 card deck, *etc*. If you know of topics that would be of general interest and could be incorporated in the `prob` package framework, I would be happy to hear about them. Comments and suggestions are always welcomed.

The `prob` package is a first step toward addressing probability in R, and has been written in the spirit of simplicity. The procedures work best to solve problems that are manageable in scope. Users that wish to investigate especially large or intricate problems are encouraged to modify and streamline the code to suit their individual needs.

Characteristic functions for the base probability distributions have been included. For details, type `vignette("charfunc")` at the command prompt.

Details

Package:	prob
Version:	1.0-1
Date:	2018-08-27
Depends:	combinat, fAsianOptions
Suggests:	VGAM, reshape, MASS, hypergeo
LazyLoad:	no
License:	GPL version 3 or newer
URL:	http://prob.r-forge.r-project.org, http://gkerns.people.ysu.edu/

Author(s)

G. Jay Kerns <gkerns@ysu.edu>

Maintainer: G. Jay Kerns <gkerns@ysu.edu>

---

addrv   *Adding Random Variables to a Probability Space*

Description

Adds a column to a data frame probability space containing the values of a random variable computed from the existing columns of the space.

Usage

```
addrv(space, FUN = NULL, invars = NULL, name = NULL, ...)
```

Arguments

`space`	a data frame with a `probs` column.
`FUN`	a function to be applied to each row of outcomes in `space`
`invars`	a character vector indicating input columns of `space`
`name`	an (optional) name to give the defined random variable.
`...`	an expression defining a random variable.

Details

There are two ways to add a random variable to a data frame probability space; see the examples. The argument `FUN` has precedence and will be used if specified. If `name` is not specified, then the new random variable will be called `X`. Note that this function only works for data frames, as a method for objects of class `ps` has not been implemented.

Value

The input data frame with an additional column called `name`.

Author(s)

G. Jay Kerns <gkerns@ysu.edu>.

See Also

See [transform](#) to add a column to a data frame of outcomes (not yet a probability space).

Examples

```
S <-rolldie(3, makespace = TRUE)
addrv(S, sum, name = "Y")
addrv(S, Z = X3 - X2 )
```

---

cards	*A Standard Set of Playing Cards*

Description

The title says it all.

Usage

```
cards(jokers = FALSE, makespace = FALSE)
```

Arguments

jokers	logical.
makespace	logical.

Details

This generates a data frame sample space of a standard deck of 52 playing cards. Optionally, the user can specify that Jokers be included, which have a rank but with suit a missing value.

Value

A data frame with columns rank and suit, and optionally a column of equally likely probs.

See Also

[rolldie](#), [tosscoin](#), and [roulette](#)

Examples

```
cards()
cards(makespace = TRUE)
```

CharFunc *Characteristic functions*

Description

The characteristic functions for selected probability distributions supported by R. All base distributions are included, with the exception of wilcox and signedrank. For more resources please see the References, and for complete details and formulas see the charfunc vignette, which can be accessed by vignette("charfunc") at the command prompt. Only the simplest formulas are listed below.

Usage

```
cfbeta(t, shape1, shape2, ncp = 0)
cfbinom(t, size, prob)
cfcauchy(t, location = 0, scale = 1)
cfchisq(t, df, ncp = 0)
cfexp(t, rate = 1)
cff(t, df1, df2, ncp, kmax = 10)
cfgamma(t, shape, rate = 1, scale = 1/rate)
cfgeom(t, prob)
cfhyper(t, m, n, k)
cflnorm(t, meanlog = 0, sdlog = 1)
cflogis(t, location = 0, scale = 1)
cfnbinom(t, size, prob, mu)
```

```
cfnorm(t, mean = 0, sd = 1)
cfpois(t, lambda)
cfsignrank(t, n)
cft(t, df, ncp)
cfunif(t, min=0, max=1)
cfweibull(t, shape, scale = 1)
cfwilcox(t, m, n)
```

Arguments

t	numeric value. Some of the above are vectorized functions.
df	degrees of freedom (> 0, maybe non-integer)
df1, df2	degrees of freedom (> 0, maybe non-integer)
k	the number of balls drawn from the urn.
kmax	upper limit of summation.
lambda	vector of (positive) means.
location, scale	
	location and scale parameters; scale must be positive.
m	the number of white balls in the urn.
meanlog, sdlog	mean and standard deviation of the distribution on the log scale with default values of 0 and 1 respectively.
mean	vector of means.
min, max	(unif) lower and upper limits of the distribution. Must be finite and in the correct order.
mu	(nbinom) alternative parametrization via mean
n	the number of black balls in the urn.
ncp	non-centrality parameter δ
prob	probability of success in each trial.
rate	an alternative way to specify the scale; must be positive.
sd	vector of standard deviations.
shape	shape parameter, must be positive (gamma, weibull)
shape1, shape2	shape parameters (beta).
size	number of trials (binom) or target for number of successful trials (nbinom).

Details

The characteristic function ϕ of a random variable X is defined by

$$\phi(t) = E e^{itX}$$

for all $-\infty < t < \infty$.

Every random variable has a characteristic function, and every characteristic function uniquely determines the distribution of its associated random variable. For more details on characteristic functions and their properties, see Lukacs (1970).

Value

a complex number in rectangular (cartesian) coordinates.

Beta distribution

For the probability density function, see dbeta.

The characteristic function for central Beta is given by

$$\phi(t) =_1 F_1(\alpha; \alpha + \beta, it)$$

where F is the confluent hypergeometric function calculated with kummerM in the fAsianOptions package.

As of the time of this writing, we must calculate the characteristic function of the noncentral Beta with numerical integration according to the definition.

Binomial distribution

For the probability mass function, see dbinom.

The characteristic function is given by

$$\phi(t) = [pe^{it} + (1 - p)]^n$$

Cauchy Distribution

For the probability density function, see dcauchy.

The characteristic function is given by

$$\phi(t) = e^{(}it\theta - \sigma|t|)$$

Chi-square Distribution

For the probability density function, see dchisq.

The characteristic function is given by

$$\phi(t) = \frac{\exp(\frac{i\delta t}{1 - 2it})}{(1 - 2it)^{df/2}}$$

Exponential Distribution

For the probability density function, see dexp.

This is the special case of gamma when $\alpha = 1$.

F Distribution

For the probability density function, see df.

For the central F we use confluent hypergeometric function of the second kind, also known as kummerU, from the fAsianOptions package.

For noncentral F we use confluent hypergeometric function of the first kind. See the vignette for details.

Gamma Distribution

For the probability density function, see dgamma.

The characteristic function is given by

$$\phi(t) = (1 - \beta it)^{(} - \alpha)$$

Geometric Distribution

For the probability mass function, see dgeom.

This is the special case of negative binomial when $r = 1$.

Hypergeometric Distribution

For the probability mass function, see dhyper.

The formula for the characteristic function is based on the Gaussian hypergeometric series, calculated with hypergeo in package hypergeo. It is too complicated to be included here; please see the vignette.

Logistic Distribution

For the probability density function, see dlogis.

The characteristic function is given by

$$\phi(t) = \pi t / \sinh(\pi t)$$

Lognormal Distribution

For the probability density function, see dlnorm.

This characteristic function is uniquely complicated and delicate, but there is a recent numerical algorithm for computation due to Beaulieu (2008). See the vignette and the References.

Negative Binomial Distribution

For the probability mass function, see dnbinom.

The characteristic function is given by

$$\phi(t) = (p/(1 - (1 - p) * e^{it}))^r$$

Normal Distribution

For the probability density function, see dnorm.

The characteristic function is

$$\phi(t) = e^{i\mu t + t^2 \sigma^2 / 2}$$

Poisson Distribution

For the probability mass function, see dpois.

The characteristic function is

$$\phi(t) = e^{\lambda(e^{it} - 1)}$$

Wilcoxon Sign Rank Distribution

For the probability density function, see dsignrank.

The characteristic function is calculated according to the definition.

Student's t Distribution

For the probability density function, see dt.

See the vignette for a formula for the characteristic function for central t.

As of the time of this writing, we must calculate the characteristic function of the noncentral t with numerical integration according to the definition.

Continuous Uniform Distribution

For the probability density function, see dunif.

The characteristic function is

$$\phi(t) = \frac{e^{itb} - e^{ita}}{(b-a)it}$$

Weibull Distribution

For the probability density function, see dweibull.

We must at the time of this writing calculate the characteristic function with numerical integration according to the definition.

Wilcoxon Rank Sum Distribution

For the probability density function, see dwilcox.

The characteristic function is calculated according to the definition.

Author(s)

G. Jay Kerns <gkerns@ysu.edu>.

Source

For clnorm a fast numerical algorithm is used that originated with and was published and communicated to me by N. C. Beaulieu: see

References

Abramowitz, M. and Stegun, I. A. (1972) *Handbook of Mathematical Functions.* New York: Dover.

Beaulieu, N.C. (2008) Fast convenient numerical computation of lognormal characteristic functions, IEEE Transactions on Communications, Volume **56**, Issue 3, 331–333.

Hurst, S. (1995) The Characteristic Function of the Student-t Distribution, Financial Mathematics Research Report No. FMRR006-95, Statistics Research Report No. SRR044-95.

Johnson, N. L., Kotz, S., and Kemp, A. W. (1992) *Univariate Discrete Distributions*, Second Edition. New York: Wiley.

Johnson, N. L., Kotz, S. and Balakrishnan, N. (1995) *Continuous Univariate Distributions*, volume 1. New York: Wiley.

Johnson, N. L., Kotz, S. and Balakrishnan, N. (1995) *Continuous Univariate Distributions*, volume 2. New York: Wiley.

Lukacs, E. (1970) *Characteristic Functions*, Second Edition. London: Griffin.

See Also

[besselK](#) [kummerM](#) [kummerU](#) [hypergeo](#)

---

countrep   *Count Repetitions*

Description

Counts the number of repetitions of `vals` in a given vector `x`.

Usage

```
countrep(x, ...)

## Default S3 method:
countrep(x, vals = unique(x), nrep = 2,  ...)

## S3 method for class 'data.frame'
countrep(x, ...)
```

Arguments

x	an object in which repeats should be counted.
vals	values that may be repeated.
nrep	exact number of repeats desired, defaults to pairs.
...	further arguments to be passed to or from other methods.

Details

This is a generic function, with methods supplied for data frames and vectors. The default behavior counts the number of pairs of elements of `x`. One can find the number of triples, etc., by changing the `nrep` argument. If there are specific values for which one is looking for repeats, these can be specified with the `vals` argument. Note that the function only checks for *exactly* `nrep` instances, so two pairs of a specific element would be counted as 0 pairs and 1 quadruple. See the examples.

The data frame method uses `apply` to apply `countrep.default` to each row of the data frame.

Value

If `x` is a vector, then the value is an integer. If `x` is a data frame then the value is a vector, with entries the corresponding value for the respective rows of `x`

Author(s)

G. Jay Kerns <gkerns@ysu.edu>.

See Also

[isrep](#)

Examples

```
x <- c(3,3,2,2,3,3,4,4)
countrep(x)  # one pair each of 2s and 4s
countrep(x, nrep = 4)
countrep(x, vals = 4) # one pair of 4s
```

---

empirical	*Empirical Summary of a Simulation*

Description

Calculates relative frequencies of the rows of a data frame.

Usage

```
empirical(x)
```

Arguments

x   a data frame.

Details

The function works by adding a `probs` column to x with equally likely entries of $1/n$, where n is the number of rows. Then it aggregates the duplicated rows of x while accumulating the probabilities associated with each.

Value

A data frame formed by aggregating the rows of x. A `probs` column is added giving the relative frequencies of each of the rows.

Author(s)

G. Jay Kerns <gkerns@ysu.edu>.

See Also

[sim](#)

Examples

```
S <- tosscoin(2, makespace = TRUE)
sims <- sim(S, ntrials = 50000)
empirical(sims)
```

---

euchredeck   *A Deck of Playing Cards for Euchre*

Description

The title says it all.

Usage

```
euchredeck(benny= FALSE, makespace = FALSE)
```

Arguments

benny	logical.
makespace	logical.

Details

This is a conventional Euchre deck which uses a deck of 24 standard playing cards consisting of Ace, King, Queen, Jack, 10, and 9 of each of the four suits. If benny = TRUE then a Joker is added to the deck.

Value

A data frame with columns value and suit, and optionally a column of equally likely probs.

See Also

[cards](), [tosscoin](), and [roulette]()

Examples

```
euchredeck()
euchredeck(benny = TRUE, makespace = TRUE)
```

---

gen2wayTable | *Generate Two-way Tables*

Description

A function to randomly generate arbitrary two-way tables

Usage

```
gen2wayTable(n = sample(100:500, size = 1), pmatrix = matrix(1:12, nrow = 3),
dmnames = list(X = paste("x", 1:nrow(pmatrix), sep = ""),
Y = paste("y", 1:ncol(pmatrix), sep = "")),
addmargins = TRUE, as.df = FALSE, untable = TRUE)
```

Arguments

n	sum total observations
pmatrix	matrix of nonnegative weights for the probability distribution
dmnames	names of the table dimensions
addmargins	should margins be added?
as.df	table will be returned as a data frame
untable	should counts be untabled to single observation per row

Value

An object of class table containing the generated values.

Author(s)

G. Jay Kerns

---

genIndepTable | *Generate Independent Two-way Table*

Description

A function to generate a two-way table with independent margins

Usage

```
genIndepTable(n = sample(100:500, size = 1), prow = 1:3, pcol = 1:4,
dmnames = list(X = paste("x", 1:length(prow), sep = ""),
Y = paste("y", 1:length(pcol), sep = "")), addmargins = TRUE,
as.df = FALSE, untable = TRUE)
```

Arguments

n	sum total of observations generated
prow	nonnegative weights for the row marginal distribution
pcol	nonnegative weights for the col marginal distribution
dmnames	names for the table dimensions
addmargins	should margins be added to the table
as.df	should the result be returned as a data frame
untable	if true then data frame will be expanded to one observation per row

Details

This function will generate a two-way table with independent marginal distributions.

Value

Either an object of class table or a data frame.

Author(s)

G. Jay Kerns

---

genLogRegData   *Generate data for logistic regression*

Description

This function generates data ready for a logistic regression model

Usage

```
genLogRegData(xdata, beta = rep(1, ncol(xdata)), yname = "y")
```

Arguments

xdata	the model matrix
beta	vector of parameters to multiply the model matrix
yname	the name for the generated y values

Details

This function generates data ready for a logistic regression model

Value

A data frame with the model matrix and the generated y values added

Author(s)

G. Jay Kerns

---

genXdata   *Generate continuous model matrix data*

Description

This function generates correlated normal data to serve as a model matrix in a regression model.

Usage

```
genXdata(n, nvar = 1, mu = rep(0, nvar), Sigma = diag(length(mu)),
varnames = paste("x", 1:length(mu), sep = ""), roundto = NULL)
```

Arguments

n	how many rows
nvar	how many columns
mu	the mean of the multivariate normal distribution
Sigma	the variance-covariance matrix of the normal distribution
varnames	how you would like the variables to be named in the result
roundto	number of places to round the generated values

Details

This function generates correlated normal data to serve as a model matrix in a regression model.

Value

A data frame of generated data

Author(s)

G. Jay Kerns

---

iidspace   *Independent Identical Experiments*

Description

Sets up a probability space corresponding to independent, identical experiments.

Usage

```
iidspace(x, ntrials, probs = NULL)
```

Arguments

x	a vector of outcomes.
ntrials	number of times to perform the experiment.
probs	vector of non-negative weights corresponding to x.

Details

The elementary experiment to be repeated consists of drawing an element of x according to the probabilities contained in `probs`. The entries of `probs` need not sum to one, but they will be normalized before any computations. If `probs` is not specified, the equally likely model will be assumed.

Value

A data frame, with a `probs` column, where `probs` is calculated to be the probability of observing the outcome in its row under the assumption of independence and identical distribution of draws from x.

Author(s)

G. Jay Kerns <gkerns@ysu.edu>.

See Also

[probspace](#)

Examples

```
iidspace( 1:6, ntrials = 3) # same as rolldie(3)
iidspace( 1:6, ntrials = 3, probs = 3:8 ) # unbalanced die
```

---

intersect   *Intersection of Subsets*

Description

Calculates the intersection of subsets of a probability space. Comparisons are made row-wise, so that in the data frame case, intersect(A,B) is a data frame with those rows that are both in A and in B.

Usage

```
intersect(x, ...)

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

## S3 method for class 'data.frame'
intersect(x, y, ...)

## S3 method for class 'ps'
intersect(x, y, ...)
```

Arguments

x, y   vectors, data frames, or ps objects containing a sequence of elements (concep-
                tually).

...   further arguments to be passed to or from other methods.

Details

This is a generic function, extended from the intersect function in the base package. The ele-
ments of intersect(x,y) are those elements in x and in y. The original definition is preserved in
the case that x and y are vectors of the same mode.

Value

A vector, data frame, or subset of a probability space of the same type as its arguments.

Author(s)

G. Jay Kerns <gkerns@ysu.edu>, based on a suggestion made by Brian Ripley in R-devel, 12/11/07.

See Also

union, setdiff

Examples

```
S <- cards()
A <- subset(S, suit == "Heart")
B <- subset(S, rank == "A" )
intersect(A, B)
```

---

is.probspace *Testing for a Probability Space*

Description

Decides whether a given object is a probability space.

Usage

```
is.probspace(x)
```

Arguments

x   an object for which probability space status should be checked.

Details

It first checks if the class of the object includes ps, and if so TRUE is returned. If not, then it checks that the object is a data frame and contains a probs column. Lastly, it checks whether all entries of probs are nonnegative. Note that it does not check whether the sum of probs is one, to allow for the possibility that the input object is a proper subset of a probability space.

Value

Logical.

Author(s)

G. Jay Kerns <gkerns@ysu.edu>.

See Also

probspace

Examples

```
S <- rolldie(3, makespace = TRUE)
is.probspace(S)
```

---

isin   *Test Whether One Vector Is In Another Vector*

Description

See the title.

Usage

```
isin(x, ...)

## Default S3 method:
isin(x, y, ordered = FALSE, ...)

## S3 method for class 'data.frame'
isin(x, ...)
```

Arguments

x, y   vectors.

ordered   logical.

...   further arguments to be passed to or from other methods.

Details

The function will only return TRUE if every element of y is present in the vector x, counting multi-plicity. See the examples below. Of ordered = TRUE, then elements must be in the vector x in the order specified in y. Compare this to the behavior of the %in% function in the base package.

This is a generic function with a moethod for data frames, which applies isin() to each row of the data frame, with a vector as a result.

Value

Logical, or a vector of logicals.

Author(s)

G. Jay Kerns <gkerns@ysu.edu>.

See Also

isrep

Examples

```
x <- 1:10
y <- 3:7
z <- c(3,3,7)
isin(x,y)
isin(x,z)
isin(x, c(3,4,5), ordered = TRUE)
isin(x, c(3,5,4), ordered = TRUE)

S <- rolldie(4)
subset(S, isin(S, c(2,2,6), ordered = TRUE))
```

---

isrep   *Is Repeated in a Vector*

Description

Tests for a certain number of repetitions of `vals` in a given vector `x`.

Usage

```
isrep(x, ...)

## Default S3 method:
isrep(x, vals = unique(x), nrep = 2,  ...)

## S3 method for class 'data.frame'
isrep(x, ...)
```

Arguments

x	an object with potential repeated values.
vals	values that may be repeated.
nrep	exact number of repeats desired, defaults to pairs.
...	further arguments to be passed to or from other methods.

Details

This is a generic function, with methods supplied for data frames and vectors. The default behavior tests for existence of pairs of elements of `x`. One can test existence of triples, etc., by changing the `nrep` argument. If there are specific values for which one is looking for repeats, these can be specified with the `vals` argument. Note that the function only checks for *exactly* `nrep` instances, so two pairs of a specific element would be counted as 0 pairs and 1 quadruple. See the examples.

The data frame method uses `apply` to apply `isrep.default` to each row of the data frame.

Value

Logical.

Author(s)

G. Jay Kerns <gkerns@ysu.edu>.

See Also

countrep

Examples

```
x <- c(3,3,2,2,3,3,4,4)
isrep(x)  # one pair each of 2s and 4s
isrep(x, nrep = 4)
isrep(x, vals = 4) # one pair of 4s
```

---

marginal   *Marginal Distributions*

Description

Computes the marginal distribution of a set of variables.

Usage

```
marginal(space, vars = NULL)
```

Arguments

space	a data frame probability space or a subset of one.
vars	an optional character vector of variable names in space.

Details

If vars is not specified, then marginal() will set vars to be all non-probs columns, which can be useful in the case that it is desired to aggregate duplicated rows.

Value

A data frame with a probs column.

Author(s)

G. Jay Kerns <gkerns@ysu.edu>.

See Also

See [addrv](#) for adding random variables to a data frame probability space.

Examples

```
S <- rolldie(3, makespace = TRUE)
marginal(S, vars = c("X1", "X2"))
```

---

noorder   *Sort and Merge Probability Space Outcomes*

Description

This function sorts the rows (outcomes) of a data frame probability space, effectively removing the original order present) and aggregates the sorted rows into a new probability data frame with unique, sorted outcomes.

Usage

```
noorder( space )
```

Arguments

space   a data frame probability space or a subset of one.

Details

The data frame space must have at least two non-probs columns or an error will result.

Value

A data frame with a probs column and sorted, unique rows.

Author(s)

G. Jay Kerns <gkerns@ysu.edu>.

See Also

[addrv](#), [marginal](#)

Examples

```
S <- tosscoin(3, makespace = TRUE)
noorder(S)
```

---

nsamp   *Number of Samples from an Urn*

Description

Calculates the number of samples from an urn under different sampling scenarios.

Usage

```
nsamp(n, k, replace = FALSE, ordered = FALSE)
```

Arguments

n	an integer or integer vector.
k	an integer or integer vector.
replace	logical indicating whether sampling should be done with replacement.
ordered	logical indicating whether order among samples is important.

Details

The nsamp() function will calculate the number of samples from an urn under assorted assumptions on the sampling procedure. The arguments are: n, the number of (distinguishable) objects in the urn, k, the sample size, and replace, ordered as documented in [urnsamples](#).

nsamp() is vectorized, so that entering vectors instead of numbers for n, k, replace, and ordered results in a vector of corresponding answers.

The formulas used in the four possible combinations of replace and ordered are as follows.

- When replace = TRUE and ordered = TRUE, the value is n^k.
- When replace = FALSE and ordered = TRUE, the value is $n!/(n-k)!$.
- When replace = FALSE and ordered = FALSE, the value is $n!/[k!(n-k)!]$.
- When replace = TRUE and ordered = FALSE, the value is $(n-1+k)!/[(n-1)!k!]$.

Value

A number.

Author(s)

G. Jay Kerns <gkerns@ysu.edu>.

See Also

[urnsamples](#)

Examples

```
nsamp(n=3, k=2, replace = TRUE, ordered = TRUE)
nsamp(n=3, k=2, replace = TRUE, ordered = FALSE)
nsamp(n=3, k=2, replace = FALSE, ordered = FALSE)
nsamp(n=3, k=2, replace = FALSE, ordered = TRUE)
```

---

permsn   *Generate All Permutations of x Elements Taken m at a Time*

Description

Generate all permutations of the elements of x taken m at a time. If x is a positive integer, returns all permutations of the elements of seq(x) taken m at a time.

Usage

```
permsn(x, m)
```

Arguments

x	vector source for permutations, or integer n for x <- seq(n).
m	number of elements to permute.

Value

a list or array (in nondegenerate cases).

Author(s)

G. Jay Kerns <gkerns@ysu.edu>, modified from the combn function in the package utils.

See Also

[combn](#)

Examples

```
permsn(3,2)
```

| prob | *Probability and Conditional Probability* |

Description

Calculates probability and conditional probability of events.

Usage

```
Prob(x, ...)

## Default S3 method:
Prob(x, event = NULL, given = NULL, ...)

## S3 method for class 'ps'
Prob(x, event = NULL, given = NULL, ...)
```

Arguments

x	a probability space or a subset of one.
event	logical expression indicating elements or rows of space to keep: missing values are taken as false.
given	either a subset of a probability space or a logical expression indicating elements or rows of space to keep: missing values are taken as false.
...	further arguments to be passed to or from other methods.

Details

This function calculates the probability of events or subsets of a given sample space. Conditional probability is also implemented. In essence, the Prob() function operates by summing the probs column of its argument. It will find subsets on the fly if desired.

The event argument is used to define a subset of x, that is, the only outcomes used in the probability calculation will be those that are elements of x and satisfy event simultaneously. In other words, Prob(x,event) calculates Prob(intersect(x, subset(x, event))). Consequently, x should be the entire probability space in the case that event is non-null.

There is some flexibility in the given argument in that it can be either a data frame or it can be a logical expression that defines the subset. However, that flexibility is limited. In particular, if given is a logical expression, then event must also be specified (also a logical expression). And in this case, the argument x should be the entire sample space, not a subset thereof.

Value

A number in the interval [0,1].

Author(s)

G. Jay Kerns <gkerns@ysu.edu>.

See Also

   `probspace`, `iidspace`

Examples

```
S <- rolldie(times = 3, makespace = TRUE )
Prob(S, X1+X2 > 9 )
Prob(S, X1+X2 > 9, given = X1+X2+X3 > 7 )
```

---

probspace   *Probability Spaces*

Description

   Forms a probability space from a set of outcomes and (optional) vector of probabilities.

Usage

```
probspace(x, ...)

## Default S3 method:
probspace(x, probs, ...)

## S3 method for class 'list'
probspace(x, probs, ...)
```

Arguments

x	a vector, data frame, or list of outcomes.
probs	a vector of non-negative weights of the same length as `outcomes`
...	further arguments to be passed to or from other methods.

Details

   The elements of `probs` will be normalized to ensure that their sum is one. If `probs` is not specified, then the equally likely model is assumed in which every outcome has the same probability.

Value

   If `outcomes` is a vector or data frame, then the value is a data frame with an added `probs` column. If `outcomes` is a list, then the value is a list with components `outcomes` (the supplied list) and a `probs` component.

Author(s)

   G. Jay Kerns <gkerns@ysu.edu>.

See Also

iidspace

Examples

```
R <- rolldie(3)
probspace(R)
```

---

rolldie	*Rolling a Die*

Description

Sets up a sample space for the experiment of rolling a die repeatedly.

Usage

```
rolldie(times, nsides = 6, makespace = FALSE)
```

Arguments

times	number of rolls.
nsides	number of sides on the die.
makespace	logical.

Details

The function uses expand.grid() to generate all possible rolls resulting from the experiment of rolling a die. Sides on the die are 1:nsides. Columns of the data frame are called X1, X2, up to Xtimes

Value

A data frame, with an equally likely probs column if makespace is TRUE.

Author(s)

G. Jay Kerns <gkerns@ysu.edu>.

See Also

tosscoin

Examples

```
rolldie(2)
rolldie(3, nsides = 4)
rolldie(3, nsides = 4, makespace = TRUE)
```

---

roulette   *Roulette*

Description

Sets up a sample space for the experiment of spinning a roulette wheel once.

Usage

```
roulette(european = FALSE, makespace = FALSE)
```

Arguments

european   logical.

makespace   logical.

Details

If `european` is `TRUE`, then a traditional EU roulette wheel with 37 pockets is used, otherwise, a standard US roulette wheel with 38 pockets is used. Entries in the data frame are ordered in the customary way to facilitate the calculation probabilities regarding called bets.

Value

A data frame, with columns `num` and `color`, and an equally likely `probs` column if `makespace` is `TRUE`.

Author(s)

G. Jay Kerns <gkerns@ysu.edu>.

See Also

[cards](#)

Examples

```
roulette()
roulette(european = TRUE, makespace = TRUE)
```

---

setdiff   *Set Difference of Subsets*

Description

Calculates the (nonsymmetric) set difference of subsets of a probability space.

Usage

```
setdiff(x, ...)

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

## S3 method for class 'data.frame'
setdiff(x, y, ...)

## S3 method for class 'ps'
setdiff(x, y, ...)
```

Arguments

x, y   vectors, data frames, or ps objects containing a sequence of items (conceptu-
             ally).

...   further arguments to be passed to or from other methods.

Details

This function operates row-wise on dataframes, and element-wise among the outcomes of ps ob-
jects. The elements of setdiff(x,y) are those elements in x but not in y. The definition is taken
to match the version in the base package.

Value

A data frame or subset of a probability space of the same type as its arguments.

Author(s)

G. Jay Kerns <gkerns@ysu.edu>, essentially verbatim from a suggestion made by Brian Ripley on
R-devel, 12/11/07.

See Also

[intersect](), [union]()

Examples

```
S <- cards()
A <- subset(S, suit == "Heart")
B <- subset(S, rank == "A" )
setdiff(B, A)
```

---

sim   *Simulate Draws from a Sample Space*

Description

Simulates the experiment of drawing from a sample space.

Usage

```
sim(x, ...)

## Default S3 method:
sim(x, ntrials, ...)

## S3 method for class 'ps'
sim(x, ntrials, ...)
```

Arguments

x	a probability space or a subset of one.
ntrials	number of times to repeat the experiment.
...	further arguments to be passed to or from other methods.

Details

The `sim()` function is a wrapper for `sample()`, except that it strips the `probs` component from the result and (if x is a data frame) renames the rownames of the data frame consecutively from `1:ntrials`.

Value

A data frame if space is a data frame, or a list if space is of class ps.

Author(s)

G. Jay Kerns <gkerns@ysu.edu>.

See Also

[empirical](#)

Examples

```
S <- cards(makespace = TRUE)
sim(S, ntrials = 5)

T <- urnsamples(S, 2)
U <- probspace(T)
sim(U, ntrials = 4)
```

---

subset   *Subsets of Probability Spaces*

Description

This is a method for subset() for the case when the input object is a probability space of class ps.

Usage

```
subset(x, ...)

## S3 method for class 'ps'
subset(x, subset, ...)
```

Arguments

x	a probability space.
subset	logical expression indicating elements or rows of space to keep: missing values are taken as false.
...	further arguments to be passed to or from other methods.

Details

This function simply extends the existing subset() function to ps objects.

Value

A ps object, a subset of a probability space.

Author(s)

G. Jay Kerns <gkerns@ysu.edu>.

See Also

[intersect](#), [setdiff](#), [union](#), [isin](#)

Examples

```
L <- tosscoin(2)
M <- urnsamples(L, 3)
N <- probspace(M)
subset(N, all(toss1=="H"))
subset(N, any(toss2=="T"))
```

---

tosscoin   *Tossing a Coin*

Description

Sets up a sample space for the experiment of tossing a coin repeatedly with the outcomes "H" or "T".

Usage

```
tosscoin( times, makespace = FALSE )
```

Arguments

times   number of tosses.

makespace   logical.

Details

The function uses expand.grid() to generate all possible sequences of flips resulting from the experiment of tossing a coin. Columns of the dataframe are denoted toss1, toss2, up to tosstimes,

Value

A data frame, with an equally likely probs column if makespace is TRUE.

Author(s)

G. Jay Kerns <gkerns@ysu.edu>.

See Also

rolldie

Examples

```
tosscoin(2)
tosscoin(3, makespace = TRUE)
```

---

union *Union of Subsets*

Description

Calculates the union of subsets of a probability space.

Usage

```
union(x, ...)

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

## S3 method for class 'data.frame'
union(x, y, ...)

## S3 method for class 'ps'
union(x, y, ...)
```

Arguments

x, y	vectors, data frames, or ps objects containing a sequence of items (conceptually)
...	further arguments to be passed to or from other methods.

Details

This function operates row-wise on dataframes, and element-wise among the outcomes of ps objects. The elements of union(x,y) are those elements in x or y, or both. The definition is taken to match the version in the base package.

Value

A data frame or subset of a probability space of the same type as its arguments.

Author(s)

G. Jay Kerns <gkerns@ysu.edu>, based on a suggestion made by Brian Ripley in R-devel, 12/11/07.

See Also

[intersect](), [setdiff]()

Examples

```
S <- cards()
A <- subset(S, suit == "Heart")
B <- subset(S, rank == "A" )
union(A, B)
```

---

urnsamples   *Sampling from Urns*

Description

This function creates a sample space associated with the experiment of sampling distinguishable objects from an urn.

Usage

```
urnsamples(x, ...)

## Default S3 method:
urnsamples(x, size, replace = FALSE, ordered = FALSE, ...)

## S3 method for class 'data.frame'
urnsamples(x, size, replace = FALSE, ordered = FALSE, ...)
```

Arguments

x	a vector or data frame from which sampling should take place.
size	number indicating the sample size.
replace	logical indicating whether sampling should be done with replacement.
ordered	logical indicating whether order among samples is important.
...	further arguments to be passed to or from other methods.

Details

The function operates on the indices of the urn (or rows, in the case urn is a data frame). It then takes those samples and substitutes back into urn to generate the entries of the data frame (or list, respectively). In the case that urn has repeated values, the result will be repeated values in the output.

Note that urnsamples strips x of any existing probs column before sampling.

Value

A data frame if urn is a vector, and a list if urn is a data frame.

Author(s)

G. Jay Kerns <gkerns@ysu.edu>.

See Also

[iidspace](), [probspace nsamp]()

Examples

```
urnsamples(1:10, size = 5)
S <- cards()
urnsamples(S, size = 2)
```

Index