Package ‘probs’

July 23, 2025

Type Package

Title Elementary Probability on Finite Sample Spaces

Version 0.9.9

Date 2024-06-17

Description Performs elementary probability calculations on finite sample spaces, which may be represented by data frames or lists.
This package is meant to rescue some widely used functions from the archived 'prob' package (see <`https://cran.r-project.org/src/contrib/Archive/prob/`>).
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)

Encoding UTF-8

Language en-US

RoxygenNote 7.3.1

Imports stats, utils, combinat, MASS, reshape

NeedsCompilation no

Author G. Jay Kerns [aut, cph],
Joe gr. Schlarmann [cre]

Maintainer Joe gr. Schlarmann <`schlarmann@produnis.de`>

Repository CRAN

Date/Publication 2024-06-20 16:20:02 UTC

Contents

addrv 3
cards 4
countrep 5
countrep.data.frame 6
countrep.default 7

empirical 8
euchredeck 9
gen2wayTable 9
genIndepTable 10
genLogRegData 11
genXdata 12
iidspace 13
intersect 14
intersect.data.frame 15
intersect.default 16
intersect.ps 17
is.probspace 18
isin 19
isin.data.frame 20
isin.default 21
isrep 22
isrep.data.frame 23
isrep.default 24
marginal 25
noorder 26
nsamp 27
permsn 28
Prob 29
Prob.default 30
Prob.ps 31
probspace 32
probspace.default 33
probspace.list 34
rolldie 35
roulette 36
setdiff 37
setdiff.data.frame 38
setdiff.default 39
setdiff.ps 40
sim 41
subset.ps 42
tosscoin 43
union 44
union.data.frame 45
union.default 46
union.ps 47
urnsamples 48
urnsamples.data.frame 49
urnsamples.default 50

Index **51**

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. Include jokers in the deck.
`makespace`	logical. Include a column of equally likely probabilities.

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`.

Author(s)

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

See Also

`rolldie`, `tosscoin`, and `roulette`

Examples

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

`countrep`	*Count Repetitions Counts the number of repetitions of* `vals` *in a given vector* `x`*.*

Description

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

Usage

```
countrep(x, ...)
```

Arguments

`x`	an object in which repeats should be counted.
`...`	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.

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
```

`countrep.data.frame` *Count Repetitions Counts the number of repetitions of* `vals` *in a given vector* `x`.

Description

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

Usage

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

Arguments

`x`	an object in which repeats should be counted.
`...`	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.

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
```

`countrep.default` *Count Repetitions Counts the number of repetitions of* `vals` *in a given vector* `x`*.*

Description

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

Usage

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

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.

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. Include a Joker if `TRUE`.
`makespace`	logical. Include a column of equally likely probabilities if `TRUE`.

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

Description

Independent Identical Experiments 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 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`*.*

Description

Intersection of Subsets 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, ...)
```

Arguments

`x`	vectors, data frames, or `ps` objects containing a sequence of elements (conceptually).
`...`	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 elements 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)
```

`intersect.data.frame` *Intersection of Subsets 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`*.*

Description

Intersection of Subsets 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

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

Arguments

`x`	vectors, data frames, or `ps` objects containing a sequence of elements (conceptually).
`y`	vectors, data frames, or `ps` objects containing a sequence of elements (conceptually).
`...`	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 elements 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)
```

`intersect.default` *Intersection of Subsets 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`*.*

Description

Intersection of Subsets 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

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

Arguments

`x`	vectors, data frames, or `ps` objects containing a sequence of elements (conceptually).
`y`	vectors, data frames, or `ps` objects containing a sequence of elements (conceptually).
`...`	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 elements 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)
```

intersect.ps — *Intersection of Subsets 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`*.*

Description

Intersection of Subsets 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

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

Arguments

`x`	vectors, data frames, or `ps` objects containing a sequence of elements (conceptually).
`y`	vectors, data frames, or `ps` objects containing a sequence of elements (conceptually).
`...`	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 elements 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 Decides whether a given object is a probability space.*

Description

Testing for a Probability Space 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

Test Whether One Vector Is In Another Vector

Usage

```
isin(x, ...)
```

Arguments

`x`	vectors
`...`	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 multiplicity. 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 method 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))
```

isin.data.frame *Test Whether One Vector Is In Another Vector*

Description

Test Whether One Vector Is In Another Vector

Usage

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

Arguments

`x`	vectors
`...`	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 multiplicity. 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 method 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))
```

isin.default *Test Whether One Vector Is In Another Vector*

Description

Test Whether One Vector Is In Another Vector

Usage

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

Arguments

x	vectors
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 multiplicity. 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 method 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 Tests for a certain number of repetitions of* `vals` *in a given vector* `x`.

Description

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

Usage

```
isrep(x, ...)
```

Arguments

`x`	an object with potential repeated values.
`...`	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
```

`isrep.data.frame` *Is Repeated in a Vector Tests for a certain number of repetitions of* `vals` *in a given vector* `x`.

Description

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

Usage

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

Arguments

`x`	an object with potential repeated values.
`...`	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
```

isrep.default — *Is Repeated in a Vector Tests for a certain number of repetitions of* `vals` *in a given vector* `x`*.*

Description

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

Usage

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

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 Calculates probability and conditional probability of events.*

Description

Probability and Conditional Probability Calculates probability and conditional probability of events.

Usage

```
Prob(x, ...)
```

Arguments

`x`	a probability space or a subset of one.
`...`	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 )
```

Prob.default — *Probability and Conditional Probability Calculates probability and conditional probability of events.*

Description

Probability and Conditional Probability Calculates probability and conditional probability of events.

Usage

```
## Default S3 method:
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 )
```

`Prob.ps` *Probability and Conditional Probability Calculates probability and conditional probability of events.*

Description

Probability and Conditional Probability Calculates probability and conditional probability of events.

Usage

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

Description

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

Usage

```
probspace(x, ...)
```

Arguments

x	a vector, data frame, or list of 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>.

Examples

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

`probspace.default` *Probability Spaces Forms a probability space from a set of outcomes and (optional) vector of probabilities.*

Description

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

Usage

```
## Default S3 method:
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`>.

Examples

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

`probspace.list`	*Probability Spaces Forms a probability space from a set of outcomes and (optional) vector of probabilities.*

Description

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

Usage

```
## 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`>.

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. Include a column of equally likely probabilities if `TRUE`.

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. Use a European roulette wheel with 37 pockets if `TRUE`, otherwise use a standard US roulette wheel with 38 pockets.
`makespace`	logical. Include a column of equally likely probabilities if `TRUE`.

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

Description

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

Usage

```
setdiff(x, ...)
```

Arguments

`x`	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 `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)
```

`setdiff.data.frame` *Set Difference of Subsets Calculates the (nonsymmetric) set difference of subsets of a probability space.*

Description

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

Usage

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

Arguments

`x`	vectors, data frames, or `ps` objects containing a sequence of items (conceptually).
`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 `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)
```

setdiff.default — *Set Difference of Subsets Calculates the (nonsymmetric) set difference of subsets of a probability space.*

Description

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

Usage

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

Arguments

`x`	vectors, data frames, or `ps` objects containing a sequence of items (conceptually).
`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 `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)
```

setdiff.ps — *Set Difference of Subsets Calculates the (nonsymmetric) set difference of subsets of a probability space.*

Description

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

Usage

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

Arguments

`x`	vectors, data frames, or `ps` objects containing a sequence of items (conceptually).
`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 `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.

Simulates the experiment of drawing from a sample space.

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.
`...`	further arguments to be passed to or from other methods.
`ntrials`	number of times to repeat the experiment.

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`.

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`.

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`.

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

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

empirical

empirical

Examples

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

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

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

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

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

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

`subset.ps` *Subsets of Probability Spaces This is a method for* `subset()` *for the case when the input object is a probability space of class* `ps`.

Description

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

Usage

```
## 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`>.

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. Include a column of equally likely probabilities if `TRUE`.

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 Calculates the union of subsets of a probability space.*

Description

Union of Subsets Calculates the union of subsets of a probability space.

Usage

```
union(x, ...)
```

Arguments

`x`	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.

Examples

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

`union.data.frame` *Union of Subsets Calculates the union of subsets of a probability space.*

Description

Union of Subsets Calculates the union of subsets of a probability space.

Usage

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

Arguments

`x`	vectors, data frames, or `ps` objects containing a sequence of items (conceptually)
`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.

Examples

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

union.default — *Union of Subsets Calculates the union of subsets of a probability space.*

Description

Union of Subsets Calculates the union of subsets of a probability space.

Usage

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

Arguments

x	vectors, data frames, or ps objects containing a sequence of items (conceptually)
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.

Examples

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

union.ps — *Union of Subsets Calculates the union of subsets of a probability space.*

Description

Union of Subsets Calculates the union of subsets of a probability space.

Usage

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

Arguments

`x`	vectors, data frames, or `ps` objects containing a sequence of items (conceptually)
`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.

Examples

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

urnsamples — *Sampling from Urns This function creates a sample space associated with the experiment of sampling distinguishable objects from an urn.*

Description

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

Usage

```
urnsamples(x, ...)
```

Arguments

`x`	a vector or data frame from which sampling should take place.
`...`	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`>.

Examples

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

urnsamples.data.frame *Sampling from Urns This function creates a sample space associated with the experiment of sampling distinguishable objects from an urn.*

Description

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

Usage

```
## 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`>.

Examples

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

`urnsamples.default`	*Sampling from Urns This function creates a sample space associated with the experiment of sampling distinguishable objects from an urn.*

Description

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

Usage

```
## Default S3 method:
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`>.

Examples

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

Index

∗ **datagen**
  gen2wayTable, 9
  genIndepTable, 10
  genLogRegData, 11
  genXdata, 12
∗ **data**
  cards, 4
  euchredeck, 9
∗ **manip**
  addrv, 3
  empirical, 8
  marginal, 25
∗ **misc**
  noorder, 26
  nsamp, 27
  permsn, 28
  rolldie, 35
  roulette, 36
  sim, 41
  tosscoin, 43

addrv, 3, *25*, *26*

cards, 4, *9*, *36*
combn, *28*
countrep, 5, *22–24*
countrep.data.frame, 6
countrep.default, 7

empirical, 8, *42*
euchredeck, 9

gen2wayTable, 9
genIndepTable, 10
genLogRegData, 11
genXdata, 12

iidspace, 13, *29*, *30*, *32*
intersect, 14, *37–40*
intersect.data.frame, 15
intersect.default, 16
intersect.ps, 17
is.probspace, 18
isin, 19
isin.data.frame, 20
isin.default, 21
isrep, *5–7*, *19–21*, 22
isrep.data.frame, 23
isrep.default, 24

marginal, 25, *26*

noorder, 26
nsamp, 27

permsn, 28
Prob, 29
Prob.default, 30
Prob.ps, 31
probspace, *13*, *18*, *29*, *30*, *32*, 32
probspace.default, 33
probspace.list, 34

rolldie, *4*, 35, *44*
roulette, *4*, *9*, 36

setdiff, *14–17*, 37
setdiff.data.frame, 38
setdiff.default, 39
setdiff.ps, 40
sim, *8*, 41
subset.ps, 42

tosscoin, *4*, *9*, *35*, 43
transform, *3*

union, *14–17*, *37–40*, 44
union.data.frame, 45
union.default, 46
union.ps, 47
urnsamples, *27*, 48
urnsamples.data.frame, 49
urnsamples.default, 50