Package: Rdice (via r-universe)

October 31, 2024

Type Package

Title A Collection of Functions to Experiment Dice Rolls

Version 1.0.0

Author Gennaro Tedesco

Maintainer Gennaro Tedesco: <gennarotedesco@gmail.com>

Description A collection of functions to simulate dice rolls and the like. In particular, experiments and exercises can be performed looking at combinations and permutations of values in dice rolls and coin flips, together with the corresponding frequencies of occurrences. When applying each function, the user has to input the number of times (rolls, flips) to toss the dice. Needless to say, the more the tosses, the more the frequencies approximate the actual probabilities. Moreover, the package provides functions to generate non-transitive sets of dice (like Efron's) and to check whether a given set of dice is non-transitive with given probability.

Imports data.table

Suggests knitr, rmarkdown

VignetteBuilder knitr

License GPL-2

LazyData TRUE

RoxygenNote 5.0.1

NeedsCompilation no

Date/Publication 2016-09-24 12:58:32

Repository https://gennaro-tedesco.r-universe.dev

RemoteUrl https://github.com/cran/Rdice

RemoteRef HEAD

RemoteSha 46d15e0e201880149ef75de94ef386509db0b1e9

Contents

`coin.flip` *Coin flip*

Description

Simulates a coin flip.

Usage

```
coin.flip(coins = 5, flips = 100, weights = c(0.5, 0.5), getExact)
```

Arguments

`coins`	The number of coins to flip. If unspecified, it defaults to 5.
`flips`	The number of flips. If missing, it defaults to 100.
`weights`	A vector of probability weights to assign to each face of the coin; if unspecified, it defaults to a fair coin with equally likely faces. If specified, its lenght must obviously be a vector of length two whose values sum up to 1.
`getExact`	A vector containing values to be matched *exactly*, namely the function returns only those combinations containing *all* the above mentioned variables. Since this is a coin flip, values must be specified in the form `c("H", "H", "T")` as head and tails (make sure to provide the labels in quotation marks).

Details

The function is a particular case of `dice.roll`, namely a roll with 2 faces and 1 die.

Value

A table containing the frequencies for each of the two occurrences (head and tail) after the specified number of flips.

Examples

```
coin.flip(coins = 5, flips = 100)
```

`dice.combinations` *Dice combinations*

Description

Calculates all possible combinations as result of rolling a set of dice. Similar permutations are identified under the same combination and counted as many times as many occurrencies. Thee user can choose wheter to match exact values or to perform partial matches.

Usage

```
dice.combinations(faces, dice, rolls, weights, getPartial, getExact, toSum = FALSE)
```

Arguments

`faces`	The number of faces the dice have; if unspecified, it defaults to 6.
`dice`	The number of dice to roll; if unspecified, it defaults to 2.
`rolls`	The number of times to roll the die; if unspeciefid, it defaults to 5.
`weights`	A vector of probability weights to assign to each face of the die; if unspecified, it defaults to a fair die with weights $1/N$. If specified, its lenght must obviously be equals to the number of faces and all the single weights must sum up to 1.
`getExact`	A vector containing values to be matched *exactly*, namely the function returns only those combinations containing *all* the above mentioned variables.
`getPartial`	A vector containing values to be matched *partially*, namely the function returns only those combinations containing *at least one* of the above mentioned variables. If missing, it defaults to c(1:faces), namemly the function returns all combinations.
`toSum`	A logical value, defaulting to FALSE. If TRUE, the function returns the sum of the frequencies of the matches (to be used together with getExact or getPartial)

Details

The function returns an object of class *diceRoll*, namely a list whose values are themselves `data.table` objects, in turn, so that one can directly apply any `data.table` function thereupon.

Value

`values`	If toSum = FALSE, a list of all possible combinations rolled, together with corresponding frequencies. If toSum = TRUE, the function returns the sum of all frequencies in correspondence of matched valules.

Note

The case face = 2 corresponds to the `coin.flip`.

See Also

Makes use of `dice.roll`.

Examples

```
dice.combinations(faces = 6, dice = 4, rolls = 100,
    getExact = c(3,5), getPartial = c(1,2), toSum = TRUE)
```

`dice.roll` *Dice simulator*

Description

Simulates rolling of a set of dice.

Usage

```
dice.roll(faces, dice, rolls, weights)
```

Arguments

`faces`	The number of faces the dice have; if unspecified, it defaults to 6.
`dice`	The number of dice to roll; if unspecified, it defaults to 2.
`rolls`	The number of times to roll the die; if unspeciefid, it defaults to 5.
`weights`	A vector of probability weights to assign to each face of the die; if unspecified, it defaults to a fair die with weights $1/N$. If specified, its lenght must obviously be equals to the number of faces and all the single weights must sum up to 1.

Details

The function returns an object of class *diceRoll*, namely a list whose values are themselves `data.table` objects, in turn, so that one can directly apply any `data.table` function thereupon.

Value

`results`	The numerical results rolled.
`frequencies`	A table containing each single occurrency (permutation) of results and the corresponding frequencies.
`sums_freq`	A table containing the frequencies of the sums of the values obtained in each single roll by all the dice. A cumulative sum is provided too.
`exp_value_sum`	The expectation value of the sum of the values obtained.

Examples

```
dice.roll(faces = 6, dice = 3, rolls = 5)
```

efron — *The standard set of Efron's dice*

Description

A dataset containing the four standard Efron's dice, non-transitive set of dice with winning probabilities of 2/3.

Usage

```
data(efron)
```

Format

A data table with 4 columns. Each column represents a die with six faces.

Examples

```
data(efron)
is.nonTransitive(efron, prob = 2/3)
```

is.nonTransitive — *Checks truth value of non-transitive sets of dice.*

Description

Checks whether a given set of dice is non-transitive with given probability. If no probability is given, checks whether a given set of dice is generally non-transitive.

Usage

```
is.nonTransitive(df, prob)
```

Arguments

`df`	A `data.frame` containing the set of dice to be checked.
`prob`	The non-transitive probability according to which to check for non-transitivity. If unspecified, the function checks for general non-transitivity.

Value

A logical value: TRUE or FALSE.

See Also

See also `nonTransitive.generator`.

Examples

```
df <- data.frame(
  die1 = c(5,4,6,15),
  die2 = c(3,6,16,2),
  die3 = c(0,2,14,15),
  die4 = c(8,10,1,9)
)

is.nonTransitive(df, prob = 9/16)
```

`miwin` *Miwin set of dice.*

Description

A dataset containing the Miwin set of dice. This set is non-transitively losing, with losing probabilities of 17/36.

Usage

```
data(miwin)
```

Format

A data table with 3 columns. Each column represents a die with six faces.

Examples

```
data(miwin)
is.nonTransitive(miwin, prob = 17/36)
```

`nonTransitive.generator` *Non-transitive dice generator.*

Description

This function generates Z random non-transitive dice given the number faces and the corresponding non-transitive probabilities.

Usage

```
nonTransitive.generator(dice, faces, max_value = faces, prob, error = 0.001)
```

Arguments

`dice`	The number Z of non-transitive dice to generate.
`faces`	The number of faces of each die.
`max_value`	The maximum integer allowed as nominal value for the faces. Standard choices are usually `max_value = faces` (default) or `max_value = faces^2`.
`prob`	The probability one wants the set of dice to be non-transitive. If unspecified, a set of dice with different non-transitive probabilities for each pairing will be generated.
`error`	Computational error to check for machine precision equality. It defaults to 0.001: no need to be specified.

Details

The function randomly generates sets of dice and stops as soon as a non-transitive set of dice matching the specified conditions is found and only if so. As such, it might happen that it never returns any value, should the condition for non-transitive dice not be matched. One may need to manually interrupt the run (time delay is provided, one could set it as max threshold.)

Value

A `data.table` containing the set of non-transitive dice matching the specified conditions.

See Also

See also `is.nonTransitive`.

`oskar` *Oskar van Deventer set of dice.*

Description

A dataset containing the Oskar van Deventer dice, non-transitive set of dice where A beats B,C,E; B beats C,D,F; C beats D,E,G; #' D beats A,E,F; E beats B,F,G; F beats A,C,G; G beats A,B,D. #' Consequently, for arbitrarily chosen two dice there is a third one that beats both of them.

Usage

```
data(oskar)
```

Format

A data table with 6 columns. Each column represents a die with six faces.

Examples

```
data(oskar)
is.nonTransitive(oskar)
```

Index