

Package: dice (via r-universe)

August 21, 2024

Type Package

Title Calculate probabilities of various dice-rolling events

Version 1.2

Date 2014-10-13

Author Dylan Arena

Maintainer Dylan Arena <dylanarena1@gmail.com>

Description This package provides utilities to calculate the probabilities of various dice-rolling events, such as the probability of rolling a four-sided die six times and getting a 4, a 3, and either a 1 or 2 among the six rolls (in any order); the probability of rolling two six-sided dice three times and getting a 10 on the first roll, followed by a 4 on the second roll, followed by anything but a 7 on the third roll; or the probabilities of each possible sum of rolling five six-sided dice, dropping the lowest two rolls, and summing the remaining dice.

License GPL (>= 2)

Depends R (>= 2.0.0), gtools

NeedsCompilation no

Repository CRAN

Date/Publication 2014-10-14 08:25:25

Contents

dice-package	2
getEventProb	3
getSumProbs	4

Index	7
--------------	----------

`dice-package`*Calculate probabilities of various dice-rolling events*

Description

This package provides utilities to calculate the probabilities of various dice-rolling events, such as the probability of rolling a four-sided die six times and getting a 4, a 3, and either a 1 or 2 among the six rolls (in any order); the probability of rolling two six-sided dice three times and getting a 10 on the first roll, followed by a 4 on the second roll, followed by anything but a 7 on the third roll; or the probabilities of each possible sum of rolling five six-sided dice, dropping the lowest two rolls, and summing the remaining dice.

Details

Package: dice
Type: Package
Version: 1.2
Date: 2014-10-13
License: GPL (>= 2)

Although initially conceived as a utility for role-playing game calculations, functions in the dice package can be used to answer questions in any dice-rolling context (e.g., calculating probabilities for the game of craps, solving problems for an introductory probability course, etc.)

The dice package requires the `gtools` package.

For a complete list of functions, use `library(help="dice")`.

Author(s)

Dylan Arena <dylanarena1@gmail.com>

References

The implementation for the `getSumProbs` function originated with the ideas presented in the following forum thread:

<http://www.enworld.org/showthread.php?t=56352&page=1&pp=40>

Examples

```
getEventProb(nrolls = 6,  
             ndicePerRoll = 1,  
             nsidesPerDie = 4,  
             eventList = list(4, 3, c(1,2)),  
             orderMatters = FALSE)
```

```
getEventProb(nrolls = 3,
```

```

    ndicePerRoll = 2,
    nsidesPerDie = 6,
    eventList = list(10, 4, c(2:6, 8:12)),
    orderMatters = TRUE)

getSumProbs(ndicePerRoll = 5,
            nsidesPerDie = 6,
            nkept = 3,
            dropLowest = TRUE)

```

<code>getEventProb</code>	<i>Calculate the probability of a specified set of dice-rolling events</i>
---------------------------	--

Description

For a specified dice-rolling process, `getEventProb` calculates the probability of an event (i.e., a non-empty set of outcomes) that is specified by passing a list object in to `eventList`.

Usage

```
getEventProb(nrolls, ndicePerRoll, nsidesPerDie, eventList, orderMatters = FALSE)
```

Arguments

<code>nrolls</code>	A single positive integer representing the number of dice rolls to make
<code>ndicePerRoll</code>	A single positive integer representing the number of dice to use in each dice roll
<code>nsidesPerDie</code>	A single positive integer representing the number of sides on each die (<code>getEventProb</code> 's dice-rolling process involves only one type of die per call)
<code>eventList</code>	A list object, each element of which is a vector that constrains a single dice roll in the dice-rolling process (see Details below)
<code>orderMatters</code>	A logical flag indicating whether the order of the elements of <code>eventList</code> should constrain the event space; if <code>TRUE</code> , <code>eventList</code> must specify constraints for every dice roll—i.e., it must contain exactly <code>nrolls</code> elements (some of which may be "empty" constraints listing all possible outcomes of a dice roll, i.e., a vector from <code>ndicePerRoll</code> to $(\text{ndicePerRoll} * \text{nsidesPerDie})$)

Details

The crux of this function is `eventList`, which sets the conditions that acceptable dice-rolls must meet. E.g., to get the probability of rolling at least one 6 when rolling four six-sided dice, `eventList` would be `list(6)` and `orderMatters` would be `FALSE`; to get the probability of rolling a 6, followed by a 5, followed by either a 1, 2, or 3 when rolling three six-sided dice, `eventList` would be `list(6,5,1:3)` and `orderMatters` would be `TRUE`.

Value

A single number representing the probability of an event that meets the constraints of the specified dice-rolling process

Author(s)

Dylan Arena

Examples

```
## Probability of rolling at least one 6 when rolling four six-sided dice

getEventProb(nrolls = 4,
             ndicePerRoll = 1,
             nsidesPerDie = 6,
             eventList = list(6))

## Probability of rolling a 6, followed by a 5, followed by either a 1, 2,
## or 3 when rolling three six-sided dice

getEventProb(nrolls = 3,
             ndicePerRoll = 1,
             nsidesPerDie = 6,
             eventList = list(6, 5, 1:3),
             orderMatters = TRUE)

## Probability of rolling no 10's when rolling two ten-sided dice

getEventProb(nrolls = 2,
             ndicePerRoll = 1,
             nsidesPerDie = 10,
             eventList = list(1:9,1:9))
```

getSumProbs

Calculate the probabilities of all possible outcome sums of a dice roll

Description

For a specified number of dice with a specified number of sides per die (and dropping a specified number of dice—those with either the lowest or highest values), `getSumProbs` calculates the probabilities of all possible outcome sums (i.e., all possible sums of those dice whose results are not dropped); the function also accommodates modifiers (either to each die roll or to the sum), such as rolling five four-sided dice and adding 1 to the outcome of each roll, or rolling one twenty-sided die and adding 12 to the outcome. (Such modified rolls frequently occur in the context of role-playing games, e.g., Dungeons & Dragons, Mutants & Masterminds, or BESM.)

Usage

```
getSumProbs(ndicePerRoll,
            nsidesPerDie,
```

```
nkept = ndicePerRoll,  
dropLowest = TRUE,  
sumModifier = 0,  
perDieModifier = 0,  
perDieMinOfOne = TRUE)
```

Arguments

ndicePerRoll	A single positive integer representing the number of dice to roll
nsidesPerDie	A single positive integer representing the number of sides on each die (getSumProbs's dice-rolling process involves only one type of die per call)
nkept	A single positive integer representing the number of dice whose values to include when calculating the sum (the dice to be kept will always be those with the highest values)
dropLowest	A single logical indicating whether to drop the lowest outcome values (FALSE drops the highest values instead)
sumModifier	A single integer representing an amount to add to or subtract from the outcome sum
perDieModifier	A single integer representing an amount to add to or subtract from each die roll
perDieMinOfOne	A logical flag indicating whether each die roll should be considered to have a minimum value of 1 (as is often true in role-playing-game contexts)

Value

probabilities	A matrix with a row for each possible outcome sum and three columns: one that lists each sum, one for the probability of that sum, and one for the number of ways to roll that sum
average	A single number representing the expected value of the specified dice-rolling process

Author(s)

Dylan Arena

References

This function's implementation originated with the ideas presented in the following forum thread:

<http://www.enworld.org/showthread.php?t=56352&page=1&pp=40>

Examples

```
## Rolling four six-sided dice and keeping the three highest die rolls  
  
getSumProbs(ndicePerRoll = 4,  
            nsidesPerDie = 6,  
            nkept = 3)
```

```
## Rolling five four-sided dice and adding 1 to each die roll

getSumProbs(ndicePerRoll = 5,
            nsidesPerDie = 4,
            perDieModifier = 1)

## Rolling one twenty-sided die and adding 12 to the result

getSumProbs(ndicePerRoll = 1,
            nsidesPerDie = 20,
            sumModifier = 12)
```

Index

* **distribution**

 getEventProb, 3

 getSumProbs, 4

* **package**

 dice-package, 2

dice (dice-package), 2

dice-package, 2

getEventProb, 3

getSumProbs, 2, 4