Package 'droll'

May 8, 2026

Title Analyze Roll Distributions

Version 0.1.0

Description A toolkit for parsing dice notation, analyzing rolls, calculating success probabilities, and plotting outcome distributions.

License MIT + file LICENSE

Depends R (>= 3.3.0)

Imports methods, Ryacas

Suggests covr, ggplot2, mockery, testthat (>= 3.0.0), tibble, vdiffr

Config/testthat/edition 3

Encoding UTF-8

RoxygenNote 7.1.1

NeedsCompilation no

Author C. Lente [aut, cre], Curso-R [cph, fnd]

Maintainer C. Lente <clente@curso-r.com>

Repository CRAN

Date/Publication 2021-08-06 17:50:09 UTC

Contents

check 2
d 3
Dice 4
r 6
roll 7
roll-plot 8

Index **10**

check *Work with skill checks' probabilities and DCs*

Description

These are simple functions that manipulate probabilities and DCs for skill checks. `check_prob()` calculates the success/failure probability of a check with the given DC. `check_dc()` calculates the required difficulty class so that a skill check has the given success/failure probability. See below for more details.

Usage

```
check_dc(roll, p, success = TRUE)

check_prob(roll, dc, success = TRUE)
```

Arguments

`roll`	A roll expression (e.g., `2 * d6 + 5`) or a data frame returned by `r()`.
`p`	The probability of success/failure of the check (or attack).
`success`	Whether to aim for success (the default) or for failure on the check (or attack).
`dc`	The difficulty class to beat for a skill check (or the armor class to beat for an attack).

Details

These functions hide the complexities of the roll family so users unfamiliar with R's d/p/q/r notation can get quickly up and running with the package. Since attacks and checks work in the same way (i.e., success means rolling a value higher than or equal to a certain threshold), there are no `attack_*()` functions.

For more details on roll expressions, see `r()` and the Dice S4 class.

Value

A numeric scalar.

See Also

roll, `r()`, Dice.

Examples

```
# Probability of d20 + 8 passing a DC 15 skill check
check_prob(d20 + 8, 15)

# Probability of d20 + 8 missing an AC 15 attack
check_prob(d20 + 8, 15, success = FALSE)
```

d *Create a die*

Description

Create an instance of the Dice S4 class, allowing for the use of non-standard dice and for interactive dice rolling without having to recur to functions of the roll family. See below for more details.

Usage

```
d(faces)
```

Arguments

faces	Either the number of faces (`length(faces) == 1`) or a numeric vector specifying the values of the faces.

Details

If given a numeric vector, `d()` creates an object of the Dice S4 class representing a die with these values for faces. On the other hand, if given a numeric scalar, it creates a die with faces running from 1 to this value. At the moment, there is no support for specifying each face's probability, although it is possible to create a die where more than one face have the same value.

This function has two main purposes: creating non-standard dice and allowing for interactive dice rolling. Non-standard dice are arbitrary objects that might not have a real world counterpart, e.g., a `d(17)` or a `d(c(1, 1, 3))`. Interactive rolling is the ability to get a random result from a die without having to resort to functions of the roll family, explained in detail in the documentation for the Dice S4 class.

Value

An object of the Dice S4 class.

See Also

Dice, roll.

Examples

```
# Create a d6
d6 <- d(6)
d6

# Create a die with even faces
dEven <- d(c(2, 4, 6))
dEven

# Create a loaded die
```

```
dLoaded <- d(c(1:6, 6))
dLoaded
```

`Dice` *An S4 class to represent dice*

Description

A virtual representation of a die that supports the same arithmetic operations as a numeric scalar, with the special property that, when operated on, its value is randomly sampled from the die's faces. See below for more details.

Usage

```
## S4 method for signature 'Dice'
show(object)

## S4 method for signature 'Dice,numeric'
Ops(e1, e2)

## S4 method for signature 'numeric,Dice'
Ops(e1, e2)

## S4 method for signature 'Dice,Dice'
Ops(e1, e2)

## S4 method for signature 'numeric,Dice'
e1 * e2

## S4 method for signature 'Dice'
Math(x)

## S4 method for signature 'Dice'
Math2(x, digits)

## S4 method for signature 'Dice'
Summary(x, ..., na.rm = FALSE)
```

Arguments

`object`	A Dice object.
`e1`	A numeric scalar or a Dice object.
`e2`	A numeric scalar or a Dice object.
`x`	A Dice object
`digits`	Number of digits to be used in `round()` or `signif()`.
`...`	Numeric arguments.
`na.rm`	A logical indicating whether missing values should be removed.

Details

This S4 class extends numeric with the goal of creating an interactive die inside of R. In short, an instance of this class functions as a numeric scalar for most intents and purposes except that, when its value is needed, it returns one of its faces at random.

For more information on exactly what operations are supported, see the **Operations** section below. To learn more about how to create an object of this class, see the dice creating function `d()`. For roll distributions, see the roll family. For plotting those distributions, see the roll-plot family.

Slots

`faces` A numeric vector with the die's faces.

Operations

By default, when printed, an object of this class returns a numeric vector with all of its faces. In order to actually "roll" the die (that is, get one of its faces at random), one can simply operate on it. Any arithmetic expression should trigger a die into sampling its faces, even `dX + 0` and `1 * dX`.

All standard arithmetic operations are supported, along with comparisons, logic assertions, mathematical functions, and summaries: every group described in S4groupGeneric except for `Complex`. Note that, when used in other situations, like `c()`, the die will return all of its faces.

These functions also work in the exact same way as they would with regular numeric scalars, with the exception of multiplication. With the goal of supporting the very common operation `NdX` ("rolling `N` dice with `X` faces and adding the results"), the multiplication symbol behaves differently depending on the context: `N * dX` will work as `NdX` and `dX * N` will work as `N x dX` ("rolling a die with `X` faces and multiplying the result by `N`").

The roll and roll-plot families of functions make ample use of roll expressions like the ones described here. They even support some built-in dice that can be used without being created with `d()`.

See Also

`d()`, roll, roll-plot.

Examples

```
set.seed(42)

# Create some dice with d()
d6 <- d(6)
d20 <- d(20)

# Print faces from die
d6

# Roll 1d6
1 * d6

# Check if an attack hits and deal damage
if (d20 + 8 >= 12) {
```

```
  print(4 * d6)
} else {
  print(0)
}
```

`r` *Get full distribution of a roll*

Description

Return a data frame with most information necessary to work with the roll distribution: a column containing the possible outcomes of the roll expression, a column counting how many different ways each outcome can be obtained, a column with the associated densities, and a column with the associated probabilities. See below for more details.

Usage

```
r(roll, precise = FALSE)
```

Arguments

`roll`	A roll expression (e.g., `2 * d6 + 5`) or a data frame returned by `r()`.
`precise`	Whether to return values with arbitrary precision.

Details

A roll expression is a piece of R code that describes a dice roll with or without modifiers, e.g., `2 * d6 + 5`. This function processes roll expressions in order to power both the roll and roll-plot family of functions. Given an expression of this form it calculates the complete distribution of the outcomes. This is possible because the distribution is discrete and has a finite number of outcomes.

Standard dice notation should mostly work out of the box, with the notable exception of `NdX`, i.e., "roll `N` dice with `X` faces and add the results". In this case, the user must write `N * dX`; this also means that, when translating "roll a die with `X` faces and multiply the result by `N`" to a roll expression, the user must then write `dX * N`. All other expressions involving dice can usually be pasted straight into these functions.

For more details on what operations are supported, see the Dice S4 class.

Value

A data frame with four columns: `outcome`, `n`, `d`, and `p`.

Built-in Dice

It is possible to define any die with `d()`, but some are already built-in. Because of R's restrictions on what kind of object can be exported, they are not readily available for the user, but can be used inside a roll expression nonetheless These are the standard D&D dice: `d4`, `d6`, `d8`, `d10`, `d12`, `d20`, and `d100`.

Arbitrary Precision

Most dice programs that can calculate probabilities are forced to round their results due to the fact that these quantities might become exceptionally low when dealing with a lot of dice. This, however, can lead to error magnification.

In order to avoid rounding as much as possible, all functions described here use `Ryacas::yac_str()` to run computations symbolically. By default, results are converted to numeric vectors just before returning to the user, but one is able to access the symbolic strings returned by Ryacas by setting `precise = TRUE`.

See Also

roll, roll-plot, Dice.

Examples

```
# Get full distribution of 2d6 + 5
r(2 * d6 + 5)
```

`roll` *The roll distribution*

Description

Density, distribution function, quantile function, and random generation for the discrete distribution described by a roll expression. See below for more details.

Usage

```
droll(x, roll)

proll(q, roll, lower.tail = TRUE)

qroll(p, roll, lower.tail = TRUE)

rroll(n, roll)
```

Arguments

`x`	A numeric vector of outcomes.
`roll`	A roll expression (e.g., `2 * d6 + 5`) or a data frame returned by `r()`.
`q`	A numeric vector of outcomes.
`lower.tail`	Whether to calculate `P[X <= x]` or `P[X > x]`.
`p`	A numeric vector of probabilities.
`n`	Number of random deviates to return.

Details

Given a roll expression (i.e., an arithmetic expression involving dice), `r()` calculates the complete distribution of the outcomes. This is possible because the distribution is discrete and has a finite number of outcomes.

From this distribution, `droll()` returns the density, `proll()` returns the distribution function, `qroll()` returns the quantile function, and `rroll()` generates random deviates. They mirror functions from the Distributions family.

For more details on roll expressions, see `r()` and the Dice S4 class.

Value

A numeric vector.

Source

The main algorithm for calculating dice probabilities comes from MathWorld.

Symbolic calculations are handled by Ryacas, and, by extension, by Yacas.

See Also

`r()`, Dice, roll-plot.

Examples

```
set.seed(42)

# Density of 2d6 + 5
droll(12, 2 * d6 + 5)

# Distribution function of 2d6 + 5
proll(12, 2 * d6 + 5)

# Quantile function of 2d6 + 5
qroll(0.5, 2 * d6 + 5)

# Roll 2d6 + 5 (generate random deviate)
rroll(1, 2 * d6 + 5)
```

`roll-plot` *Plot the roll distribution*

Description

Plot density, distribution function, quantile function, and random generation for the discrete distribution described by a roll expression. See below for more details.

Usage

```
droll_plot(roll, ...)

proll_plot(roll, lower.tail = TRUE, ...)

qroll_plot(roll, lower.tail = TRUE, ...)

rroll_plot(n, roll, ...)
```

Arguments

`roll`	A roll expression (e.g., `2 * d6 + 5`) or a data frame returned by `r()`.
`...`	Other arguments passed on to plotting functions (`graphics::barplot()` or `ggplot2::qplot()` if available).
`lower.tail`	Whether to calculate `P[X <= x]` or `P[X > x]`.
`n`	Number of random deviates to return.

Details

Given a roll expression (i.e., an arithmetic expression involving dice), `r()` calculates the complete distribution of the outcomes. This is possible because the distribution is discrete and has a finite number of outcomes.

From this distribution, `droll_plot()` plots the density, `proll_plot()` plots the distribution function, `qroll_plot()` plots the quantile function, and `rroll_plot()` plots random deviates.

For more information, see the generating functions: roll.

Value

For `droll_plot()`, `proll_plot()`, and `qroll_plot()` a bar plot. For `rroll_plot()` a histogram.

See Also

`graphics::barplot()`, `ggplot2::qplot()`, `d()`, roll

Examples

```
set.seed(42)

# Density of 2d6 + 5
droll_plot(2 * d6 + 5)

# Distribution function of 2d6 + 5
proll_plot(2 * d6 + 5)

# Quantile function of 2d6 + 5
qroll_plot(2 * d6 + 5)

# Roll 2d6 + 5
rroll_plot(1000, 2 * d6 + 5)
```

Index

*,numeric,Dice-method *(Dice)*, 4

c(), *5*
check, 2
check_dc *(check)*, 2
check_dc(), *2*
check_prob *(check)*, 2
check_prob(), *2*

d, 3
d(), *3*, *5*, *6*, *9*
Dice, *2*, *3*, 4, *6–8*
Distributions, *8*
droll *(roll)*, 7
droll(), *8*
droll_plot *(roll-plot)*, 8
droll_plot(), *9*

ggplot2::qplot(), *9*
graphics::barplot(), *9*

Math,Dice-method *(Dice)*, 4
Math2,Dice-method *(Dice)*, 4

numeric, *5*

Ops,Dice,Dice-method *(Dice)*, 4
Ops,Dice,numeric-method *(Dice)*, 4
Ops,numeric,Dice-method *(Dice)*, 4

proll *(roll)*, 7
proll(), *8*
proll_plot *(roll-plot)*, 8
proll_plot(), *9*

qroll *(roll)*, 7
qroll(), *8*
qroll_plot *(roll-plot)*, 8
qroll_plot(), *9*

r, 6
r(), *2*, *6–9*
roll, *2*, *3*, *5–7*, 7, *9*
roll-plot, *5–8*, 8
round(), *4*
rroll *(roll)*, 7
rroll(), *8*
rroll_plot *(roll-plot)*, 8
rroll_plot(), *9*
Ryacas::yac_str(), *7*

S4groupGeneric, *5*
show,Dice-method *(Dice)*, 4
signif(), *4*
Summary,Dice-method *(Dice)*, 4