Package 'rcube'

July 23, 2025

Type Package

Title Simulations and Visualizations of Rubik's Cube (with Mods)

Version 0.5

Date 2019-05-15

Author Wojciech Rosa

Maintainer Wojciech Rosa <w.rosa@pollub.pl>

Description

Provides simplified methods for managing classic Rubik's cubes and many other modifications of it (such as NxNxN size cubes, void cubes and 8-coloured cubes - so called octa cubes). Includes functions of handling special syntax for managing such cubes; and different approach to plotting 3D cubes without using external libraries (for example 'OpenGL').

License GPL-3

Imports magrittr

LazyData TRUE

RoxygenNote 6.1.1

NeedsCompilation no

Repository CRAN

Date/Publication 2019-05-15 14:00:04 UTC

Contents

createCube 2
is.solved 3
plot.cube 4
plot3dCube 4
plot3dFlat 5
positions 6
print.cube 6
scramble 7
translate 7
twistCube 8

Index **10**

createCube *Creating cube size NxNxN*

Description

Creates a cube object with empty moves and color scheme information

Usage

```
createCube(N = 3, mode = "normal", scheme = c("orange", "yellow",
  "blue", "white", "green", "red"))
```

Arguments

N	integer - size of cube. Default value is 3, and minimum is 1.
mode	string "normal" (default) or "octa" or "void". There are also are also available learning modes. Use keywords: "cross","first layer", "first two layers", "corners", "edges" to obtain your mode. For example "cross,centers", "corners" or "edges and centers". Default color scheme is the same as defined in parameter with added gray color.
scheme	string vector - colour scheme for plotting cube. Name of colours should be given in specified order: front, top, right, bottom, left, back. In learning mode there is possibility to define 12 colors (standard 6 colors and 6 which are default gray). Default value is c("orange","yellow","blue","white","green","red")

Value

Cube class object

Examples

```
# Create 3x3x3 cube with original color scheme:
cube <- createCube()
# Create 14x14x14 cube with original color scheme:
cube <- createCube(N = 14)
# Create 3x3x3 cube with "japanese" color scheme:
cube <- createCube(scheme = c("green","white","red","blue","orange","yellow"))
# Create 3x3x3 learning cubes:
c <- createCube(mode = "cross and centers")
c2 <- createCube(mode = "first layer and centers")
c3 <- createCube(mode = "first two layers")
# Create cube with coloured corners and edges:
c <- createCube(N = 4, mode = "corners and edges")
```

`is.solved` *Testing if cube is solved*

Description

Function returns TRUE if cube is solved (each side contains exactly one colour) and FALSE otherwise.

Usage

```
is.solved(cube)
```

Arguments

cube	- cube object to be tested

Value

TRUE/FALSE

Examples

```
## Create new cube:
cube <- createCube(3)
## And it is solved:
is.solved(cube) # TRUE
## Now, test how many times repeating LFRB moves will bring back initial state:
cube <- twistCube(cube,'LFRB')
i <- 1
while(!is.solved(cube))
{
cube <- twistCube(cube,'LFRB')
i <- i + 1
}
print(i) # 315
## Check one more time if this is a solution:
is.solved(twistCube(cube,'LFRB',315)) # TRUE
## Check if really 314 moves and 316 moves don't give solution:
is.solved(twistCube(cube,'LFRB',314)) || is.solved(twistCube(cube,'LFRB',316)) # FALSE
```

plot.cube *Plotting cube*

Description

Plots cube in 2D

Usage

```
## S3 method for class 'cube'
plot(x, ...)
```

Arguments

x	- cube object
...	- not used

Value

plot

Examples

```
cube <- createCube()
plot(cube)
# using pipe
require(magrittr)
createCube() %>% plot()
```

plot3dCube *Plotting cube in 3D*

Description

Plotting cube in 3D

Usage

```
plot3dCube(cube, sides = "both", rotate = "O")
```

Arguments

cube	- cube object
sides	- string parameter determining which side of cube should be plotted, correct values are: top, bottom, and both (default).
rotate	- string defaulting initial rotating of cube. Correct are strings containing characters: o, O, p, P. Default is 'O'

Value

plot

Examples

```
cube <- createCube()
plot3dCube(cube) # generates plot of solved cube
# 'checkerboard' pattern
require(magrittr)
cube %>% twistCube("(LLFFRRBB) x3") %>% plot3dCube()
```

`plot3dFlat` *Plotting cube in 2D*

Description

Plotting cube in 2D, but holds 3D advantages

Usage

```
plot3dFlat(cube)
```

Arguments

cube	- cube object

Value

plot

Examples

```
cube <- createCube()
plot3dFlat(cube) # generates plot of solved cube
# Plotting 'checkerboard' pattern using pipe:
require(magrittr)
createCube() %>% twistCube("(LLFFRRBB) x3") %>% plot3dFlat()
```

positions *Positions dataset*

Description

A dataset containing famous cubes positions. The variables are as follows:

Usage

```
data(positions)
```

Format

A data frame with popular positions of cubes

Details

- n: size of cube
- name: pattern name
- moves: moves to make

print.cube *Printing cube*

Description

Prints cube in console

Usage

```
## S3 method for class 'cube'
print(x, ...)
```

Arguments

x	- cube object
...	- not used

Value

plain text

Examples

```
cube <- createCube()
print(cube)
```

scramble *Scrambling cube*

Description

Scrambling cube

Usage

```
scramble(cube, times = 0)
```

Arguments

cube	- cube object to scramble
times	- how many random moves should be done on cube. Default is 0 which means N*10 moves where N is the size of the cube.

Value

cube

Examples

```
cube <- createCube()
set.seed(1)
cube <- scramble(cube)
bigcube <- createCube(N = 15)
set.seed(1)
bigcube <- scramble(bigcube)
```

translate *Translating notation*

Description

Translating notation

Usage

```
translate(moves, from = "singmaster")
```

Arguments

moves	- cube object
from	- Singmaster

Value

moves

Examples

```
cube <- createCube()
cube <- twistCube(cube, moves = translate("U R2 F B R B2 R U2 L B2 R U' D' R2 F R' L B2 U2 F2 "))
# Superflip pattern, https://en.wikipedia.org/wiki/Superflip
plot3dCube(cube)
```

`twistCube` *Twist cube*

Description

Twist the cube by given string of moves and number of times.

Usage

```
twistCube(cube, moves = "", times = 1)
```

Arguments

`cube`	- cube object
`moves`	- string parameter Syntax: The main QTM clockwise movements are the same as in the Singmasters notation: "U", "D", "F", "B", "R", "L". However moves from HTM such as U2 is not move of upper layer by 180 degrees (it will be explained further). Counter clockwise moves are denoted by lowercase letters: "u", "d", "f", "b", "r", "l". Rotations of the cube are denoted by "O" (rotate cube horizontally, "o" means rotation horizontally in different direction); and "P" (rotate cube vertically, "p" means rotation vertically in different direction). Repetitions of the moves: there are several ways to repeat given sequence of moves. The simplest way is to copy commands. The most effective way to do this is using parameter times. However, in some cases it is useful to repeat only parts of sequence of moves - then we could use bracketing terms and operator times "x".
`times`	- integer (default is 1). Number of repetitions of moves.

Value

cube - cube object

Examples

```
# Create classic Rubik's cube:
c <- createCube()
# Check moves LL FF RR BB
c <- twistCube(c,"LLFFRRBB")
# Check if LFRB repeated 316 times is cycle:
c <- twistCube(c,"(LFRB)x316")
is.solved(c)
# TRUE
# Twisted chicken feet pattern:
c <- createCube()
c <- twistCube(c,positions[21,"moves"])
plot3dCube(c)
# The same pattern using pipe %>% from magrittr package
require(magrittr)
createCube() %>% twistCube(positions[21,"moves"]) %>% plot3dCube()
# Rubik's Revenge
createCube(N = 4) %>% plot3dCube()
# Creating Professor's Cube
createCube(N = 5) %>% plot3dCube()
# Rotating and moving edges:
createCube(N = 5) %>% twistCube("(u3RUrFrfRU3)x12") %>% plot3dCube()
# Moving and rotating edges part 2:
createCube(5) %>% twistCube("((R1:2)x2 BBUU (L1:2)x2 UU rr2
UU RR2 UUFF RR2 FF ll2 BB (R1:2)x2 )x2 dd") %>% plot3dCube()
# Hearts pattern on a cube sized 13x13x13:
createCube(13) %>% twistCube("OP U2
l4:5 R4:5 u2 L4:5 r4:5 U3
l3:6 R3:6 u3 L3:6 r3:6 U4
l2:4 R2:4 l6:8 u4 L2:4 r2:4 L6:8 U5
l2:3 R2:3 l7 u5 L2:3 r2:3 L7 U6
l2:3 R2:3 u6 L2:3 r2:3 U7
l2:4 R2:4 u7 L2:4 r2:4 U8
l3:5 R3:5 u8 L3:5 r3:5 U9
l4:6 R4:6 u9 L4:6 r4:6 d4 l5:9 D4
L5:9 d3 l6:8 D3
L6:8 d2 l7 D2 L7") %>% plot3dCube()
# Creating octa cube
createCube(N = 4, mode = "octa") %>% plot3dCube()
# Rotating centers which is not visible on a classic cube (URL algorithm):
createCube(N = 4, mode = "octa") %>% twistCube("(URLuurl)x2") %>% plot3dCube()
# Creating void cube 8x8x8
createCube(N = 8,mode = "void") %>% plot3dCube()
```

Index

∗ **datasets**
positions, 6

createCube, 2

is.solved, 3

plot.cube, 4
plot3dCube, 4
plot3dFlat, 5
positions, 6
print.cube, 6

scramble, 7

translate, 7
twistCube, 8