Package ‘lightsout’

August 29, 2016

Title Implementation of the 'Lights Out' Puzzle Game

Version 0.3

Description Lights Out is a puzzle game consisting of a grid of lights that are either on or off. Pressing any light will toggle it and its adjacent lights. The goal of the game is to switch all the lights off. This package provides an interface to play the game on different board sizes, both through the command line or with a visual application. Puzzles can also be solved using the automatic solver included. View a demo online at http://daattali.com/shiny/lightsout/.

URL https://github.com/daattali/lightsout

BugReports https://github.com/daattali/lightsout/issues

Depends R (>= 3.0.0)

Imports magrittr (>= 1.5), shiny (>= 0.10.0), shinyjs (>= 0.3.0), stats, utils

Suggests knitr (>= 1.7), testthat (>= 0.9.1), rmarkdown

License MIT + file LICENSE

SystemRequirements pandoc with https support

LazyData true

VignetteBuilder knitr

RoxygenNote 5.0.1

NeedsCompilation no

Author Dean Attali [aut, cre]

Maintainer Dean Attali <daattali@gmail.com>

Repository CRAN

Date/Publication 2016-07-26 12:09:35
R topics documented:

- board_entries ................................................................. 2
- empty_board ................................................................. 3
- is_solvable ................................................................. 3
- is_solved ................................................................. 4
- launch ................................................................. 5
- lightsout ................................................................. 5
- new_board ................................................................. 5
- play ................................................................. 6
- random_board ............................................................... 8
- solve_board ............................................................... 9

Index 10

---

**board_entries**

*Get the board entries (configuration of the lights)*

**Description**

Get the board entries (configuration of the lights)

**Usage**

```r
code

boardEntries(board)
```

**Arguments**

- `board` A lightsout board object

**Value**

A matrix representing the current state of the lights (0 for off, 1 for on) in the board

**Examples**

```r
code

board <- random_board(5)
board
boardEntries(board)
```
empty_board

Initialize a Lights Out board with all lights switched off

Usage

empty_board(size, classic = TRUE)

Arguments

size Number of rows and columns for the board
classic If TRUE, then pressing a light will toggle it and its adjacent neighbours only. If FALSE, then pressing a light will toggle the entire row and column of the pressed light.

Value

A lightsout board.

See Also

random_board new_board

Examples

empty_board(5)

is_solvable

Is a given Lights Out board solvable?

Description

Not every Lights Out configuration has a solution (this has been mathematically proven). This function determines whether a given board has a solution or not.

Usage

is_solvable(board)

Arguments

board A lightsout board
is_solved

Value

TRUE if the given board has a solution; FALSE otherwise.

See Also

is_solved solve_board

Examples

# The following board is solvable using the classic mode (only adjacent lights # are toggled), but has no solution in the variant mode.
lights <- c(1L, 1L, 0, 1, 0, 0, 0, 0, 0)
board_classic <- new_board(lights)
board_variant <- new_board(lights, classic = FALSE)
is_solvable(board_classic)
is_solvable(board_variant)

is_solved Is the given board is a solved state?

Description

A board is considered solved if all the lights are switched off (have a state of 0).

Usage

is_solved(board)

Arguments

board A lightsout board

Value

TRUE if the given board is solved; FALSE otherwise.

See Also

is_solvable solve_board
Examples

# Create a board solved with one move and solve it.
lights <- c(1, 1, 0,
           1, 0, 0,
           0, 0, 0)
board <- new_board(lights)
is_solved(board)
board <- board %>% play(1, 1)
is_solved(board)

launch

Run the graphical interface to the game in a web browser

Description

Run the graphical interface to the game in a web browser

Usage

launch()

lightsout

lightsout package - Implementation of the ‘Lights Out’ Puzzle Game

Description

Lights Out is a puzzle game consisting of a grid of lights that are either on or off. Pressing any light will toggle it and its adjacent lights. The goal of the game is to switch all the lights off. This package provides an interface to play the game on different board sizes, both through the command line or with a visual application. Puzzles can also be solved using the automatic solver included. View a demo online or see the full README on GitHub.

new_board

Initialize a Lights Out board with a given lights configuration

Description

Create a Lights Out board that can be played by the user or solved automatically. Only square boards of size 3x3, 5x5, 7x7, or 9x9 are supported. The initial lights configuration must be provided. To create a board with a random configuration, use the random_board function.

Usage

new_board(entries, classic = TRUE)
Arguments

**entries**
The initial configuration of lights on the board. entries can either be a vector or a matrix. If a vector is used, the vector is assumed to start at the top-left corner of the board and is read row-by-row. Only values of 0 (light off) and 1 (light on) are allowed in the vector or matrix. See the examples below.

**classic**
If TRUE, then pressing a light will toggle it and its adjacent neighbours only. If FALSE, then pressing a light will toggle the entire row and column of the pressed light.

Value
A lightsout board object.

See Also

random_board play solve_board

Examples

```r
vector <- c(1L, 1L, 0, 1, 0, 1, 0, 1, 1)
new_board(entries = vector)

matrix <- matrix(
  c(1L, 1L, 0, 1, 0, 1, 0, 1, 1),
  nrow = 3, byrow = TRUE)
new_board(entries = matrix)
```

---

**play**
Play (press) a single light or multiple lights on a board

**Description**

In classic mode, pressing a light will toggle it and its four adjacent lights. In variant mode, pressing a light will toggle it and all other lights in its row and column. Toggling a light means switching it from on to off or from off to on.

**Usage**

```r
play(board, row, col, matrix)
```
play

Arguments

board       A lightsout board
row         The row of the light to press. To press multiple lights, use a list of row numbers. If a list is provided, then the col argument must also be a list of the same length.
col         The column of the light to press. To press multiple lights, use a list of column numbers. If a list is provided, then the row argument must also be a list of the same length.
matrix      Instead of using row and col, a matrix can be used to specify which lights to press. The matrix must have the same dimensions as the board. Any position in the given matrix with a value of 1 will result in a press of a light in the same position in the board.

Value

A new lightsout board object after the given lights are pressed.

See Also

solve_board empty_board new_board random_board

Examples

# Create a 5x5 board with all lights switched off and then press some lights
board <- empty_board(5)
board

# Press the light at (2,1)
newboard <- play(board, 2, 1)
newboard

# Press the light at (2,1) and then at (3,4)
newboard <- board %>% play(2, 1) %>% play(3, 4)
newboard

# Press both lights with one call
newboard <- play(board, c(2, 3), c(1, 4))
newboard

# Press both lights using a matrix instead of specifying rows and columns
newboard <- play(board, matrix = matrix(
c(0, 0, 0, 0, 0,
1, 0, 0, 0, 0,
0, 0, 0, 1, 0,
0, 0, 0, 0, 0,
0, 0, 0, 0, 0),
nrow = 5, byrow = TRUE))
newboard

# Press the same lights, but this time when the game mode is not classic,
random_board

Description
Create a Lights Out board that can be played by the user or solved automatically. Only square boards of size 3x3, 5x5, 7x7, or 9x9 are supported. The initial lights configuration is randomly generated, but always solvable. To create a board with a user-defined configuration, use the new_board function.

Usage
random_board(sizeL classic = TRUE)

Arguments
size Number of rows and columns for the board
classic If TRUE, then pressing a light will toggle it and its adjacent neighbours only. If FALSE, then pressing a light will toggle the entire row and column of the pressed light.

Value
A lightsout board object.

See Also
new_board play solve_board

Examples
set.seed(10)

# Create a random 5x5 classic board
board <- random_board(5)
board

# Get the solution for the board
solution <- solve_board(board)
solution

# Press the lights according to the solution, the result should be a board
# with all lights switched off
play(board, matrix = solution)
solve_board

Solve a Lights Out board

Description

Given a Lights Out board, find the set of lights that need to be pressed in order to solve the board. If no solution is possible, an error is thrown.

Usage

solve_board(board)

Arguments

board A lightsout board object.

Details

There are a few algorithms for solving Lights Out puzzles. This function implements the Gaussian Elimination technique, which does not guarantee the minimum number of steps. Therefore, some steps in the given solution may be redundant.

If you are interested, there are many resources online outlining the exact details of how this technique works, and what the other solving strategies are.

Value

A matrix with the same dimensions as the input board, with a 1 in every position that requires a press to solve to the board. Note that the order of the light presses does not matter.

See Also

new_board random_board play is_solvable is_solved

Examples

# Create an empty 5x5 board, press two lights, and then see that the solution # tells us to press the same lights in order to solve the board.
board <- empty_board(5) %>% play(3, 2) %>% play(4, 1)
solution <- solve_board(board)
solution
board <- play(board, matrix = solution)
is_solved(board)
Index

board_entries, 2
empty_board, 3, 7
is_solvable, 3, 4, 9
is_solved, 4, 4, 9
launch, 5
lightsout, 5
lightsout-package (lightsout), 5
new_board, 3, 5, 7–9
play, 6, 6, 8, 9
random_board, 3, 5–7, 8, 9
solve_board, 4, 6–8, 9