# Package 'sudoku'

October 14, 2022

**Version** 2.8

**Date** 2022-04-19

**Title** Sudoku Puzzle Generator and Solver

**Author** David Brahm <brahm@alum.mit.edu> and Greg Snow

<Greg.Snow@intermountainmail.org>, with contributions from Curt Seeliger <Seeliger.Curt@epamail.epa.gov> and Henrik Bengtsson

<hb@maths.lth.se>.

**Maintainer** David Brahm <brahm@alum.mit.edu>

**Suggests** tkrplot

**Description** Generates, plays, and solves Sudoku puzzles. The GUI playSudoku() needs package ``tkrplot'' if you are not on Windows.

**License** GPL

**NeedsCompilation** no

**Repository** CRAN

**Date/Publication** 2022-04-19 14:30:02 UTC

## R topics documented:

fetchSudokuUK *Fetch the daily sudoku puzzle from https://www.sudoku.org.uk/*

## Description

Fetches the daily sudoku puzzle from https://www.sudoku.org.uk/ or one of their archive from the previous 31 days.

## Usage

```
fetchSudokuUK(day)
```

## Arguments

| | |
|---|---|
| day | Optional character string specifying the day of the puzzle to download. This is in European date format 'dd/mm/yyyy' and needs to represent a date within the last 31 days. |

## Value

A 9x9 matrix representing a sudoku puzzle (blank squares have value 0).

## Note

See the website for copyright information. Don't submit your solution for the prize contest if you used `solveSudoku` or `playSudoku` with `solve=TRUE`. This function requires a working internet connection.

## Author(s)

Greg Snow <greg.snow@intermountainmail.org>

## References

https://www.sudoku.org.uk/

## See Also

`solveSudoku`, `playSudoku`, `generateSudoku`

## Examples

```
## Not run:

#todays puzzle
puz <- fetchSudokuUK()

# puzzle from 25 Jan 2006 (if still available)
puza <- fetchSudokuUK('25/01/2006')
```

```
playSudoku(puza)

## End(Not run)
```

---

`generateSudoku` *Randomly Generate a Sudoku Puzzle Grid*

---

## Description

Creates a 9x9 Sudoku grid suitable for use by `playSudoku`.

## Usage

```
generateSudoku(Nblank=50, print.it=FALSE)
```

## Arguments

| | |
|---|---|
| `Nblank` | Number of cells to blank out |
| `print.it` | Logical. If true, print result to screen. |

## Details

The basic algorithm is to start with a 'primordial' Sudoku grid, swap around some rows and columns, then blank out some cells.

## Value

A matrix, representing a 9x9 Sudoku grid.

## Author(s)

Curt Seeliger <<Seeliger.Curt@epamail.epa.gov>>, Henrik Bengtsson <<hb@maths.lth.se>>, and David Brahm <<brahm@alum.mit.edu>>

## References

https://sudoku.com/

## Examples

```
generateSudoku(print.it=TRUE)
```

## hintSudoku *Give a Hint for a Sudoku Cell*

### Description

Generates a text string containing a 'hint' for cell (i,j) of Sudoku grid 'z'.

### Usage

```
hintSudoku(z, i, j)
```

### Arguments

| | |
|---|---|
| z | A 9x9 numeric matrix |
| i | Row index |
| j | Column index |

### Value

A character string, suitable for cat.

### Author(s)

Greg Snow <greg.snow@intermountainmail.org> and David E. Brahm <<brahm@alum.mit.edu>>

## playSudoku *Interactively play a game of Sudoku*

### Description

Interactively play a game of 9x9 Sudoku with hints and undo

### Usage

```
playSudoku(z=NULL, hist.len=100, solve=TRUE,
           display=c("guess","windows","tk"),
           hscale=1.25, vscale=1.25, ...)
```

## Arguments

| | |
|---|---|
| z | Either a 9x9 numeric matrix representing the Sudoku grid (with '0' representing a blank cell), or 0 (zero) for an empty matrix, or a filename (passed to readSudoku), or NULL to generate a puzzle randomly. |
| hist.len | Integer representing the number of history steps to remember (number of undo's possible). |
| solve | Logical indicating if the solution should be computed (used for checking current answer or cheating). |
| display | Type of display. The default 'guess' uses a windows graphics device if getOption('device')=='windows', otherwise it uses tk (requiring the 'tkrplot' package). |
| hscale | Passed to tkrplot |
| vscale | Passed to tkrplot |
| ... | Arguments passed to generateSudoku |

## Details

To play, move the mouse arrow over an empty cell and press the number key to enter the number in the cell. Typing '?' brings up a menu of additional commands:

```
?     -- a short help message
1-9   -- insert digit
0,' ' -- clear cell
r     -- replot the puzzle
q     -- quit
h     -- hint/help
c     -- correct wrong entries (show in red)
u     -- undo last entry
s     -- show number in cell
a     -- show all (solve the puzzle)
```

## Value

An invisible matrix with the solution or current state of the puzzle. Save this if you stop part way through, and use it as the input for the function to start again where you left off (undo info is lost so make sure that everything is correct).

## Note

display='windows' makes use of the getGraphicsEvent function, which currently only works on Windows.

## Author(s)

Greg Snow <greg.snow@intermountainmail.org> and David E. Brahm <<brahm@alum.mit.edu>>

## See Also

solveSudoku

## Examples

```
## Not run:
puz1 <- playSudoku(0)        # Use as an editor to create a puzzle, then quit
sol1 <- playSudoku(puz1)     # now play the puzzle

puz2 <- edit(matrix(0,9,9))  # Or use this editor
sol2 <- playSudoku(puz2)     # now play the puzzle

playSudoku()                 # Play a randomly generated game

playSudoku(fetchSudokuUK())  # Play today's game

## End(Not run)
```

---

`printSudoku` *Print a Sudoku Grid to the Terminal.*

---

## Description

Prints a Sudoku grid (a 9x9 matrix) to the terminal.

## Usage

```
printSudoku(z)
```

## Arguments

| | |
|---|---|
| z | A 9x9 numeric matrix, with '0' representing a blank cell. |

## Value

None; used for side effect.

## Author(s)

David E. Brahm <<brahm@alum.mit.edu>>

readSudoku *Read a File Containing a Sudoku Grid*

## Description

Reads a file containing a Sudoku grid (a 9x9 matrix).

## Usage

```
readSudoku(fn, map)
```

## Arguments

| | |
|---|---|
| fn | A filename. |
| map | Vector of unique puzzle elements (possibly longer than necessary). The default is c(1:9, letters), so an N=16 puzzle should be encoded using '1'-'9' and 'a'-'g'. |

## Details

The input file should look like this:

```
-6-1-4-5-
--83-56--
2-------1
8--4-7--6
--6---3--
7--9-1--4
5-------2
--72-69--
-4-5-8-7-
```

Blank cells can be indicated with any character not in "map", such as the '-' used here.

## Value

A numeric matrix (usually 9x9).

## Author(s)

David E. Brahm <<brahm@alum.mit.edu>>

## Examples

```
z <- readSudoku(system.file("puz1.txt", package="sudoku"))
```

`solveSudoku` *Solve a Sudoku Puzzle*

## Description

Solves a Sudoku Puzzle.

## Usage

```
solveSudoku(z, verbose=FALSE, map=c(1:9,letters), level=0,
            print.it=TRUE)
```

## Arguments

| | |
|---|---|
| `z` | A filename (passed to `readSudoku`), or a numeric matrix. |
| `verbose` | If TRUE, report on progress. |
| `map` | Vector of unique puzzle elements (possibly longer than necessary). The default is `c(1:9, letters)`, so an N=16 puzzle should be encoded using '1'-'9' and 'a'-'g'. |
| `level` | Recursion level (should not be set by user). |
| `print.it` | Logical: print the solution? |

## Details

A Sudoku puzzle consists of an NxN grid, where N is a perfect square (usually N=9). The grid is subdivided into N [sqrt(N) x sqrt(N)] boxes. You must fill in the missing values so that each row, each column, and each box contains the integers 1:N exactly once.

The algorithm uses an NxNxN array of logicals, representing the NxN cells and the N possible elements. For example, if a[1,2,3]=TRUE, then z[1,2] is known to be '3'. If a[1,2,4]=FALSE, then z[1,2] is known not to be '4'. The basic rules of Sudoku are used to fill in FALSE's, then elimination is used to find the TRUE's. If that approach runs out of steam, a guess is made and the program recurses to find either a solution or an inconsistency. No attempt is made to prove a solution's uniqueness.

## Value

Invisibly returns the solved (numerical) matrix, and prints the character version.

## Author(s)

David E. Brahm <<brahm@alum.mit.edu>>

## References

Example "puz1" comes from https://sudoku.com/.

## Examples

```
## Not run:
  solveSudoku(system.file("puz1.txt",package="sudoku"), verbose=TRUE)

## End(Not run)
```

---

`writeSudoku` *Write a Sudoku Grid to a File*

---

## Description

Writes a Sudoku grid (a matrix) to a file.

## Usage

```
writeSudoku(z, fn)
```

## Arguments

| | |
|---|---|
| `z` | A Sudoku grid. |
| `fn` | A filename. |

## Value

None; used for its side effect.

## Author(s)

David E. Brahm <<brahm@alum.mit.edu>>

## See Also

`readSudoku`

# Index