# Package 'mazing'

May 8, 2026

**Title** Utilities for Making and Plotting Mazes

**Version** 1.0.5

**Description** Functionality for generating and plotting random mazes. The mazes are based on matrices, so can only consist of vertical and horizontal lines along a regular grid. But there is no need to use every possible space, so they can take on many different shapes.

**License** MIT + file LICENSE

**Depends** R (>= 4.0)

**Imports** graphics, methods, stats

**Suggests** rmarkdown, emojifont, knitr, markdown, png, testthat (>= 3.0.0), covr

**VignetteBuilder** knitr

**RoxygenNote** 7.1.1

**BugReports** https://github.com/kstreet13/mazing/issues

**Config/testthat/edition** 3

**NeedsCompilation** no

**Author** Kelly Street [aut, cre] (ORCID: <https://orcid.org/0000-0001-6379-5013>)

**Maintainer** Kelly Street <street.kelly@gmail.com>

**Repository** CRAN

**Date/Publication** 2021-10-05 23:20:31 UTC

## Contents

---

`condense_matrix` *Halve the dimensions of a matrix*

---

## Description

A function to decrease the size of a matrix while attempting to preserve the original values.

## Usage

```
condense_matrix(m, fun = median)
```

## Arguments

| | |
|---|---|
| `m` | A `matrix`. |
| `fun` | A function for summarizing four values that condenses them down to one value. Default is `median`. |

## Details

This function can be used for image manipulation and is included for the purpose of making mazes from images. PNG images can be read in as arrays via `png::readPNG` and if the image is too big, this funtion can help get it down to a more manageable size.

## Value

A matrix with dimensions half the size of the original matrix, where each value is a summary of the four corresponding values in the original.

## Examples

```
m <- outer(1:4, 1:6)
condense_matrix(m)
```

---

`expand_matrix` *Double the dimensions of a matrix*

---

## Description

A function to double the size of a matrix, with each cell in the original becoming four cells in the expanded matrix.

## Usage

```
expand_matrix(m)
```

### Arguments

| | |
|---|---|
| m | A matrix. |

### Value

A matrix with the same distribution of values as the original, but doubled in size along both dimensions.

### Examples

```
m <- matrix(1:6, nrow = 2)
expand_matrix(m)
```

---

## find_maze_refpoint *Get coordinates for a point in a maze*

---

### Description

A function that takes a description of a point in a maze and finds the matrix indices corresponding to that point.

### Usage

```
find_maze_refpoint(point, maze)
```

### Arguments

| | |
|---|---|
| point | A description of a relative position in the maze, such as "topleft" or "bottomright". See Details for all possible values. |
| maze | A maze object. |

### Details

For standard values of point, this function will identify a "target" (such as the top left corner of the matrix) and select the point in the maze that is closest to that target, by Euclidean distance. The standard choices for point are: "topleft", "top", "topright", "righttop", "right", "rightbottom", "bottomright", "bottom", "bottomleft", "leftbottom", "left", "lefttop", and "center". For convenience, there are several redundancies built in; for example, "topleft" is identical to "lefttop".

In addition to the standard values, there is a complementary set of "matrix" or "manhattan-like" values, each of which is prepended with an "m" (for example, "mtopleft"). These options select the most extreme value along one dimension (ie. the highest possible row), fix that value, and then select the local extreme in the other dimension (column). For example, the value "mtopleft" will select the highest possible row in the maze before selecting the left-most point in that row. Note that this means values such as "mtopleft" and "mlefttop" are not synonymous.

For convenience, if `point` is a 2-column matrix or numeric vector that can be coerced into a 2-column matrix, it will be returned as such, with minor formatting changes to match the usual output. This is so that functions which rely on `find_maze_refpoint` can use either relative descriptions or exact coordinates.

## Value

A matrix of integers with 2 columns, giving the coordinates of the desired point(s). Note that the x-coordinate (column index) comes first, so for the corresponding index in the original matrix, these coordinates will need to be reversed.

## Examples

```
m <- maze(15,15)
r <- find_maze_refpoint('topright', m)

plot(m, walls = TRUE)
points(r[1], r[2], col = 2, pch = 16)
```

---

`maze` *Random Mazes*

---

## Description

Functions for producing and identifying procedurally generated random mazes.

## Usage

```
maze(nrow, ncol)

is.maze(x)

as.maze(x)
```

## Arguments

| | |
|---|---|
| `nrow` | Number of rows. |
| `ncol` | Number of columns. |
| `x` | A matrix-like object to be made into a maze (for `as.maze`), or an object to be identified as either a `maze` or not (for `is.maze`) |

## Details

For `as.maze`, if `x` is a binary matrix (or otherwise contains only two unique values), the maze will be constrained to occupy only those cells containing a 1 (the higher value).

## Value

For `maze` and `as.maze`, an object of class `maze`, which is a subclass of `matrix`.

For `is.maze` a logical value indicating if the input is a valid `maze`.

## Examples

```
maze(10,10)
mat <- matrix(NA, 10, 10)
is.maze(mat)
m <- as.maze(mat)
is.maze(mat)

# circular maze
mat <- matrix(1, 30, 30)
for(i in 1:nrow(mat)){
    for(j in 1:ncol(mat)){
        if((i-15.5)^2+(j-15.5)^2 > 220){
            mat[i,j] <- 0
        }
    }
}
m <- as.maze(mat)
```

---

# maze2binary *Convert maze to binary matrix*

---

## Description

A function to convert a maze object into a binary matrix. This can be useful for visualization (as when plotting the walls of the maze) and for constructing complex mazes, such as a maze-within-a-maze.

## Usage

```
maze2binary(m)
```

## Arguments

| | |
|---|---|
| `m` | A `maze` object. |

## Value

A binary matrix where values of 1 denote paths through the maze and values of 0 denote the walls (impassable regions) of the maze.

## Examples

```
m <- maze(10,10)
m2 <- maze2binary(m)
```

---

`plot.maze` *Plot a maze object*

---

## Description

Plot a maze object

## Usage

```
## S3 method for class 'maze'
plot(x, walls = FALSE, ...)

## S3 method for class 'maze'
lines(
  x,
  walls = FALSE,
  adjust = c(0, 0),
  openings = NULL,
  openings_direction = NULL,
  ...
)
```

## Arguments

| | |
|---|---|
| `x` | A `maze` object. |
| `walls` | logical value, indicating that the walls of the maze should be plotted, rather than the paths through the maze. Default is `FALSE`. |
| `...` | Additional arguments passed to `lines` |
| `adjust` | A vector by which to adjust the overall position of the maze. Should be a numeric vector of length 2, indicating the `x` and `y` adjustments, respectively. |
| `openings` | Locations in the maze where a certain wall(s) should not be drawn (only applies when `walls = TRUE`). May be specified as coordinates, or as a relative description (see `find_maze_refpoint`) |
| `openings_direction` | Character vector describing which walls should not be drawn at the locations specified by `openings`. See Details. |

## Details

When plotting, the coordinates for locations in the maze are taken from the indices of the corresponding locations in the matrix representation. This means that the plot will appear to be flipped vertically relative to the matrix representation of the maze.

The `openings_direction` argument specifies where to create an opening (ie. where not to draw walls), relative to a location in the maze given by `openings`. Possible values are: `"left"`, `"right"`, `"top"`, `"bottom"`, `"topleft"`, `"topright"`, `"bottomright"`, `"bottomleft"`, and `"all"`. If not specified, this will be set as the first value in `c("left","right","top","bottom")` to reference a wall that would otherwise have been drawn.

## Value

Returns `NULL`, invisibly.

## Examples

```
m <- maze(10,10)
plot(m, walls = TRUE)
lines(m, lwd = 5, col = 3)
```

---

# solve_maze *Find a path through a maze*

---

## Description

A function that finds the shortest path between points in a maze.

## Usage

```
solve_maze(maze, start = "bottomleft", end = "topright")
```

## Arguments

| | |
|---|---|
| `maze` | A `maze` object. |
| `start` | The coordinates of the starting point, or a description of a relative location (see `find_maze_refpoint`). If not provided, this will be as close as possible to the bottom left corner. |
| `end` | The coordinates of the end point, or a description of a relative location (see `find_maze_refpoint`). If not provided, this will be as close as possible to the top right corner. |

## Details

For the `start` and `end` arguments (as well as the output matrix), these coordinates refer to the plotting coordinates, not the matrix indices. For plotting, the x-coordinate (column index) is listed first, whereas in matrix notation, the row (y-coordinate) is listed first.

## Value

A matrix containing the coordinates of the path through the maze. Note that the x-coordinate (column index) comes first, so for the corresponding indices in the original matrix, these coordinates will need to be reversed.

## Examples

```
m <- maze(15,15)
p <- solve_maze(m)

plot(m, walls = TRUE)
lines(p, col = 2, lwd = 3)
```

---

# widen_paths *Spread a value within a matrix*

---

## Description

A function to spread a particular value into neighboring cells of a matrix. When that matrix is a binary representation of a maze, this can be used to widen the paths of the maze (subsequently narrowing the walls).

## Usage

```
widen_paths(m, value = 1, blocked = NULL, square_corners = FALSE)
```

## Arguments

| | |
|---|---|
| m | A matrix. |
| value | The value to be expanded into neighboring cells. For widening the paths of the maze, this is the value that represents paths. |
| blocked | Values that should be preserved, regardless of whether or not the expansion would normally encroach on their cells. |
| square_corners | Whether or not to include diagonals in the expansion. Including diagonals will preserve square corners, whereas excluding them (default) creates more rounded looking corners. |

## Details

The idea of this function is to let the paths of a maze "seep" into the walls. For making a maze-within-a-maze, the walls of the big maze are generally not as interesting as the paths, which contain smaller paths. Hence why we might want to make the paths wider than the walls. Remember that this will affect walls from both sides, so it is necessary to expand the matrix twice (via expand_matrix) before widening the paths for the first time.

## Value

A matrix with the same dimensions as the original, in which cells that border a cell of a particular `value` have been assigned that `value`.

## Examples

```
m <- matrix(0, nrow = 5, ncol = 5)
m[3,3] <- 1
widen_paths(m)
widen_paths(m, square_corners = TRUE)

m[,2] <- 2
widen_paths(m, blocked = 2)
```

# Index