Package ‘mazeGen’

October 13, 2022

Type Package

Title Elithorn Maze Generator

Version 0.1.3

Date 2017-12-04

Author Bao Sheng Loe (Aiden) [aut,cre,cph],
Maria Sanchez[ctb]

Maintainer Bao Sheng Loe (Aiden) <bsl28@cam.ac.uk>

Description A maze generator that creates the Elithorn Maze (HTML file) and the functions to calculate the associated maze parameters (i.e. Difficulty and Ability).

License GPL-3

Imports igraph, grDevices

LazyData TRUE

RoxygenNote 6.0.1

NeedsCompilation no

Repository CRAN

Date/Publication 2017-12-04 17:34:53 UTC

R topics documented:

genEMLseed 3
genMaze 4
genPathSeed 5
gridEightDown 6
gridEighteenDown 7
gridEighteenLeft 7
gridEighteenRight 8
gridEighteenUp 9
gridEightLeft 9
gridEightRight 10
gridEightUp 11
gridElevenDown 11

gridElevenLeft 12
gridElevenRight 13
gridElevenUp 13
gridFifteenDown 14
gridFifteenLeft 15
gridFifteenRight 15
gridFifteenUp 16
gridFiveDown 17
gridFiveLeft 17
gridFiveRight 18
gridFiveUp 19
gridFourDown 19
gridFourLeft 20
gridFourRight 21
gridFourteenDown 21
gridFourteenLeft 22
gridFourteenRight 23
gridFourteenUp 23
gridFourUp 24
gridNineDown 25
gridNineLeft 25
gridNineRight 26
gridNineteenUp 27
gridNineUp 28
gridSevenDown 29
gridSevenLeft 29
gridSevenRight 30
gridSeventeenDown 31
gridSeventeenLeft 31
gridSeventeenRight 32
gridSeventeenUp 33
gridSevenUp 33
gridSixDown 34
gridSixLeft 35
gridSixRight 35
gridSixteenDown 36
gridSixteenLeft 37
gridSixteenRight 37
gridSixteenUp 38
gridSixUp 39
gridTenDown 39
gridTenLeft 40
gridTenRight 41
gridTenUp 41
gridThirteenDown 42
gridThirteenLeft 43
gridThirteenRight 43
gridThirteenUp 44

gridThreeDown 45
gridThreeLeft 45
gridThreeRight 46
gridThreeUp 47
gridTwelveDown 47
gridTwelveLeft 48
gridTwelveRight 49
gridTwelveUp 49
gridTwentyDown 50
gridTwentyLeft 51
gridTwentyRight 51
gridTwentyUp 52
howMany 53
lowerGrid 53
maxScore 54
mazeAbility 55
mazeDiff 56
mazeEst 57
mazeGen 59
mazeHTML 60
mazeObject 61
np 62
topNodes 63

Index **65**

genEMLseed *Generate Equal Minimum Legs Seed*

Description

This generate the solution by searching for the SEED that returns the specific number of paths to achieve the maximum score for a given rank and saturation.

Usage

```
genEMLseed(path = 3, rank = 5, satPercent = 0.5, seed = 1,
  runSeed = 500)
```

Arguments

`path`	Selecting the specific number of paths to achieve the maximum score.
`rank`	This is the rank of the maze.
`satPercent`	This is of saturation percentage ranging from 0-1.
`seed`	The starting seed to begin searching for the seed with specific paths.
`runSeed`	This determines the number of searches for the specific paths before stopping.

Details

This might be computationally intensive as the maze size increases. The seed is necessary so that the algorithm does not always begin from the smallest seed value. Based on the starting seed value, it will search for the next seed that returns the desired number of path defined by the user. To limit the search time, The function will stop looking for the seed based on the runSeed value. Using this function will guarantee that the minimum number of steps to achieve the maximum score will be the same for all possible paths. If the number of steps does not need to be equal across all possible paths for the maximum score, please use the genPathSeed function instead.

Author(s)

Aiden Loe and Maria Sanchez

See Also

np,mazeEst, genPathSeed

Examples

```
rank <- 5
satPercent <- 0.5
seed <- 1

#Search for just one unique path
justOne <- genEMLseed(path=1,rank=rank,satPercent=satPercent,seed=seed)
nodePosition <- np(rank,satPercent,seed=justOne)
mazeEst(nodePosition)

#Search for three path
justThree <- genEMLseed(path=3,rank=rank,satPercent=satPercent,seed=seed, runSeed=300)
nodePosition <- np(rank,satPercent,seed=justThree)
mazeEst(nodePosition)
```

genMaze *genMaze*

Description

This function generates the list of edges.

Usage

```
genMaze(rank = 5)
```

Arguments

rank	This is the Rank of the maze.

Details

The Genmaze function generates the list of edges. The edges will be used to construct the maze.

Author(s)

Aiden Loe

Examples

```
genMaze(rank=5)
```

`genPathSeed` *Generate Path Seed*

Description

This generate the solution by searching for the SEED that returns the specific number of paths to achieve the maximum score for a given rank and saturation.

Usage

```
genPathSeed(path = 3, rank = 5, satPercent = 0.5, seed = 1,
  runSeed = 500)
```

Arguments

`path`	Selecting the specific number of paths to achieve the maximum score.
`rank`	This is the rank of the maze.
`satPercent`	This is of saturation percentage ranging from 0-1.
`seed`	The starting seed to begin searching for the seed with specific paths.
`runSeed`	This determines the number of searches for the specific paths before stopping.

Details

This might be computationally intensive as the maze size increases. The seed is necessary so that the algorithm does not always begin from the smallest seed value. Based on the starting seed value, it will search for the next seed that returns the desired number of path defined by the user. To limit the search time, The function will stop looking for the seed based on the runSeed value. Using this function does not guarantee that the minimum number of steps will be the same for all possible paths to achieve the maximum score. To ensure that the number of steps are equal across all possible paths for the maximum score, please use the `genEMLseed` function instead.

Author(s)

Aiden Loe and Maria Sanchez

See Also

np,mazeEst, genEMLseed

Examples

```
rank <- 5
satPercent <- 0.5
seed <- 1

#Search for just one unique path
justOne <- genPathSeed(path=1,rank=rank,satPercent=satPercent,seed=seed)
nodePosition <- np(rank,satPercent,seed=justOne)
mazeEst(nodePosition)

#Search for three path
justThree <- genPathSeed(path=3,rank=rank,satPercent=satPercent,seed=seed, runSeed=300)
nodePosition <- np(rank,satPercent,seed=justThree)
mazeEst(nodePosition)
```

gridEightDown *Grid Eight Down*

Description

This returns a eight grid downwards maze. These are standardized coordinates.

Usage

```
data(gridEightDown)
```

Format

A data frame with 2 columns

start Coordinates of Start Node.

end Coordinates End Node.

Examples

```
## Not run:

# Returns a Grid with rank = 8
data(gridEightDown)
coordinates <- gridEightDown


## End(Not run)
```

`gridEighteenDown` *Grid Eighteen Down*

Description

This returns a eighteen grid downards maze. These are standardized coordinates.

Usage

```
data(gridEighteenDown)
```

Format

A data frame with 2 columns

start Coordinates of Start Node.

end Coordinates End Node.

Examples

```
## Not run:

# Returns a Grid with rank = 18
data(gridEighteenDown)
coordinates <- gridEighteenDown


## End(Not run)
```

`gridEighteenLeft` *Grid Eighteen Left*

Description

This returns a eighteen grid left maze. These are standardized coordinates.

Usage

```
data(gridEighteenLeft)
```

Format

A data frame with 2 columns

start Coordinates of Start Node.

end Coordinates End Node.

Examples

```
## Not run:

# Returns a Grid with rank = 18
data(gridEighteenLeft)
coordinates <- gridEighteenLeft


## End(Not run)
```

`gridEighteenRight` *Grid Eighteen Right*

Description

This returns a eighteen grid right maze. These are standardized coordinates.

Usage

```
data(gridEighteenRight)
```

Format

A data frame with 2 columns

start Coordinates of Start Node.

end Coordinates End Node.

Examples

```
## Not run:

# Returns a Grid with rank = 18
data(gridEighteenRight)
coordinates <- gridEighteenRight


## End(Not run)
```

gridEighteenUp *Grid Eighteen Up*

Description

This returns a eighteen grid upwards maze. These are standardized coordinates.

Usage

```
data(gridEighteenUp)
```

Format

A data frame with 2 columns

start Coordinates of Start Node.

end Coordinates End Node.

Examples

```
## Not run:

# Returns a Grid with rank = 18
data(gridEightteenUp)
coordinates <- gridEighteenUp


## End(Not run)
```

gridEightLeft *Grid Eight Left*

Description

This returns a eight grid left maze. These are standardized coordinates.

Usage

```
data(gridEightLeft)
```

Format

A data frame with 2 columns

start Coordinates of Start Node.

end Coordinates End Node.

Examples

```
## Not run:

# Returns a Grid with rank = 8
data(gridEightLeft)
coordinates <- gridEightLeft


## End(Not run)
```

`gridEightRight` *Grid Eight Right*

Description

This returns a eight grid right maze. These are standardized coordinates.

Usage

```
data(gridEightRight)
```

Format

A data frame with 2 columns

start Coordinates of Start Node.

end Coordinates End Node.

Examples

```
## Not run:

# Returns a Grid with rank = 8
data(gridEightRight)
coordinates <- gridEightRight


## End(Not run)
```

gridEightUp *Grid Eight Up*

Description

This returns a eight grid upwards maze. These are standardized coordinates.

Usage

```
data(gridEightUp)
```

Format

A data frame with 2 columns

start Coordinates of Start Node.

end Coordinates End Node.

Examples

```
## Not run:

# Returns a Grid with rank = 8
data(gridEightUp)
coordinates <- gridEightUp


## End(Not run)
```

gridElevenDown *Grid Eleven Down*

Description

This returns a eleven grid downards maze. These are standardized coordinates.

Usage

```
data(gridElevenDown)
```

Format

A data frame with 2 columns

start Coordinates of Start Node.

end Coordinates End Node.

Examples

```
## Not run:

# Returns a Grid with rank = 11
data(gridElevenDown)
coordinates <- gridElevenDown


## End(Not run)
```

`gridElevenLeft` *Grid Eleven Left*

Description

This returns a eleven grid left maze. These are standardized coordinates.

Usage

```
data(gridElevenLeft)
```

Format

A data frame with 2 columns

start Coordinates of Start Node.

end Coordinates End Node.

Examples

```
## Not run:

# Returns a Grid with rank = 11
data(gridElevenLeft)
coordinates <- gridElevenLeft


## End(Not run)
```

gridElevenRight *Grid Eleven Right*

Description

This returns a eleven grid right maze. These are standardized coordinates.

Usage

```
data(gridElevenRight)
```

Format

A data frame with 2 columns

start Coordinates of Start Node.

end Coordinates End Node.

Examples

```
## Not run:

# Returns a Grid with rank = 11
data(gridElevenRight)
coordinates <- gridElevenRight


## End(Not run)
```

gridElevenUp *Grid Eleven Up*

Description

This returns a eleven grid upwards maze. These are standardized coordinates.

Usage

```
data(gridElevenUp)
```

Format

A data frame with 2 columns

start Coordinates of Start Node.

end Coordinates End Node.

Examples

```
## Not run:

# Returns a Grid with rank = 11
data(gridElevenUp)
coordinates <- gridElevenUp


## End(Not run)
```

---

gridFifteenDown *Grid Fifteen Down*

Description

This returns a fifteen grid downards maze. These are standardized coordinates.

Usage

```
data(gridFifteenDown)
```

Format

A data frame with 2 columns

start Coordinates of Start Node.

end Coordinates End Node.

Examples

```
## Not run:

# Returns a Grid with rank = 15
data(gridFifteenDown)
coordinates <- gridFifteenDown


## End(Not run)
```

`gridFifteenLeft` *Grid Fifteen Left*

Description

This returns a fifteen grid left maze. These are standardized coordinates.

Usage

```
data(gridFifteenLeft)
```

Format

A data frame with 2 columns

start Coordinates of Start Node.

end Coordinates End Node.

Examples

```
## Not run:

# Returns a Grid with rank = 15
data(gridFifteenLeft)
coordinates <- gridFifteenLeft


## End(Not run)
```

`gridFifteenRight` *Grid Fifteen Right*

Description

This returns a fifteen grid right maze. These are standardized coordinates.

Usage

```
data(gridFifteenRight)
```

Format

A data frame with 2 columns

start Coordinates of Start Node.

end Coordinates End Node.

Examples

```
## Not run:

# Returns a Grid with rank = 15
data(gridFifteenRight)
coordinates <- gridFifteenRight


## End(Not run)
```

`gridFifteenUp` *Grid Fifteen Up*

Description

This returns a fifteen grid upwards maze. These are standardized coordinates.

Usage

```
data(gridFifteenUp)
```

Format

A data frame with 2 columns

start Coordinates of Start Node.

end Coordinates End Node.

Examples

```
## Not run:

# Returns a Grid with rank = 15
data(gridFifteenUp)
coordinates <- gridFifteenUp


## End(Not run)
```

`gridFiveDown` *Grid Five Down*

Description

This returns a five grid downwards maze. These are standardized coordinates.

Usage

```
data(gridFiveDown)
```

Format

A data frame with 2 columns

start Coordinates of Start Node.

end Coordinates End Node.

Examples

```
## Not run:

# Returns a Grid with rank = 5
data(gridFiveDown)
coordinates <- gridFiveDown


## End(Not run)
```

`gridFiveLeft` *Grid Five Left*

Description

This returns a five grid left maze. These are standardized coordinates.

Usage

```
data(gridFiveLeft)
```

Format

A data frame with 2 columns

start Coordinates of Start Node.

end Coordinates End Node.

Examples

```
## Not run:

# Returns a Grid with rank = 5
data(gridFiveLeft)
coordinates <- gridFiveLeft


## End(Not run)
```

`gridFiveRight` *Grid Five Right*

Description

This returns a five grid right maze. These are standardized coordinates.

Usage

```
data(gridFiveRight)
```

Format

A data frame with 2 columns

start Coordinates of Start Node.

end Coordinates End Node.

Examples

```
## Not run:

# Returns a Grid with rank = 5
data(gridFiveRight)
coordinates <- gridFiveRight


## End(Not run)
```

gridFiveUp *Grid Five Up*

Description

This returns a five grid upwards maze. These are standardized coordinates.

Usage

```
data(gridFiveUp)
```

Format

A data frame with 2 columns

start Coordinates of Start Node.

end Coordinates End Node.

Examples

```
## Not run:

# Returns a Grid with rank = 5
data(gridFiveUp)
coordinates <- gridFiveUp


## End(Not run)
```

gridFourDown *Grid Four Down*

Description

This returns a four grid downwards maze. These are standardized coordinates.

Usage

```
data(gridFourDown)
```

Format

A data frame with 2 columns

start Coordinates of Start Node.

end Coordinates End Node.

Examples

```
## Not run:

# Returns a Grid with rank = 4
data(gridFourDown)
coordinates <- gridFourDown


## End(Not run)
```

`gridFourLeft` *Grid Four Left*

Description

This returns a four grid left maze. These are standardized coordinates.

Usage

```
data(gridFourLeft)
```

Format

A data frame with 2 columns

start Coordinates of Start Node.

end Coordinates End Node.

Examples

```
## Not run:

# Returns a Grid with rank = 4
data(gridFourLeft)
coordinates <- gridFourLeft


## End(Not run)
```

gridFourRight *Grid Four Right*

Description

This returns a four grid right maze. These are standardized coordinates.

Usage

```
data(gridFourRight)
```

Format

A data frame with 2 columns

start Coordinates of Start Node.

end Coordinates End Node.

Examples

```
## Not run:

# Returns a Grid with rank = 4
data(gridFourRight)
coordinates <- gridFourRight


## End(Not run)
```

gridFourteenDown *Grid Fourteen Down*

Description

This returns a fourteen grid downards maze. These are standardized coordinates.

Usage

```
data(gridFourteenDown)
```

Format

A data frame with 2 columns

start Coordinates of Start Node.

end Coordinates End Node.

Examples

```
## Not run:

# Returns a Grid with rank = 14
data(gridFourteenDown)
coordinates <- gridFourteenDown


## End(Not run)
```

`gridFourteenLeft` *Grid Fourteen Left*

Description

This returns a fourteen grid left maze. These are standardized coordinates.

Usage

```
data(gridFourteenLeft)
```

Format

A data frame with 2 columns

start Coordinates of Start Node.

end Coordinates End Node.

Examples

```
## Not run:

# Returns a Grid with rank = 14
data(gridFourteenLeft)
coordinates <- gridFourteenLeft


## End(Not run)
```

gridFourteenRight *Grid Fourteen Right*

Description

This returns a fourteen grid right maze. These are standardized coordinates.

Usage

```
data(gridFourteenRight)
```

Format

A data frame with 2 columns

start Coordinates of Start Node.

end Coordinates End Node.

Examples

```
## Not run:

# Returns a Grid with rank = 14
data(gridFourteenRight)
coordinates <- gridFourteenRight


## End(Not run)
```

gridFourteenUp *Grid Fourteen Up*

Description

This returns a fourteen grid upwards maze. These are standardized coordinates.

Usage

```
data(gridFourteenUp)
```

Format

A data frame with 2 columns

start Coordinates of Start Node.

end Coordinates End Node.

Examples

```
## Not run:

# Returns a Grid with rank = 14
data(gridFourteenUp)
coordinates <- gridFourteenUp


## End(Not run)
```

`gridFourUp` *Grid Four Up*

Description

This returns a four grid upwards maze. These are standardized coordinates.

Usage

```
data(gridFourUp)
```

Format

A data frame with 2 columns

start Coordinates of Start Node.

end Coordinates End Node.

Examples

```
## Not run:

# Returns a Grid with rank = 4
data(gridFourUp)
coordinates <- gridFourUp


## End(Not run)
```

`gridNineDown` *Grid Nine Down*

Description

This returns a nine grid downwards maze. These are standardized coordinates.

Usage

```
data(gridNineDown)
```

Format

A data frame with 2 columns

start Coordinates of Start Node.

end Coordinates End Node.

Examples

```
## Not run:

# Returns a Grid with rank = 9
data(gridNineDown)
coordinates <- gridNineDown


## End(Not run)
```

`gridNineLeft` *Grid Nine Left*

Description

This returns a nine grid left maze. These are standardized coordinates.

Usage

```
data(gridNineLeft)
```

Format

A data frame with 2 columns

start Coordinates of Start Node.

end Coordinates End Node.

Examples

```
## Not run:

# Returns a Grid with rank = 9
data(gridNineLeft)
coordinates <- gridNineLeft


## End(Not run)
```

`gridNineRight` *Grid Nine Right*

Description

This returns a nine grid right maze. These are standardized coordinates.

Usage

```
data(gridNineRight)
```

Format

A data frame with 2 columns

start Coordinates of Start Node.

end Coordinates End Node.

Examples

```
## Not run:

# Returns a Grid with rank = 9
data(gridNineRight)
coordinates <- gridNineRight


## End(Not run)
```

gridNineteenUp *Grid Nineteen Up*

Description

This returns a nineteen grid right maze. These are standardized coordinates.

This returns a nineteen grid right maze. These are standardized coordinates.

This returns a nineteen grid right maze. These are standardized coordinates.

This returns a nineteen grid right maze. These are standardized coordinates.

Usage

```
data(gridNineteenUp)

data(gridNineteenDown)

data(gridNineteenLeft)

data(gridNineteenRight)
```

Format

A data frame with 2 columns

start Coordinates of Start Node.

end Coordinates End Node.

Examples

```
## Not run:

# Returns a Grid with rank = 19
data(gridNineteenUp)
coordinates <- gridNineteenUp


## End(Not run)

## Not run:

# Returns a Grid with rank = 19
data(gridNineteenDown)
coordinates <- gridNineteenDown


## End(Not run)

## Not run:
```

```
# Returns a Grid with rank = 19
data(gridNineteenLeft)
coordinates <- gridNineteenLeft


## End(Not run)

## Not run:

# Returns a Grid with rank = 19
data(gridNineteenRight)
coordinates <- gridNineteenRight


## End(Not run)
```

gridNineUp *Grid Nine Up*

Description

This returns a nine grid upwards maze. These are standardized coordinates.

Usage

```
data(gridNineUp)
```

Format

A data frame with 2 columns

start Coordinates of Start Node.

end Coordinates End Node.

Examples

```
## Not run:

# Returns a Grid with rank = 9
data(gridNineUp)
coordinates <- gridNineUp


## End(Not run)
```

gridSevenDown *Grid Seven Down*

Description

This returns a seven grid downwards maze. These are standardized coordinates.

Usage

```
data(gridSevenDown)
```

Format

A data frame with 2 columns

start Coordinates of Start Node.

end Coordinates End Node.

Examples

```
## Not run:

# Returns a Grid with rank = 7
data(gridSevenDown)
coordinates <- gridSevenDown


## End(Not run)
```

gridSevenLeft *Grid Seven Left*

Description

This returns a seven grid left maze. These are standardized coordinates.

Usage

```
data(gridSevenLeft)
```

Format

A data frame with 2 columns

start Coordinates of Start Node.

end Coordinates End Node.

Examples

```
## Not run:

# Returns a Grid with rank = 7
data(gridSevenLeft)
coordinates <- gridSevenLeft


## End(Not run)
```

`gridSevenRight` *Grid Seven Right*

Description

This returns a seven grid right maze. These are standardized coordinates.

Usage

```
data(gridSevenRight)
```

Format

A data frame with 2 columns

start Coordinates of Start Node.

end Coordinates End Node.

Examples

```
## Not run:

# Returns a Grid with rank = 7
data(gridSevenRight)
coordinates <- gridSevenRight


## End(Not run)
```

`gridSeventeenDown` *Grid Seventeen Down*

Description

This returns a seventeen grid downards maze. These are standardized coordinates.

Usage

```
data(gridSeventeenDown)
```

Format

A data frame with 2 columns

start Coordinates of Start Node.

end Coordinates End Node.

Examples

```
## Not run:

# Returns a Grid with rank = 17
data(gridSeventeenDown)
coordinates <- gridSeventeenDown


## End(Not run)
```

`gridSeventeenLeft` *Grid Seventeen Left*

Description

This returns a seventeen grid left maze. These are standardized coordinates.

Usage

```
data(gridSeventeenLeft)
```

Format

A data frame with 2 columns

start Coordinates of Start Node.

end Coordinates End Node.

Examples

```
## Not run:

# Returns a Grid with rank = 17
data(gridSeventeenLeft)
coordinates <- gridSeventeenLeft


## End(Not run)
```

`gridSeventeenRight` *Grid Seventeen Right*

Description

This returns a seventeen grid right maze. These are standardized coordinates.

Usage

```
data(gridSeventeenRight)
```

Format

A data frame with 2 columns

start Coordinates of Start Node.

end Coordinates End Node.

Examples

```
## Not run:

# Returns a Grid with rank = 17
data(gridSeventeenRight)
coordinates <- gridSeventeenRight


## End(Not run)
```

gridSeventeenUp *Grid Seventeen Up*

Description

This returns a seventeen grid upwards maze. These are standardized coordinates.

Usage

```
data(gridSeventeenUp)
```

Format

A data frame with 2 columns

start Coordinates of Start Node.

end Coordinates End Node.

Examples

```
## Not run:

# Returns a Grid with rank = 17
data(gridSeventeenUp)
coordinates <- gridSeventeenUp


## End(Not run)
```

gridSevenUp *Grid Seven Up*

Description

This returns a seven grid upwards maze. These are standardized coordinates.

Usage

```
data(gridSevenUp)
```

Format

A data frame with 2 columns

start Coordinates of Start Node.

end Coordinates End Node.

Examples

```
## Not run:

# Returns a Grid with rank = 7
data(gridSevenUp)
coordinates <- gridSevenUp


## End(Not run)
```

`gridSixDown` *Grid Six Down*

Description

This returns a six grid downwards maze. These are standardized coordinates.

Usage

```
data(gridSixDown)
```

Format

A data frame with 2 columns

start Coordinates of Start Node.

end Coordinates End Node.

Examples

```
## Not run:

# Returns a Grid with rank = 6
data(gridSixDown)
coordinates <- gridSixDown


## End(Not run)
```

gridSixLeft *Grid Six Left*

Description

This returns a six grid left maze. These are standardized coordinates.

Usage

```
data(gridSixLeft)
```

Format

A data frame with 2 columns

start Coordinates of Start Node.

end Coordinates End Node.

Examples

```
## Not run:

# Returns a Grid with rank = 6
data(gridSixLeft)
coordinates <- gridSixLeft


## End(Not run)
```

gridSixRight *Grid Six Right*

Description

This returns a six grid right maze. These are standardized coordinates.

Usage

```
data(gridSixRight)
```

Format

A data frame with 2 columns

start Coordinates of Start Node.

end Coordinates End Node.

Examples

```
## Not run:

# Returns a Grid with rank = 6
data(gridSixRight)
coordinates <- gridSixRight


## End(Not run)
```

gridSixteenDown *Grid Sixteen Down*

Description

This returns a sixteen grid downards maze. These are standardized coordinates.

Usage

```
data(gridSixteenDown)
```

Format

A data frame with 2 columns

start Coordinates of Start Node.

end Coordinates End Node.

Examples

```
## Not run:

# Returns a Grid with rank = 16
data(gridSixteenDown)
coordinates <- gridSixteenDown


## End(Not run)
```

gridSixteenLeft *Grid Sixteen Left*

Description

This returns a sixteen grid left maze. These are standardized coordinates.

Usage

```
data(gridSixteenLeft)
```

Format

A data frame with 2 columns

start Coordinates of Start Node.

end Coordinates End Node.

Examples

```
## Not run:

# Returns a Grid with rank = 16
data(gridSixteenLeft)
coordinates <- gridSixteenLeft


## End(Not run)
```

gridSixteenRight *Grid Sixteen Right*

Description

This returns a sixteen grid right maze. These are standardized coordinates.

Usage

```
data(gridSixteenRight)
```

Format

A data frame with 2 columns

start Coordinates of Start Node.

end Coordinates End Node.

Examples

```
## Not run:

# Returns a Grid with rank = 16
data(gridSixteenRight)
coordinates <- gridSixteenRight


## End(Not run)
```

`gridSixteenUp` *Grid Sixteen Up*

Description

This returns a sixteen grid upwards maze. These are standardized coordinates.

Usage

```
data(gridSixteenUp)
```

Format

A data frame with 2 columns

start Coordinates of Start Node.

end Coordinates End Node.

Examples

```
## Not run:

# Returns a Grid with rank = 16
data(gridSixteenUp)
coordinates <- gridSixteenUp


## End(Not run)
```

gridSixUp *Grid Six Up*

Description

This returns a six grid upwards maze. These are standardized coordinates.

Usage

```
data(gridSixUp)
```

Format

A data frame with 2 columns

start Coordinates of Start Node.

end Coordinates End Node.

Examples

```
## Not run:

# Returns a Grid with rank = 6
data(gridSixUp)
coordinates <- gridSixUp


## End(Not run)
```

gridTenDown *Grid Ten Down*

Description

This returns a ten grid downards maze. These are standardized coordinates.

Usage

```
data(gridTenDown)
```

Format

A data frame with 2 columns

start Coordinates of Start Node.

end Coordinates End Node.

Examples

```
## Not run:

# Returns a Grid with rank = 10
data(gridTenDown)
coordinates <- gridTenDown


## End(Not run)
```

`gridTenLeft` *Grid Ten Left*

Description

This returns a ten grid left maze. These are standardized coordinates.

Usage

```
data(gridTenLeft)
```

Format

A data frame with 2 columns

start Coordinates of Start Node.

end Coordinates End Node.

Examples

```
## Not run:

# Returns a Grid with rank = 10
data(gridTenLeft)
coordinates <- gridTenLeft


## End(Not run)
```

gridTenRight *Grid Ten Right*

Description

This returns a ten grid right maze. These are standardized coordinates.

Usage

```
data(gridTenRight)
```

Format

A data frame with 2 columns

start Coordinates of Start Node.

end Coordinates End Node.

Examples

```
## Not run:

# Returns a Grid with rank = 10
data(gridTenRight)
coordinates <- gridTenRight


## End(Not run)
```

gridTenUp *Grid Ten Up*

Description

This returns a ten grid upwards maze. These are standardized coordinates.

Usage

```
data(gridTenUp)
```

Format

A data frame with 2 columns

start Coordinates of Start Node.

end Coordinates End Node.

Examples

```
## Not run:

# Returns a Grid with rank = 10
data(gridTenUp)
coordinates <- gridTenUp


## End(Not run)
```

`gridThirteenDown` *Grid Thirteen Down*

Description

This returns a thirteen grid downards maze. These are standardized coordinates.

Usage

```
data(gridThirteenDown)
```

Format

A data frame with 2 columns

start Coordinates of Start Node.

end Coordinates End Node.

Examples

```
## Not run:

# Returns a Grid with rank = 13
data(gridThirteenDown)
coordinates <- gridThirteenDown


## End(Not run)
```

`gridThirteenLeft` *Grid Thirteen Left*

Description

This returns a thirteen grid left maze. These are standardized coordinates.

Usage

```
data(gridThirteenLeft)
```

Format

A data frame with 2 columns

start Coordinates of Start Node.

end Coordinates End Node.

Examples

```
## Not run:

# Returns a Grid with rank = 13
data(gridThirteenLeft)
coordinates <- gridThirteenLeft


## End(Not run)
```

`gridThirteenRight` *Grid Thirteen Right*

Description

This returns a thirteen grid right maze. These are standardized coordinates.

Usage

```
data(gridThirteenRight)
```

Format

A data frame with 2 columns

start Coordinates of Start Node.

end Coordinates End Node.

Examples

```
## Not run:

# Returns a Grid with rank = 13
data(gridThirteenRight)
coordinates <- gridThirteenRight


## End(Not run)
```

gridThirteenUp *Grid Thirteen Up*

Description

This returns a thirteen grid upwards maze. These are standardized coordinates.

Usage

```
data(gridThirteenUp)
```

Format

A data frame with 2 columns

start Coordinates of Start Node.

end Coordinates End Node.

Examples

```
## Not run:

# Returns a Grid with rank = 13
data(gridThirteenUp)
coordinates <- gridThirteenUp


## End(Not run)
```

`gridThreeDown` *Grid Three Down*

Description

This returns a three grid downwards maze. These are standardized coordinates.

Usage

```
data(gridThreeDown)
```

Format

A data frame with 2 columns

start Coordinates of Start Node.

end Coordinates End Node.

Examples

```
## Not run:

# Returns a Grid with rank = X
data(gridThreeDown)
coordinates <- gridThreeDown


## End(Not run)
```

`gridThreeLeft` *Grid Three Left*

Description

This returns a three grid left maze. These are standardized coordinates.

Usage

```
data(gridThreeLeft)
```

Format

A data frame with 2 columns

start Coordinates of Start Node.

end Coordinates End Node.

Examples

```
## Not run:

# Returns a Grid with rank = 3
data(gridThreeLeft)
coordinates <- gridThreeLeft


## End(Not run)
```

`gridThreeRight` *Grid Three Right*

Description

This returns a three grid left maze. These are standardized coordinates.

Usage

```
data(gridThreeRight)
```

Format

A data frame with 2 columns

start Coordinates of Start Node.

end Coordinates End Node.

Examples

```
## Not run:

# Returns a Grid with rank = 3
data(gridThreeRight)
coordinates <- gridThreeRight


## End(Not run)
```

gridThreeUp *Grid Three Up*

Description

This returns a three grid upwards maze. These are standardized coordinates.

Usage

```
data(gridThreeUp)
```

Format

A data frame with 2 columns

start start, coordinates of Start Node.

carat end, coordinates End Node.

Examples

```
## Not run:

# Returns a Grid with rank = 3
data(gridThreeUp)
coordinates <- gridThreeUp


## End(Not run)
```

gridTwelveDown *Grid Twelve Down*

Description

This returns a twelve grid downards maze. These are standardized coordinates.

Usage

```
data(gridTwelveDown)
```

Format

A data frame with 2 columns

start Coordinates of Start Node.

end Coordinates End Node.

Examples

```
## Not run:

# Returns a Grid with rank = 12
data(gridTwelveDown)
coordinates <- gridTwelveDown


## End(Not run)
```

gridTwelveLeft *Grid Twelve Left*

Description

This returns a twelve grid left maze. These are standardized coordinates.

Usage

```
data(gridTwelveLeft)
```

Format

A data frame with 2 columns

start Coordinates of Start Node.

end Coordinates End Node.

Examples

```
## Not run:

# Returns a Grid with rank = 12
data(gridTwelveLeft)
coordinates <- gridTwelveLeft


## End(Not run)
```

`gridTwelveRight` *Grid Twelve Right*

Description

This returns a twelve grid right maze. These are standardized coordinates.

Usage

```
data(gridTwelveRight)
```

Format

A data frame with 2 columns

start Coordinates of Start Node.

end Coordinates End Node.

Examples

```
## Not run:

# Returns a Grid with rank = 12
data(gridTwelveRight)
coordinates <- gridTwelveRight


## End(Not run)
```

`gridTwelveUp` *Grid Twelve Up*

Description

This returns a twelve grid upwards maze. These are standardized coordinates.

Usage

```
data(gridTwelveUp)
```

Format

A data frame with 2 columns

start Coordinates of Start Node.

end Coordinates End Node.

Examples

```
## Not run:

# Returns a Grid with rank = 12
data(gridTwelveUp)
coordinates <- gridTwelveUp


## End(Not run)
```

gridTwentyDown *Grid Twenty Down*

Description

This returns a twenty grid right maze. These are standardized coordinates.

Usage

```
data(gridTwentyDown)
```

Format

A data frame with 2 columns

start Coordinates of Start Node.

end Coordinates End Node.

Examples

```
## Not run:

# Returns a Grid with rank = 20
data(gridTwentyDown)
coordinates <- gridTwentyDown


## End(Not run)
```

gridTwentyLeft *Grid Twenty Left*

Description

This returns a twenty grid right maze. These are standardized coordinates.

Usage

```
data(gridTwentyLeft)
```

Format

A data frame with 2 columns

start Coordinates of Start Node.

end Coordinates End Node.

Examples

```
## Not run:

# Returns a Grid with rank = 20
data(gridTwentyLeft)
coordinates <- gridTwentyLeft


## End(Not run)
```

gridTwentyRight *Grid Twenty Right*

Description

This returns a twenty grid right maze. These are standardized coordinates.

Usage

```
data(gridTwentyRight)
```

Format

A data frame with 2 columns

start Coordinates of Start Node.

end Coordinates End Node.

Examples

```
## Not run:

# Returns a Grid with rank = 20
data(gridTwentyRight)
coordinates <- gridTwentyRight


## End(Not run)
```

---

gridTwentyUp *Grid Twenty Up*

Description

This returns a twenty grid right maze. These are standardized coordinates.

Usage

```
data(gridTwentyUp)
```

Format

A data frame with 2 columns

start Coordinates of Start Node.

end Coordinates End Node.

Examples

```
## Not run:

# Returns a Grid with rank = 20
data(gridTwentyUp)
coordinates <- gridTwentyUp


## End(Not run)
```

howMany *howMany*

Description

Calculate how many possible variation of black dotes for a given saturation.

Usage

```
howMany(rank, satPercent)
```

Arguments

rank	This is the rank of the maze.
satPercent	The percentage of saturation. Between 0-1.

Details

Calculate how many possible variation of black dotes for a given saturation. The first node will not be a black dot.

Author(s)

Aiden Loe

See Also

lowerGrid

Examples

```
howMany(rank=5, satPercent=0.5)
```

lowerGrid *lowerGrid*

Description

This tells you all the node position in the maze.

Usage

```
lowerGrid(rank = 5)
```

Arguments

rank	This is the rank of the maze.

Details

The construction of the maze is first created in a symmetrical format. However, only half of the nodes are kept in order to create the actual maze. Hence, this function calculates the nodePosition of the actual maze.

Author(s)

Aiden Loe

Examples

```
lowerGrid(3)
```

`maxScore` *Maximum Score*

Description

This returns the maximum score for a given rank and a given colour node position.

Usage

```
maxScore(nodePosition)
```

Arguments

`nodePosition`	The position of the black dots.

Details

The maxScore function returns the maximum score for a given rank and a given colour node positions. You need to use the colour node position function first.

Author(s)

Aiden Loe

Examples

```

nodePosition <- np(rank=3,satPercent=0.5,seed=1)

maxScore(nodePosition=nodePosition)
```

mazeAbility *mazeAbility*

Description

The ability function returns the weighted score of the individual given his raw score (i.e. the number of black dotes collected).

Usage

```
mazeAbility(nodePosition, dot = 2, model = "t2")
```

Arguments

`nodePosition`	You need to calculate the nodePosition.
`dot`	This is the number of black dots.
`model`	There are 4 models to estimate ability (t1,t2,t3,t4).

Details

This function calculates the weighted score of the participant given the number of dots collected. The function adopts 4 different models which follows the Davies & Davies (1965) paper. The formula for is Model 1:

$$log(2^R/U_m)$$

where 2^R is the total number of paths and U_m is the paths through the specified number of dots. The formula for Model 2:

$$log(U_{\hat{m}}/U_m)$$

where $U_{\hat{m}}$ is the value with the maximum number of connected dots. The formula for Model 3:

$$log(2^R * s^4/U_m)$$

where s^4 is the saturation value. The formula for Model 4 is:

$$log(U_{\hat{m}} * s^4/U_m)$$

We included all four models to calculate maze ability.

Value

An 'ab' class is created which will be used for other functions in the package.

Author(s)

Aiden Loe and Maria Sanchez

See Also

mazeDiff, np

Examples

```
nodePosition <- np(rank=6,satPercent=0.5,seed=1)
mazeAbility(nodePosition,dot=3, model="t2")
```

mazeDiff *Maze Diffculty*

Description

This function tells us the difficulty level of the rank given a saturation and black node distribution

Usage

```
mazeDiff(nodePosition, model = "m1")
```

Arguments

nodePosition	This is the distribution of the colour node positions.
model	There are three types of model to select from: "m1", "m2" or "m3".

Details

This function tells us the difficulty level of the rank given a saturation and black node distribution. The calculation of the difficulty level follows the Davies & Davies (1965) paper. In the article, there are three ways to calculate maze difficulty. In Model 1, only two parameters were considered: rank and the number of possible paths through the maximum number of routes.

$$log(2^R/U_{\hat{m}})$$

where 2^R is the total number of paths and $U_{\hat{m}}$ is the paths through the maximum number of dots. Model 2 includes the saturation parameter. This is calculated based on:

$$log(2^R * s^a/U_{\hat{m}})$$

where s is the saturation and $a = 4$. The a value is recommended in the paper after using various values. Model 3 extends the second formula to include the minimum number of steps to pass through $\hat{m}$.

$$log(2^R * s^a * l^b / U_{\hat{m}})$$

where l is the minimum steps to pass through $\hat{m}$ and $b = 4$. The b value is recommended in the paper after using various values.

We included all three approaches to calculate maze difficulty. It was to incorporated all the possible parameters of the task features that may potentially influence maze difficulty.

Author(s)

Aiden Loe and Maria Sanchez

References

Davies, A. D., & Davies, M. G. (1965). The difficulty and graded scoing of Elithorn's perceptual maze test. *British Journal of Psychology, 56(2-3)*, 295-302.

See Also

`mazeEst`, `mazeAbility`, `np`

Examples

```
#Black nodes distribution
nodePosition <- np(rank=5,satPercent=0.5,seed=1)

#calculate difficulty
mazeDiff(nodePosition, model="m1")
```

mazeEst *Calculate Maze Parameters*

Description

This returns the estimate of various maze parameters.

Usage

```
mazeEst(nodePosition)
```

Arguments

nodePosition	Tells you all the position of the black dots.

Details

This function calculates the count of all the possible black node routes, the maximum score one can achieve for a given rank of a colour node position, all the minimum routes possible, and all the possible routes.

Value

rank The rank of the maze

nodePosition The location of the coloured dots

maxScore The maximum score achievable in the maze.

possibleBlackNodeRoutes All possible routes that passes a certain number of black dots

minStep The minimum steps to achieve the maximum score

allminPath The number of paths with the minimum steps to achieve the maximum score.

minRoutes All the paths with the minimum steps to achieve the maximum score.

allPath The number of possible paths to achieve the maximum score.

maxScoreRoutes All possible paths to achieve the maximum score.

Author(s)

Aiden Loe

References

Davies, A. D., & Davies, M. G. (1965). The difficulty and graded scoing of Elithorn's perceptual maze test. British Journal of Psychology, 56(2-3), 295-302.

Davies, M. G., & Davies, D. M. (1965). Some analytical properties of Elithorn's perceptual maze. Journal of Mathematical Psychology, 2(2), 371-380.

See Also

np, mazeDiff, mazeAbility

Examples

```
rank <- 10
nodePosition <- np(rank=10,satPercent=0.5,seed=16)
c <- mazeEst(nodePosition)
```

mazeGen *mazeGen: A package for generating Elithorn Maze*

Description

The mazeGen package provides a function to generate the Perceptual Elithorn Maze as well as the methods for calculating task difficulty without incorporating reponses.

Details

The mazeHTML or the link{mazeObject} function will allow you to generate the mazes according to certain specification. Currently the maximum number of row is 18. To get a summary of the maze parameters, users can use the mazeEst.

For most functions to work, you need to first get the random distribution of the coloured nodes. Using the np function will allow you to do that. There are occasions where one might want to select the number of paths for a maximum score for a given maze with a known saturation.

Calculating the maximum score for the random coloured node distribution can be done using the maxScore function. At this stage, there is no way in generating a maze based on a pre-determined specific maximum score. The maze generation is largely depending on the rank, and the saturation of the coloured nodes.

The genPathSeed function will search for the seed that returns the specific paths for a given maximum score when using it in the np function. Alternatively, one may use the genEMLseed function to search for the seed that returns the specific paths for a maximum score, with the notion that the minimum number of steps to achieve maximum score is the same for all possible paths. Once the seed is return, one can use it in the np function. Bear in mind that the SEED is restricted to the local computer.

The difficulty of the maze can be calculated using the mazeDiff. Using this approch does not consider player's responses but just the parameters involve to create the maze. Three models are used to calculate the maze difficulty using the function.

The ability score of the participants can be calculated using the mazeAbility. There are 4 different models used to calcuate the participants' ability.

Use the mazeHTML function to generate the maze in a HTML template or the mazeObject function to generate the maze in an R object. To use it with concerto, it is better to generate the maze in the R object and push it into a HTML template. This will allow an immediate generation of the maze in test mode.

References

Davies, A. D., & Davies, M. G. (1965). The difficulty and graded scoing of Elithorn's perceptual maze test. *British Journal of Psychology, 56(2-3)*, 295-302.

Davies, M. G., & Davies, D. M. (1965). Some analytical properties of Elithorn's perceptual maze. *Journal of Mathematical Psychology, 2(2)*, 371-380.

---

mazeHTML *Generate Elithorn Maze*

Description

This function generates an Elithorn Maze

Usage

```
mazeHTML(rank = 3, satPercent = 0.5, seed = 1, grid = NULL, wd = NULL,
  background = "#7abcff", boxBackground = "#66CDAA", fontColour = "white",
  Timer = TRUE, concerto = "C5")
```

Arguments

`rank`	This is the Rank of the maze.
`satPercent`	The saturation of the number of black dots created for a given grid. Range between 0-1.
`seed`	To make sure that the randomness of the created black dots is captured and not repeated.
`grid`	is the grid of the maze
`wd`	is the working directory to save the HTML source code in. If not given, the file will be saved in the default working directory.
`background`	The background colour of the page.
`boxBackground`	The background colour of the box.
`fontColour`	The font colour of the instructions.
`Timer`	If True, a time limit of 1 mintues and 30 seconds is given per question.
`concerto`	The code varies between concerto version "C4" and "C5".

Details

This function creates a maze and is saved into your working directory. A grid object needs to be called out first before runing the maze function. The grid object needs to be the same as the rank given.

Author(s)

Aiden Loe

See Also

mazeAbility, mazeDiff, np

Examples

```
rank <- 3
satPercent <- 0.5

#Grid must be same as rank
grid <- gridThreeUp

#Folder to save html/
#setwd("~/desktop")
#filePath<- getwd()

#Generate item
mazeHTML(rank,satPercent,seed=5,grid = grid,wd=NULL,
background="#7abcff",boxBackground="#66CDAA", fontColour="white ",
Timer=TRUE, concerto="C5")
```

---

mazeObject *Generate Elithorn Maze*

Description

This function generates the html template of the Elithorn Maze in an R object.

Usage

```
mazeObject(rank = 3, satPercent = 0.5, seed = 1, grid = NULL,
  background = "#7abcff", boxBackground = "#66CDAA", fontColour = "white",
  Timer = TRUE, concerto = "C5")
```

Arguments

rank	This is the Rank of the maze.
satPercent	The saturation of the number of black dots created for a given grid. Range between 0-1.
seed	To make sure that the randomness of the created black dots is captured and not repeated.
grid	is the grid of the maze
background	The background colour of the page.
boxBackground	The background colour of the box.
fontColour	The font colour of the instructions.
Timer	If True, a time limit of 4 mintues is given per question.
concerto	The code varies between concerto version "C4" and "C5".

Details

This function creates a plot with the maze blueprint into your working directory. A grid object needs to be called out first before runing the maze function. The grid object needs to be the same as the rank given.

Author(s)

Aiden Loe

See Also

mazeAbility, mazeDiff, np

Examples

```
rank <- 3
satPercent <- 0.5

#Grid must be same as rank
grid <- gridThreeUp

#Generate item
mazeObject(rank,satPercent,seed=5,grid = grid,
background="#7abcff",boxBackground="#66CDAA", fontColour="white ",
Timer=TRUE, concerto="C5")
```

---

np *Colour Node Position*

Description

Returns the colour node position. You need to use the node position function first.

Usage

```
np(rank = 3, satPercent = 0.5, seed = 1)
```

Arguments

rank	This is the rank of the maze.
satPercent	Percentage of saturation.
seed	To always get the same position for a local computer.

Details

This function will not sample from the first node position. If you consider sampling from the first node, then in javascript, the summing of the black dotes need to begin from 1 rather than 0. To keep it simple, always ensure that the first node is not sampled as a black dot.

Value

A 'np' class which will be used for other functions in the package.

Author(s)

Aiden Loe

See Also

mazeEst, genPathSeed

Examples

```
np(rank=3,satPercent=0.5,seed=1)
```

---

topNodes *Top Nodes*

Description

The node length calculates all the nodes on the longest row for a given rank.

Usage

```
topNodes(rank)
```

Arguments

rank	This is the Rank of the maze.

Details

This needs to have a rank value of greater than 1. This is needed so that you can cross check how many coloured nodes are located on the longest row.

Author(s)

Aiden Loe

Examples

```
rank <-3
topNodes(rank)
```

Index

∗ **datasets**
gridEightDown, 6
gridEighteenDown, 7
gridEighteenLeft, 7
gridEighteenRight, 8
gridEighteenUp, 9
gridEightLeft, 9
gridEightRight, 10
gridEightUp, 11
gridElevenDown, 11
gridElevenLeft, 12
gridElevenRight, 13
gridElevenUp, 13
gridFifteenDown, 14
gridFifteenLeft, 15
gridFifteenRight, 15
gridFifteenUp, 16
gridFiveDown, 17
gridFiveLeft, 17
gridFiveRight, 18
gridFiveUp, 19
gridFourDown, 19
gridFourLeft, 20
gridFourRight, 21
gridFourteenDown, 21
gridFourteenLeft, 22
gridFourteenRight, 23
gridFourteenUp, 23
gridFourUp, 24
gridNineDown, 25
gridNineLeft, 25
gridNineRight, 26
gridNineteenUp, 27
gridNineUp, 28
gridSevenDown, 29
gridSevenLeft, 29
gridSevenRight, 30
gridSeventeenDown, 31
gridSeventeenLeft, 31
gridSeventeenRight, 32
gridSeventeenUp, 33
gridSevenUp, 33
gridSixDown, 34
gridSixLeft, 35
gridSixRight, 35
gridSixteenDown, 36
gridSixteenLeft, 37
gridSixteenRight, 37
gridSixteenUp, 38
gridSixUp, 39
gridTenDown, 39
gridTenLeft, 40
gridTenRight, 41
gridTenUp, 41
gridThirteenDown, 42
gridThirteenLeft, 43
gridThirteenRight, 43
gridThirteenUp, 44
gridThreeDown, 45
gridThreeLeft, 45
gridThreeRight, 46
gridThreeUp, 47
gridTwelveDown, 47
gridTwelveLeft, 48
gridTwelveRight, 49
gridTwelveUp, 49
gridTwentyDown, 50
gridTwentyLeft, 51
gridTwentyRight, 51
gridTwentyUp, 52

∗ **lowerGrid**
howMany, 53

genEMLseed, 3, *5*, *6*, *59*
genMaze, 4
genPathSeed, *4*, 5, *59*, *63*
gridEightDown, 6
gridEighteenDown, 7
gridEighteenLeft, 7

gridEighteenRight, 8
gridEighteenUp, 9
gridEightLeft, 9
gridEightRight, 10
gridEightUp, 11
gridElevenDown, 11
gridElevenLeft, 12
gridElevenRight, 13
gridElevenUp, 13
gridFifteenDown, 14
gridFifteenLeft, 15
gridFifteenRight, 15
gridFifteenUp, 16
gridFiveDown, 17
gridFiveLeft, 17
gridFiveRight, 18
gridFiveUp, 19
gridFourDown, 19
gridFourLeft, 20
gridFourRight, 21
gridFourteenDown, 21
gridFourteenLeft, 22
gridFourteenRight, 23
gridFourteenUp, 23
gridFourUp, 24
gridNineDown, 25
gridNineLeft, 25
gridNineRight, 26
gridNineteenDown *(*gridNineteenUp*)*, 27
gridNineteenLeft *(*gridNineteenUp*)*, 27
gridNineteenRight *(*gridNineteenUp*)*, 27
gridNineteenUp, 27
gridNineUp, 28
gridSevenDown, 29
gridSevenLeft, 29
gridSevenRight, 30
gridSeventeenDown, 31
gridSeventeenLeft, 31
gridSeventeenRight, 32
gridSeventeenUp, 33
gridSevenUp, 33
gridSixDown, 34
gridSixLeft, 35
gridSixRight, 35
gridSixteenDown, 36
gridSixteenLeft, 37
gridSixteenRight, 37
gridSixteenUp, 38
gridSixUp, 39
gridTenDown, 39
gridTenLeft, 40
gridTenRight, 41
gridTenUp, 41
gridThirteenDown, 42
gridThirteenLeft, 43
gridThirteenRight, 43
gridThirteenUp, 44
gridThreeDown, 45
gridThreeLeft, 45
gridThreeRight, 46
gridThreeUp, 47
gridTwelveDown, 47
gridTwelveLeft, 48
gridTwelveRight, 49
gridTwelveUp, 49
gridTwentyDown, 50
gridTwentyLeft, 51
gridTwentyRight, 51
gridTwentyUp, 52

howMany, 53

lowerGrid, *53*, 53

maxScore, 54, *59*
mazeAbility, 55, *57–59*, *61, 62*
mazeDiff, *56*, 56, *58, 59*, *61, 62*
mazeEst, *4*, *6*, *57*, 57, *59*, *63*
mazeGen, 59
mazeHTML, *59*, 60
mazeObject, *59*, 61

np, *4*, *6*, *56–59*, *61, 62*, 62

topNodes, 63