

# Package ‘CellularAutomaton’

February 14, 2012

**Type** Package

**Title** One-Dimensional Cellular Automata

**Version** 1.1

**Date** 2011-12-28

**Author** John Hughes

**Maintainer** John Hughes <hughesj@umn.edu>

**Depends** R.oo

**Description** This package is an object-oriented implementation of one-dimensional cellular automata. It supports many of the features offered by Mathematica, including elementary rules, user-defined rules, radii, user-defined seeding, and plotting.

**License** GPL

**Repository** CRAN

**Date/Publication** 2011-12-30 16:26:34

## R topics documented:

CellularAutomaton-package . . . . .	2
CellularAutomaton . . . . .	3
getLattice . . . . .	4
getLattice.CellularAutomaton . . . . .	4
getNumberOfColors . . . . .	5
getNumberOfColors.CellularAutomaton . . . . .	5
getRadius . . . . .	6
getRadius.CellularAutomaton . . . . .	6
getRuleNumber . . . . .	7
getRuleNumber.CellularAutomaton . . . . .	7
getSteps . . . . .	8
getSteps.CellularAutomaton . . . . .	8

getTotalistic . . . . .	9
getTotalistic.CellularAutomaton . . . . .	9
plot . . . . .	10
plot.CellularAutomaton . . . . .	10

<b>Index</b>	<b>11</b>
--------------	-----------

---

CellularAutomaton-package

*One-Dimensional Cellular Automata*

---

## Description

This package is an object-oriented implementation of one-dimensional cellular automata. It supports many of the features offered by Mathematica, including elementary rules, user-defined rules, radii, user-defined seeding, and plotting.

## Details

Package: CellularAutomaton  
 Type: Package  
 Version: 1.1  
 Date: 2011-12-28  
 License: GPL

## Author(s)

John Hughes

Maintainer: John Hughes <hughesj@umn.edu>

## References

<http://reference.wolfram.com/mathematica/tutorial/CellularAutomata.html>

## Examples

```
ca = CellularAutomaton(t = 100) # Evolve Rule 30 for 100 steps. k = 2, r = 1, and the seed
# is a single black cell on a white background. Each row will
# have length 2rt + 1 = 201.
ca$plot() # Have a look.
```

```
# Evolve Rule 110 for 100 steps. k = 2, r = 1, and the seed is 001000. Each row will have the
# same length as the seed because -1 was given as the background.
```

```
ca = CellularAutomaton(n = 110, t = 100, seed = c(0, 0, 1, 0, 0, 0), bg = -1)
```

```
ca$plot(col = c("white", "darkblue")) # Plot it using Penn State colors. :-)
```

---

CellularAutomaton      *Constructor for Class CellularAutomaton*

---

### Description

This method instantiates class CellularAutomaton.

### Usage

```
CellularAutomaton(n = 30, fun = NULL, k = 2, r = 1, t = 1,  
                 totalistic = 0, seed = 1, bg = 0)
```

### Arguments

n	This is the elementary rule number for the automaton.
fun	This is a user-defined rule.
k	This is the number of colors.
r	This is the radius of a neighborhood.
t	This is the number of steps.
totalistic	0 = general; 1 = totalistic
seed	This is a seed for the automaton.
bg	This is the background upon which to place the seed.

### Value

This method returns an instance of class CellularAutomaton, provided the arguments make sense.

### Author(s)

John Hughes

### References

<http://reference.wolfram.com/mathematica/tutorial/CellularAutomata.html>

**Examples**

```

ca = CellularAutomaton(t = 100) # Evolve Rule 30 for 100 steps. k = 2, r = 1, and the seed
                                # is a single black cell on a white background. Each row will
                                # have length 2rt + 1 = 201.
ca$plot()                       # Have a look.

# Evolve Rule 110 for 100 steps. k = 2, r = 1, and the seed is 001000. Each row will have the
# same length as the seed because -1 was given as the background.

ca = CellularAutomaton(n = 110, t = 100, seed = c(0, 0, 1, 0, 0, 0), bg = -1)

ca$plot(col = c("white", "darkblue")) # Plot it using Penn State colors. :-)
```

---

```
getLattice
```

*Lattice of Cells of a One-Dimensional Cellular Automaton*

---

**Description**

This method extracts the matrix of cells from an instance of class CellularAutomaton.

**Details**

```
ca$getLattice()
```

**Value**

getLattice returns a matrix of nonnegative integers. Each row of the matrix represents one generation in the evolution of the automaton.

**Author(s)**

John Hughes

---

```
getLattice.CellularAutomaton
```

*Lattice of Cells of a One-Dimensional Cellular Automaton*

---

**Description**

This method extracts the matrix of cells from an instance of class CellularAutomaton.

**Details**

```
ca$getLattice()
```

**Value**

`getLattice` returns a matrix of nonnegative integers. Each row of the matrix represents one generation in the evolution of the automaton.

**Author(s)**

John Hughes

---

`getNumberOfColors`      *Number of Colors of a One-Dimensional Cellular Automaton*

---

**Description**

This method extracts the number of colors from an instance of class `CellularAutomaton`.

**Details**

`ca$getNumberOfColors()`

**Value**

`getNumberOfColors` returns an integer  $\geq 2$ .

**Author(s)**

John Hughes

---

`getNumberOfColors.CellularAutomaton`  
*Number of Colors of a One-Dimensional Cellular Automaton*

---

**Description**

This method extracts the number of colors from an instance of class `CellularAutomaton`.

**Details**

`ca$getNumberOfColors()`

**Value**

`getNumberOfColors` returns an integer  $\geq 2$ .

**Author(s)**

John Hughes

---

`getRadius`*Radius of a One-Dimensional Cellular Automaton*

---

**Description**

This method extracts the radius from an instance of class CellularAutomaton.

**Details**`ca$getRadius()`**Value**

`getRadius` returns an integer  $\geq 1$ .

**Author(s)**

John Hughes

---

`getRadius.CellularAutomaton`*Radius of a One-Dimensional Cellular Automaton*

---

**Description**

This method extracts the radius from an instance of class CellularAutomaton.

**Details**`ca$getRadius()`**Value**

`getRadius` returns an integer  $\geq 1$ .

**Author(s)**

John Hughes

---

getRuleNumber

*Elementary Rule of a One-Dimensional Cellular Automaton*

---

**Description**

This method extracts the rule number from an instance of class CellularAutomaton.

**Details**

ca\$getRuleNumber()

**Value**

getRuleNumber returns the rule number for the automaton, provided that an elementary rule was specified by the user. If the user supplied his/her own rule, then this method returns -1.

**Author(s)**

John Hughes

---

getRuleNumber.CellularAutomaton

*Elementary Rule of a One-Dimensional Cellular Automaton*

---

**Description**

This method extracts the rule number from an instance of class CellularAutomaton.

**Details**

ca\$getRuleNumber()

**Value**

getRuleNumber returns the rule number for the automaton, provided that an elementary rule was specified by the user. If the user supplied his/her own rule, then this method returns -1.

**Author(s)**

John Hughes

---

`getSteps`*Number of Steps of a One-Dimensional Cellular Automaton*

---

**Description**

This method extracts the number of steps (generations) from an instance of class CellularAutomaton.

**Details**`ca$getSteps()`**Value**

`getSteps` returns an integer  $\geq 1$ .

**Author(s)**

John Hughes

---

`getSteps.CellularAutomaton`*Number of Steps of a One-Dimensional Cellular Automaton*

---

**Description**

This method extracts the number of steps (generations) from an instance of class CellularAutomaton.

**Details**`ca$getSteps()`**Value**

`getSteps` returns an integer  $\geq 1$ .

**Author(s)**

John Hughes

---

`getTotalistic`

*Totalistic of a One-Dimensional Cellular Automaton*

---

**Description**

This method extracts the setting of `totalistic` from an instance of class `CellularAutomaton`.

**Details**

`ca$getTotalistic()`

**Value**

`getTotalistic` returns 0 or 1. A 0 indicates a general automaton. A 1 indicates a totalistic automaton. Outer-totalistic rules are not currently supported.

**Author(s)**

John Hughes

---

`getTotalistic.CellularAutomaton`

*Totalistic of a One-Dimensional Cellular Automaton*

---

**Description**

This method extracts the setting of `totalistic` from an instance of class `CellularAutomaton`.

**Details**

`ca$getTotalistic()`

**Value**

`getTotalistic` returns 0 or 1. A 0 indicates a general automaton. A 1 indicates a totalistic automaton. Outer-totalistic rules are not currently supported.

**Author(s)**

John Hughes

plot

*Plot a One-Dimensional Cellular Automaton*

---

**Description**

This method plots an instance of class CellularAutomaton.

**Arguments**

col                    a vector of colors

**Details**

This method uses `image()` to plot the automaton. The plot displays the automaton's steps in increasing order from top to bottom. The user may specify a vector of colors to be used by the plot. The default is `0:(k - 1)`, where `k` is the number of colors for the automaton.

```
ca$plot()
```

```
ca$plot(col = c(3, 1, 4))
```

**Author(s)**

John Hughes

---

plot.CellularAutomaton

*Plot a One-Dimensional Cellular Automaton*

---

**Description**

This method plots an instance of class CellularAutomaton.

**Arguments**

col                    a vector of colors

**Details**

This method uses `image()` to plot the automaton. The plot displays the automaton's steps in increasing order from top to bottom. The user may specify a vector of colors to be used by the plot. The default is `0:(k - 1)`, where `k` is the number of colors for the automaton.

```
ca$plot()
```

```
ca$plot(col = c(3, 1, 4))
```

**Author(s)**

John Hughes

# Index

## \*Topic **hplot**

plot, 10  
plot.CellularAutomaton, 10

## \*Topic **methods**

CellularAutomaton, 3  
getLattice, 4  
getLattice.CellularAutomaton, 4  
getNumberOfColors, 5  
getNumberOfColors.CellularAutomaton,  
5  
getRadius, 6  
getRadius.CellularAutomaton, 6  
getRuleNumber, 7  
getRuleNumber.CellularAutomaton, 7  
getSteps, 8  
getSteps.CellularAutomaton, 8  
getTotalistic, 9  
getTotalistic.CellularAutomaton, 9  
plot, 10  
plot.CellularAutomaton, 10

## \*Topic **package**

CellularAutomaton, 3  
CellularAutomaton-package, 2

CellularAutomaton, 3

CellularAutomaton-package, 2

getLattice, 4  
getLattice.CellularAutomaton, 4  
getNumberOfColors, 5  
getNumberOfColors.CellularAutomaton, 5  
getRadius, 6  
getRadius.CellularAutomaton, 6  
getRuleNumber, 7  
getRuleNumber.CellularAutomaton, 7  
getSteps, 8  
getSteps.CellularAutomaton, 8  
getTotalistic, 9  
getTotalistic.CellularAutomaton, 9

plot, 10

plot.CellularAutomaton, 10