Package 'LindenmayeR'

May 7, 2026

Type Package

Title Functions to Explore L-Systems (Lindenmayer Systems)

Version 0.1.13

Date 2017-07-31

Description L-systems or Lindenmayer systems are parallel rewriting systems which can be used to simulate biological forms and certain kinds of fractals. Briefly, in an L-system a series of symbols in a string are replaced iteratively according to rules to give a more complex string. Eventually, the symbols are translated into turtle graphics for plotting. Wikipedia has a very good introduction: en.wikipedia.org/wiki/L-system This package provides basic functions for exploring L-systems.

License GPL (>= 3)

Depends stringr, grid

ByteCompile TRUE

BugReports https://github.com/bryanhanson/LindenmayeR/issues

RoxygenNote 6.0.1

NeedsCompilation no

Author Bryan Hanson [aut, cre]

Maintainer Bryan Hanson <hanson@depauw.edu>

Repository CRAN

Date/Publication 2017-07-31 21:52:31 UTC

Contents

LindenmayeR-package *Lindenmayer System Functions*

Description

Functions to Explore L-Systems (Lindenmayer Systems)

Details

Lindenmayer or L-systems are parallel rewriting systems which can be used to simulate biological forms and certain kinds of fractals. Briefly, in an L-system a series of symbols in a string are replaced iteratively according to rules to give a more complex string. Eventually, the symbols are translated into turtle graphics for plotting. Wikipedia has a very good introduction: https://en.wikipedia.org/wiki/L-system This package provides basic functions for exploring L-systems.

Author(s)

Bryan A. Hanson

drawLsys *Draw a 2D L-System Using Turtle Graphics*

Description

This function takes input strings, previously created with Lsys, translates them into 2D turtle graphics instructions, and then plots the results.

Usage

```
drawLsys(string = NULL, drules = NULL, st = c(5, 50, 0), stepSize = 1,
  ang = 90, which = length(string), shrinkFactor = NULL, ...)
```

Arguments

string	A character vector giving the strings containing the turtle graphics instructions. Created by Lsys. The "language" and character set of this string is arbitary. Compare the examples below for the modified Koch curve and the Sierpinski triangle.
drules	A data frame containing columns "symbols" and "action". These contain the input symbols and the corresponding drawing action. The symbol column is in the character set used by Lsys and is arbitary. The action column entries must be from the set c("F", "f", "+", "-", "[", "]"). These are the final drawing instructions and are interpreted as follows: **"F"** Move forward drawing as you go. **"f"** Move forward w/o drawing.

"+" Turn by positive ang.

"-" Turn by negative ang.

"[" Save current position and heading.

" "] Restore saved position and heading (allows one to go back).

See the examples. Note that the "action" entry always uses these symbols, though not all of them need be used.

st	A numeric vector of length 3 giving the screen coordinates where the start of the curve should be placed. The screen is 100 x 100 with the lower left corner as 0,0. The third element is the initial drawing angle in degrees.
stepSize	Numeric. The length of the drawing step.
ang	Numeric. The angle in degrees when a change in direction is requested.
which	Integer. The entries in string which should be drawn. Defaults to the last (most complex) entry. If length(which) > 1 each plot is drawn in its own window.
shrinkFactor	A numeric vector of the same length as string. As each plot is made, stepSize will be divided by the corresponding value in shrinkFactor. This allows one to scale down the increasingly large/complex plots to make them occupy a space similar to the less complex plots.
...	Additional parameters to be passed to the grid drawing routines. Most likely, something of the form gp = gpar(...). See gpar and the last example.

Value

None; side effect is a plot.

Warning

Remember that if retAll = TRUE, Lsys returns the initial string plus the results of all iterations. In this case, if you want the 5th iteration, you should specify which = 6 since the initial string is in string[1].

Examples

```
require('grid')

# Modified Koch curve
rkoch1 <- data.frame(inp = c("F"), out = c("F+F-F-F+F"), stringsAsFactors = FALSE)
k1 <- Lsys(init = "F", rules = rkoch1, n = 3)
dkoch <- data.frame(symbol = c("F", "f", "+", "-", "[", "]"),
action = c("F", "f", "+", "-", "[", "]"), stringsAsFactors = FALSE)
drawLsys(string = k1, stepSize = 3, st = c(10, 50, 0), drules = dkoch)
grid.text("Modified Koch Curve (n = 3)", 0.5, 0.25)

# Classic Koch snowflake
rkoch2 <- data.frame(inp = c("F"), out = c("F-F++F-F"), stringsAsFactors = FALSE)
k2 <- Lsys(init = "F++F++F", rules = rkoch2, n = 4)
drawLsys(string = k2, stepSize = 1, ang = 60, st = c(10, 25, 0), drules = dkoch)
grid.text("Classic Koch Snowflake (n = 4)", 0.5, 0.5)
```

```
# Sierpinski Triangle
rSierp <- data.frame(inp = c("A", "B"), out = c("B-A-B", "A+B+A"), stringsAsFactors = FALSE)
s <- Lsys(init = "A", rules = rSierp, n = 6)
dSierp <- data.frame(symbol = c("A", "B", "+", "-", "[", "]"),
action = c("F", "F", "+", "-", "[", "]"), stringsAsFactors = FALSE)
drawLsys(string = s, stepSize = 1, ang = 60, st = c(20, 25, 0), drules = dSierp)
grid.text("Sierpinski Triangle (n = 6)", 0.5, 0.1)

# Islands & Lakes
islands_rules <- data.frame(inp = c("F", "f"), out = c("F+f-FF+F+FF+Ff+FF-f+FF-F-FF-Ff-FFF",
"ffffff"), stringsAsFactors = FALSE)
islands <- Lsys(init = "F+F+F+F", rules = islands_rules, n = 2)
draw_islands <- data.frame(symbol = c("F", "f", "+", "-", "[", "]"),
action = c("F", "f", "+", "-", "[", "]"), stringsAsFactors = FALSE)
drawLsys(string = islands, step = 1, ang = 90, st = c(70, 35, 90),
drules = draw_islands,  gp = gpar(col = "red", fill = "gray"))

# A primitive tree (aka Pythagoras Tree)
prim_rules <- data.frame(inp = c("0", "1"),
out = c("1[+0]-0", "11"), stringsAsFactors = FALSE)
primitive_plant <- Lsys(init = "0", rules = prim_rules, n = 7)
draw_prim <- data.frame(symbol = c("0", "1", "+", "-", "[", "]"),
action = c("F", "F", "+", "-", "[", "]"), stringsAsFactors = FALSE)
drawLsys(string = primitive_plant, stepSize = 1, ang = 45, st = c(50, 5, 90),
drules = draw_prim, which = 7)
grid.text("Primitive Tree (n = 6)", 0.5, 0.75)

# A more realistic botanical structure
# Some call this a fractal tree, I think it is more like seaweed
# Try drawing the last iteration (too slow for here, but looks great)
fractal_tree_rules <- data.frame(inp = c("X", "F"),
out = c("F-[[X]+X]+F[+FX]-X", "FF"), stringsAsFactors = FALSE)
fractal_tree <- Lsys(init = "X", rules = fractal_tree_rules, n = 7)
draw_ft <- data.frame(symbol = c("X", "F", "+", "-", "[", "]"),
action = c("f", "F", "+", "-", "[", "]"), stringsAsFactors = FALSE)
drawLsys(string = fractal_tree, stepSize = 2, ang = 25, st = c(50, 5, 90),
drules = draw_ft, which = 5, gp = gpar(col = "chocolate4", fill = "honeydew"))
grid.text("Fractal Seaweed (n = 4)", 0.25, 0.25)
```

`Lsys` *Rewrite an Axiom Using Production Rules to Give a String Ready for Turtle Graphics*

Description

This is the central function for rewriting an initial string of symbols (the axiom) into a new string using production rules. Production rules are very simple: if the symbol is A, turn it into something. If it is B, turn it into something else. Production rules typically contain instructions about moving while drawing, moving w/o drawing, changing direction, or storing the current state for re-use later.

Usage

```
Lsys(init = NULL, rules = NULL, n = 5, retAll = TRUE, verbose = 1L)
```

Arguments

`init`	A character string giving variables (symbols) to use as the initial string Also known as the axiom.
`rules`	A data frame containing columns "inp" and "out". These contain the input variables and the corresponding replacement string. See the examples in `drawLsys`.
`n`	An integer giving the number of cycles or iterations desired.
`retAll`	Logical. If `TRUE`, the result at each cycle will be returned, otherwise only the last result is returned.
`verbose`	An integer giving the level of information desired as the calculation proceeds. `verbose = 1L` gives basic information at each cycle. Any value greater than 1 gives much more detail. Supress messages by using a value less than 1.

Details

The job of this function is to take an input "axiom" and apply the "production rules" and other paramters to create a new string of drawing instructions. The "language" or character set of the axiom and production rules are arbitary, and the internet and literature contains many different examples. The same fractal could be drawn using completely different sets of symbols. The string produced by this function is processed by `drawLsys`. See there for further explanation and examples.

Value

If `retAll = FALSE`, a character vector of length 1 giving the string at the end processing. Otherwise, a character vector of length `n + 1` containing `init` plus the results at the end of each iteration.

See Also

`drawLsys` for examples, including plotting.

Index