L-Systems in R

Alexander Ptok

29. August 2024

L-Systems

```
lsystem  <- function(alphabet, axiom, productions) {

    derivation  <- function(axiom, new_word) {
        a <- substring(axiom,1,1)
        if (a  == "") {
            new_word
        } else derivation(substring(axiom,2),
                          paste(new_word,
                                productions[a],
                                sep=""))
    }

    derive_n  <- function(axiom, n) {
        if (n == 0) {
            axiom
        } else derive_n(derivation(axiom, ""),
                        n - 1)
    }

    function(n) {
        derive_n(axiom, n)
    }
}

lsystem_iter <- function(alphabet, axiom, productions) {
    function(n) {
        new_word <- ""
        while (n > 0) {
            for (char in as.list(strsplit(axiom,""))[[1]]) {
                new_word  <- paste(new_word,
                                   productions[char],
                                   sep="")
            }
            n <- n - 1
            axiom <- new_word
            new_word <- ""
        }
        axiom
    }
}
```

The Turtle Interpreter

```
turtle <- function(x, y, alpha, stepsize, delta) {

    forward <- function() {
        x <<- x + stepsize * cos(alpha)
        y <<- y + stepsize * sin(alpha)
    }

    forward_draw <- function() {
        linesxy$x1 <<- c(x, linesxy$x1)
        linesxy$y1 <<- c(y, linesxy$y1)
        forward()
        linesxy$x2 <<- c(x, linesxy$x2)
        linesxy$y2 <<- c(y, linesxy$y2)
    }

    turn_right <- function() {
        alpha <<- alpha - delta
```

```
    }

    turn_left <- function() {
        alpha <<- alpha + delta
    }

    linesxy <- list(x1=c(),x2=c(),y1=c(),y2=c())

    function_table <-
        list("F" = forward_draw,
             "f" = forward,
             "-" = turn_right,
             "+" = turn_left)

    rec_over_nu <- function(nu) {
        if (nu == "") {
            linesxy
        } else {
            function_table[[substring(nu,1,1)]]()
            rec_over_nu(substring(nu,2))
        }
    }

    iter_over_nu <- function(nu) {

        for (i in 1:nchar(nu)) {
            function_table[[substring(nu,i,1)]]()
            }
        linesxy
    }

    function(nu) {
      # rec_over_nu(nu)
        iter_over_nu(nu)
    }
}

plotting_turtle <- function(x, y, alpha, stepsize, delta) {

    forward <- function() {
        x <<- x + stepsize * cos(alpha)
        y <<- y + stepsize * sin(alpha)
    }

    forward_draw <- function() {
        x1 <- x
        y1 <- y
        x <<- x + stepsize * cos(alpha)
        y <<- y + stepsize * sin(alpha)
        lines(c(x1,x),c(y1,y))
    }

    turn_right <- function() {
        alpha <<- alpha - delta
    }

    turn_left <- function() {
        alpha <<- alpha + delta
    }

    function_table <-
        list("F" = forward_draw,
             "f" = forward,
             "-" = turn_right,
             "+" = turn_left)
```

```
    iter_over_nu <- function(nu) {

        for (i in 1:nchar(nu)) {
            function_table[[substring(nu,i,i)]]()
        }
    }

    function(nu) {
                                        # rec_over_nu(nu)
        iter_over_nu(nu)
    }
}


draw_turtle <- function(lines) {
    if (length(lines$x1) == 0) {
        TRUE
    } else {
        lines(x=c(lines$x1[1], lines$x2[1]), y=c(lines$y1[1], lines$y2[1]))
        draw_lines(list(x1=lines$x1[-1],
                        x2=lines$x2[-1],
                        y1=lines$y1[-1],
                        y2=lines$y2[-1]))
    }
}


draw_turtle_iter <- function(lines) {
    for (i in 1:length(lines$x1)) {
        lines(x=c(lines$x1[i], lines$x2[i]), y=c(lines$y1[i], lines$y2[i]))
    }
}
```

The Koch Curve

```
source("lsystem.r")
source("turtle.r")
source("draw-turtle.r")
dict <- c("F" = "F-F+F+FF-F-F+F", "+" = "+", "-" = "-", "f" = "f")
l3 <- (lsystem_iter("Ff+-", "F-F-F-F", dict) (3))
f <- plotting_turtle(0.8,0.2,pi,0.03,pi/2)
png("out.png", width=480, height=480)
plot.new(); par(xpd=TRUE)
f(l3)
dev.off()
```