Sierpinski triangle

Y. Paulsen

June 24, 2021

The Chaos Game and the Sierpinski Triangle

The Sierpinski Triangle is a well known fractal in the shape of an equilateral triangle iteratively subdivided into smaller triangles.

The Chaos Game, in its simplest form, uses random sampling from uniform distributions to generate an image of the fractal by recursion. The game itself is simple, and is illustrated in the following pseudocode:

1. create a triangle from three points
2. generate new_point_1 within the triangle by random sampling
3. new_point_2 = midpoint(new_point_1, a randomly chosen vertex of the triangle)
4. iterate Step 3 to get N points within the triangle; N should be large
5. plot the points - they will describe a Sierpinski Triangle.

Below is R code that generates the fractal by playing the Chaos Game. The basic idea of the program follows the pseudocode above.

These first few sections of code set up some parameters and define some functions.

Set the vertices of the triangle.

```
# Vertices for an equilateral triangle.
vertices <- list(A=c(0,0), B=c(6,0), C=c(3,3*sqrt(3)))

# Vertices for a random scalene triangle.
# vertices <- list(A=runif(2,0,10), B=runif(2,0,10), C=runif(2,0,10))
```

Create functions

Here I create one function to find the midpoint between two arbitrary points and another function to randomly sample from a list of coordinates.

```
# Function: calculate midpoint.
midpoint <- function(a,b){
  mid <- c();
  mid[1] <- (a[1]+b[1])/2;
  mid[2] <- (a[2]+b[2])/2;
  return(mid)
}

# Function: call random vertex.
random_vertex <- function(vertices, p=c(1/3,1/3,1/3)){
  vertices[[sample(length(vertices), 1, p=p)]]
}
```

Play the Chaos Game

Here we play the chaos game and plot the results.
I generate the starting point by uniformly sampling from within my triangle according to the methods described in Osada et al.§4.2. This method uses the vertices of a triangle and two values sampled from a uniform distribution between 0 and 1 ($r2$, $r2$) to generate a uniform distribution of points within the triangle.

```
# Set parameters and create objects.
n <- 1000000
plot_points <- list()

# Generate random starting point from within the triangle
r <- runif(2,0,1)
plot_points[[1]] <- (1-sqrt(r[1]))*vertices$A +
  (sqrt(r[1])*(1-r[2]))*vertices$B + (sqrt(r[1])*r[2])*vertices$C

# The chaos game
for(i in 2:n){
  plot_points[[i]] <- midpoint(plot_points[[i-1]], random_vertex(vertices));
}

# Plot the points
x_points <- unlist(plot_points)[c(T,F)]
y_points <- unlist(plot_points)[c(F,T)]
par(mar=c(0,0,0,0), pty="s", pch=17, cex=.15)
plot(x_points, y_points, yaxt="n", ylab="", xaxt='n', xlab = "")
```