Introduction to Statistics and Computation with Data: AppleBread Problem Solution 3

Daibik Barik (Bmat2316)

2024-03-26

```
#set.seed(2024)
library(tidyverse)
```

```
## Warning: package 'ggplot2' was built under R version 4.3.3
```

```
## Warning: package 'tidyr' was built under R version 4.3.3
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.4     v readr     2.1.5
## v forcats   1.0.0     v stringr   1.5.1
## v ggplot2   3.5.0     v tibble    3.2.1
## v lubridate 1.9.3     v tidyr     1.3.1
## v purrr     1.0.2
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

```
library(viridis)
```

```
## Loading required package: viridisLite
```

Challenge 3]

(a)

Diffusion-limited aggregation (DLA) is a phenomenon in which particles, driven by *Brownian motion*, gradually cluster together to form larger aggregates.

This process occurs as individual particles undergo *random walks* and come into contact with each other, leading to the formation of complex and often $fractal$-like structures.

The clusters formed in DLA processes are referred to as *Brownian trees*. These fractals exhibit a dimension of approximately 1.71 for free particles that are unrestricted by a lattice, however computer simulation of DLA on a lattice will change the $fractal$ dimension slightly for a DLA in the same embedding dimension.

(b)

There can be various algorithms, but here I am giving one which I will use in the next question to simulate the plot in R:

1. Initialize the step length, end cluster(*possible stop points*) and end points(*actual points where walk ended*), stickiness(*probability of attaching to a cluster*); along with the boundary(*starting distance*) from which we start the random walks.
2. Now, let's say, we start a random walk from an **uniformly** chosen point from the boundary.
3. We set a an **outer limit**, such that if we see the random walker getting in the opposite direction from the end cluster, we deflect it back inside the *limit radius circle*, and thus increasing probability of it to reach the end cluster.
4. We set **inner limits** (*five of them*), to restrict the movement of the walker outside the *limit radius circle* once it came into the circle, thus limiting the points it can travel to and increasing the probability of it to reach the end cluster.
5. We also add a sticking probability which determines, if and when a particle approches the end cluster, will it attach to the cluster or continue the walk to reach another cluster
6. After the random walk finishes, we update the end points and the end cluster as follows:
    - the ending point is appended into the end points list.
    - the ending point should be present in the end cluster (*otherwise won't be an ending point*), which would be removed and the neighboring points of the ending point would be added in the cluster, **provided they are not already present in both end points and end cluster**.

(c)

```
##Initialization
step_value= c(1, -1, 1i, -1i)
end_cluster= c(0)
end_points= c()
stickiness= 0.5
```

```
##Boundary Values (Taxicab)
distance= 250
boundary_values= c(distance, -distance,
                   distance *1i, -distance *1i)
boundary_point= boundary_values[3]
for(addends in 1:(distance -1)) {
  boundary_point=  boundary_point +1 -1i
  boundary_points= c(boundary_point, Conj(boundary_point),
                     -boundary_point, -Conj(boundary_point))
  boundary_values <- append(boundary_values, boundary_points)
}
```

```
##Boundary Restriction
limits= ceiling(distance* c(1.1, 0.8, 0.6, 0.4, 0.2))
check_limit <- function(point) {
  i = 1
  while(limits[i] >= Mod(point) && i <= length(limits)) {
    i= i +1
  }
  current_limit= limits[i-1]
  probable_steps= step_value
  for(step in step_value) {
    new_point= point +step
    if(Mod(new_point) >= current_limit) {
      probable_steps <- probable_steps[probable_steps != step]
    }
  }
  return(probable_steps)
}
```

```
##End Cluster Modification
end_cluster_modify <- function(end_point) {
  end_cluster <- end_cluster[end_cluster != end_point]
  for(step in step_value) {
    new_point= end_point +step
    if(! new_point %in% end_cluster &&
       ! new_point %in% end_points) {
      end_cluster <- append(end_cluster, new_point)
    }
  }
  return(end_cluster)
}
```

```
##Random Walk
walks= 5000
for(rw in 1:walks) {
  #print(rw)
  position= sample(boundary_values, 1)
  while(TRUE) {
    #print(position)
    step= sample(check_limit(position), 1)
    position= position +step
    if(position %in% end_cluster) {
      stick_prob= runif(1)
      if(stick_prob <stickiness) {
        break
      }
    }
  }
  end_points <- append(end_points, position)
  end_cluster= end_cluster_modify(position)
}
```

```
ggplot() +
  geom_point(mapping= aes(x= Re(end_points), y= Im(end_points)),
             color= turbo(length(end_points)),
             show.legend= FALSE, size= 0.2) +
  theme(
    panel.background= element_rect(fill= "black"),
    panel.grid= element_blank(),
    panel.border= element_blank(),
    axis.title= element_blank(),
    axis.ticks= element_blank(),
    axis.text= element_blank(),
    plot.background= element_rect(fill= "black")
  )
```

