N-dimensional random walk

# Yiyi Yang

## Method

The main purpose of this assignment is to calculate the probability of a particle returning to the Origin after infinite steps of random walk in 1-D, 2-D and 3-D situations.

Previous analytical studies show that the probability should be 1 in 1-D, almost 1 in 2-D, and ~0.34 in 3-D.

The main steps I used to solve this problem are as follows:

1. set a total of N particles
2. let each particle walk a total of n steps (estimate the "infinite steps" or " walk forever" situation using a finite number of steps)
3. Once a given particle returns to the Origin during random walk, stop this walk, and let the next particle start walking
4. Count the total number of particles that ever come back to the Origin
5. The estimated probability = (the total number of particles that come back to the Origin) / (a total of N particles)

## 1-Dimensional

The R code for 1-D is as follow:

# Set total number trail walkers  
walker.number <- 100  
  
# Set total trail steps for each walker  
steps.total <-1000  
  
# the total count of walkers that come back to the origin  
to.zero.n <- 0  
  
# random walk function  
 for (i in 1 : walker.number){  
 x.total <- 0 # set initial position in x direction  
 dx <- 0   
 distance <- 1  
 for (j in 1: steps.total){  
 if (distance == 0){  
 to.zero.n <- to.zero.n+1  
 break  
 } # when a walker come back to the origin, stop and count+1  
 else {  
 dx = floor(runif (1, min=1, max=3))  
 if (dx == 1){  
 x.total <- x.total+dx  
 } # go right  
 else {x.total <- x.total+dx-3  
 } #go left  
 distance <- x.total^2 # calculate the distance from origin  
 }  
 }  
 }  
Prob <- to.zero.n/walker.number # the probability of walkers that ever come back to the origin

I ran this code several times with different walker.number and steps.total. I notice that when steps.total > 10000 and walker.number > 1000, the probability is typically ranging from 0.99 to 1.

So in 1-dimensional case, the probability of the walker coming back to the origin is ~1.

## 2-Dimensional

The code for the 2-d case is very similar. Just add to y direction parameters y.total and dy into the code:

# Set total number trail walkers  
walker.number <- 1000  
  
# Set total trail steps for each walker  
steps.total <-1000  
  
to.zero.n <- 0  
  
# random walk function  
for (i in 1 : walker.number){  
 x.total <- 0 # set initial position in x direction  
 y.total <- 0 # set initial position in y direction  
 dx <- 0  
 dy <- 0  
 distance <- 1  
 for (j in 1: steps.total){  
 if (distance == 0){  
 to.zero.n <- to.zero.n+1  
 break  
 } # when a walker come back to the origin, stop and count+1  
 else {  
 dx = floor(runif (1, min=1, max=3))  
 if (dx == 1){  
 x.total <- x.total+dx} # go right  
 else {x.total <- x.total+dx-3  
 } # go left  
 dy = floor(runif (1, min=1, max=3))  
 if (dy == 1){  
 y.total <- y.total+dy} # go up  
 else {y.total <- y.total+dy-3  
 } # go down  
 distance <- x.total^2+y.total^2  
 }  
 }  
}  
Prob <- to.zero.n/walker.number

I ran this code several times with different walker.number and steps.total. The results are listed below:

walker.number / steps.total / Probability

1000 / 1000 / 0.705

1000 / 10000 / 0.756

1000 / 100000 / 0.821

1000 / 1000000 / 0.899

100 / 1000000 / 0.90

10000 / 1000000 / 0.9051

The analytical solution of the probability for 2-d is 1. As shown in the above list, it indicates that, with the increase of step.total, the estimated probability of returning to the Origin is quickly getting closer to 1. On the other hand, for the same step.total, increasing walker.number contributes much less to the estimated probability.

In summary, the estimated probability of returning to the Origin in 2-d is between 0.9 to 1, depending on the selection of step.total and walker.total.

## 3-Dimensional

Similar to the 1-d and 2-d cases, just add to z direction parameters z.total and dz. The code for 3-d is:

# Set total number trail walkers  
walker.number <- 100  
  
# Set total trail steps for each walker  
steps.total <-100  
  
to.zero.n <- 0  
  
for (i in 1 : walker.number){  
 x.total <- 0  
 y.total <- 0  
 z.total <- 0  
 dx <- 0  
 dy <- 0  
 dz <- 0  
 distance <- 1  
 for (j in 1: steps.total){  
 if (distance == 0){  
 to.zero.n <- to.zero.n+1  
 break  
 }  
 else {  
 dx = floor(runif (1, min=1, max=3))  
 if (dx == 1){  
 x.total <- x.total+dx} # go forward  
 else {x.total <- x.total+dx-3  
 } # go backward  
 dy = floor(runif (1, min=1, max=3))  
 if (dy == 1){  
 y.total <- y.total+dy} # go right  
 else {y.total <- y.total+dy-3  
 } # go left  
 dz = floor(runif (1, min=1, max=3))  
 if (dz == 1){  
 z.total <- z.total+dz} # go up  
 else {z.total <- z.total+dz-3  
 } # go down  
 distance <- x.total^2+y.total^2+z.total^2  
 }  
 }  
}  
Prob <- to.zero.n/walker.number

I ran this code several times with different walker.number and steps.total. The results are listed below:

walker.number / steps.total / Probability

1000 / 1000 / 0.274

1000 / 10000 / 0.277

1000 / 100000 / 0.291

1000 / 1000000 / 0.311

100 / 1000000 / 0.30

The analytical solution of the probability for 3-d is ~0.34. As shown in the above list, it indicates that, with the increase of step.total, the estimated probability of returning to the Origin is getting closer to 0.34. On the other hand, for the same step.total, increasing walker.number contributes much less to the estimated probability.

In summary, the estimated probability of returning to the Origin in 3-d is ~0.3, depending on the selection of step.total and walker.total.

## Conclusions

1. The estimated probability of the particle that returns to the Origin is 1 for 1-d, 0.9-1 for 2-d, and ~0.3 for 3-d.
2. The estimated probability is influenced by the selection of walker.number and step.total. The step.total seems to have a larger effect on the accuracy of probability estimation, compared to walker.number.