ISING - MCMC - Script

Paola Serra and Marzio De Corato

The Monte-Carlo method here implemented was adapted from the pseudo-code provided in the textbook Frenkel, D., & Smit, B. (2001). Understanding molecular simulation: from algorithms to applications. For a better performance the user should consider to implement in C++ or in Fortran (this was the Frenkel and Smit idea).

Loading of the libraries

```
library(ggplot2)
library(plot.matrix)
```

```
## Warning: package 'plot.matrix' was built under R version 4.0.5
```

```
library(ggplotify)
```

```
## Warning: package 'ggplotify' was built under R version 4.0.5
```

```
library(grid)
library(dppmix)
```

```
## Warning: package 'dppmix' was built under R version 4.0.5
```

```
library(SpatEntropy)
```

```
## Warning: package 'SpatEntropy' was built under R version 4.0.5

## Loading required package: spatstat

## Warning: package 'spatstat' was built under R version 4.0.5

## Loading required package: spatstat.data

## Warning: package 'spatstat.data' was built under R version 4.0.5

## Loading required package: spatstat.geom

## Warning: package 'spatstat.geom' was built under R version 4.0.5

## spatstat.geom 2.3-1

## Loading required package: spatstat.core

## Warning: package 'spatstat.core' was built under R version 4.0.5

## Loading required package: nlme

## Warning: package 'nlme' was built under R version 4.0.5

## Loading required package: rpart

## spatstat.core 2.3-2

## Loading required package: spatstat.linnet

## Warning: package 'spatstat.linnet' was built under R version 4.0.5
## spatstat.linnet 2.3-1
```

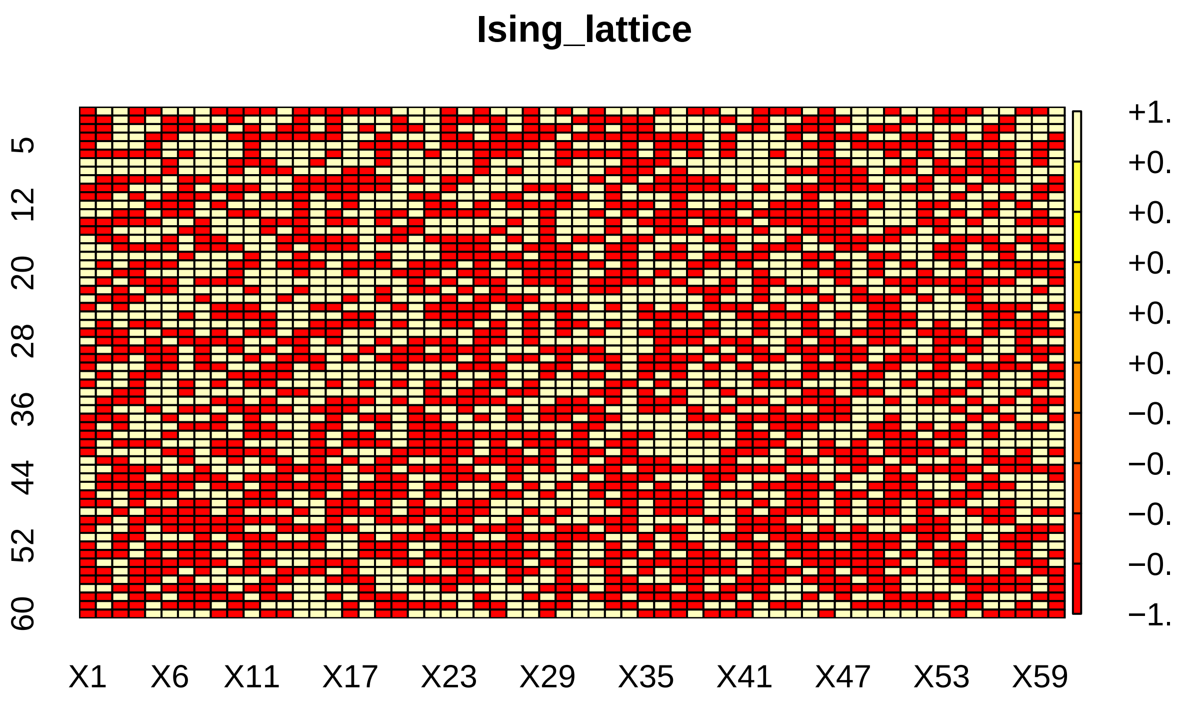
```
##
## spatstat 2.3-0       (nickname: 'That's not important right now')
## For an introduction to spatstat, type 'beginner'
```

Variables intialization

```
ffn<-as.double(0)
ffn<-as.double(0)
sfn<-as.double(0)
tfn<-as.double(0)
J <- as.double(1) #SET HERE THE COUPLING CONSTANT
m <- as.double(0)

energy <- as.double(0)
energy_new <- as.double(0)
dimension <-as.integer(60) # SET HERE THE LATTICE DIMENSION
energy_history<-data.frame(matrix(0, ncol = 6, nrow = 20000))
colnames(energy_history)<-c("Step","Energy","Magnetization","Mag_sd","T-val","S_Entropy")


lattice <- data.frame(replicate(dimension,replicate(dimension,0)))
lattice_save <- data.frame(replicate(dimension,replicate(dimension,0)))
#lattice[1:dimension,1:dimension] <- data.frame(replicate(dimension,sample(c(-1,1),dimension,rep=TRUE))
lattice[1:dimension,1:dimension] <- data.frame(matrix(2*rbinom(dimension*dimension,1,1/2)-1,ncol = dimen
correlation_matrix_start <- array(dim=c(dimension ,dimension ,dimension,dimension ))
correlation_matrix <- array(dim=c(dimension ,dimension ,dimension,dimension ))
mean_correlation_matrix <- array(dim=c(dimension ,dimension ))
lattice_save <- data.frame(replicate(dimension,replicate(dimension,0)))
```

Plot the starting configuration

```
Ising_lattice<-data.matrix(lattice)
plot(Ising_lattice,breaks=c(-1,1),xaxt = "n",ylab='',xlab='',tick = FALSE)
```



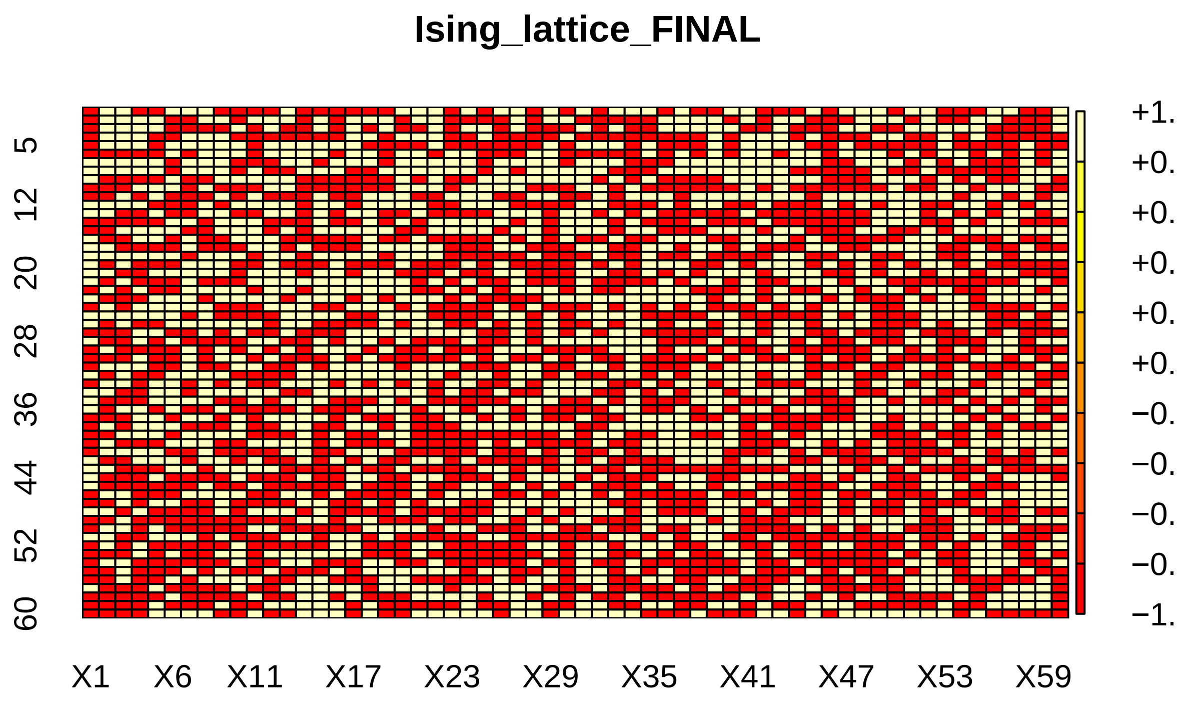
The core of the script

```
for (iter in 1:10) { #Set here the number of MC cycles, note that larger lattice will require larger cy
                   # -------->CAUTION<----------- Just for this example we set 10, but the user should
  m <- 0
  energy <- 0
  energy_new <- 0

  lattice_save<-lattice # The old lattice is saved

for (i in 1:dimension) {  #Energy evaluation before the MC move
  for (j in 1:dimension) {
    i_right= ((i) %% dimension)+1   #Note that the PBC are applied in order to delete the borders
    i_left= ((i-2) %% dimension)+1
    j_down= ((j-2) %% dimension)+1
    j_up= ((j) %% dimension)+1

    if ( lattice[i,j]==lattice[i_right,j]*J){
      energy = energy -1
    }
    if ( lattice[i,j]==lattice[i_left,j]*J){
      energy = energy -1
    }
    if ( lattice[i,j]==lattice[i,j_up]*J){
      energy = energy -1
    }
    if ( lattice[i,j]==lattice[i,j_down]*J){
      energy = energy -1
    }

  }

}

  p<-shannon(as.matrix(lattice)) #The important values are saved in to a dataframe
  energy_history[iter,"Energy"]=energy/(dimension*dimension)
  energy_history[iter,"Step"]=iter
  energy_history[iter,"Magnetization"]=mean(as.matrix(lattice[1:dimension,1:dimension]))
  energy_history[iter,"Mag_sd"]=sd(as.matrix(lattice[1:dimension,1:dimension]))
  energy_history[iter,"T-val"]=energy_history[iter,"Magnetization"]/energy_history[iter,"Mag_sd"]
  energy_history[iter,"S_Entropy"]=p$rel.shann

print(c(iter,energy_history[iter,"Energy"],energy_history[iter,"Magnetization"],energy_history[iter,"T-v

lattice_save<-lattice



  x <- sample(1:dimension , 1)  #MC move
  y <- sample(1:dimension , 1)



  if ( lattice[x,y] > 0){
    lattice[x,y]=-1
    } else  {
    lattice[x,y]=+1
    }





#  print(x)
#  print(y)

  for (k in 1:3) {  #Use this portion of code to put the diffusion centers
    for (j in 1:3) {
      lattice[k+1,j+1]=+1
    }
  }

  for (k in 1:3) {
    for (j in 1:3) {
      lattice[k+dimension-4,j+1]=-1
    }
  }

  for (k in 1:3) {
    for (j in 1:3) {
      lattice[k+1,j+dimension-4]=-1
    }
  }

  for (k in 1:3) {
    for (j in 1:3) {
      lattice[k+dimension-4,j+dimension-4]=+1
    }
  }




  #   lattice[dimension-k-3,dimension-j-3]=-2


  for (i in 1:dimension) {    #Energy calculation after the MC move
    for (j in 1:dimension) {
      i_right= ((i) %% dimension)+1
      i_left= ((i-2) %% dimension)+1
      j_down= ((j-2) %% dimension)+1
      j_up= ((j) %% dimension)+1


      if ( lattice[i,j]==lattice[i_right,j]*J){
        energy_new = energy_new -1
      }
      if ( lattice[i,j]==lattice[i_left,j]*J){
        energy_new = energy_new -1
      }
      if ( lattice[i,j]==lattice[i,j_up]*J){
        energy_new = energy_new -1
      }
      if ( lattice[i,j]==lattice[i,j_down]*J){
        energy_new = energy_new -1
      }

    }

  }



  if (energy_new < energy ){    #Metropolis code: if the new energy is lower accept it, otherwise exctra
   # print("Accepted")
  }

 if (energy_new  >= energy ){
    delta<-as.double(0)
    q<-as.double(0)
    p<-sample(1:10,1)/10
    delta<-abs(energy_new-energy)
    #print(delta)
    q<-exp(-delta/0.0000000001)
    #print(c(p,q))
    if(p > q){

   lattice<-lattice_save   #Restore the original lattice if the move is rejected
   }
  }

  Ising_lattice_FINAL<-data.matrix(lattice)







}
```

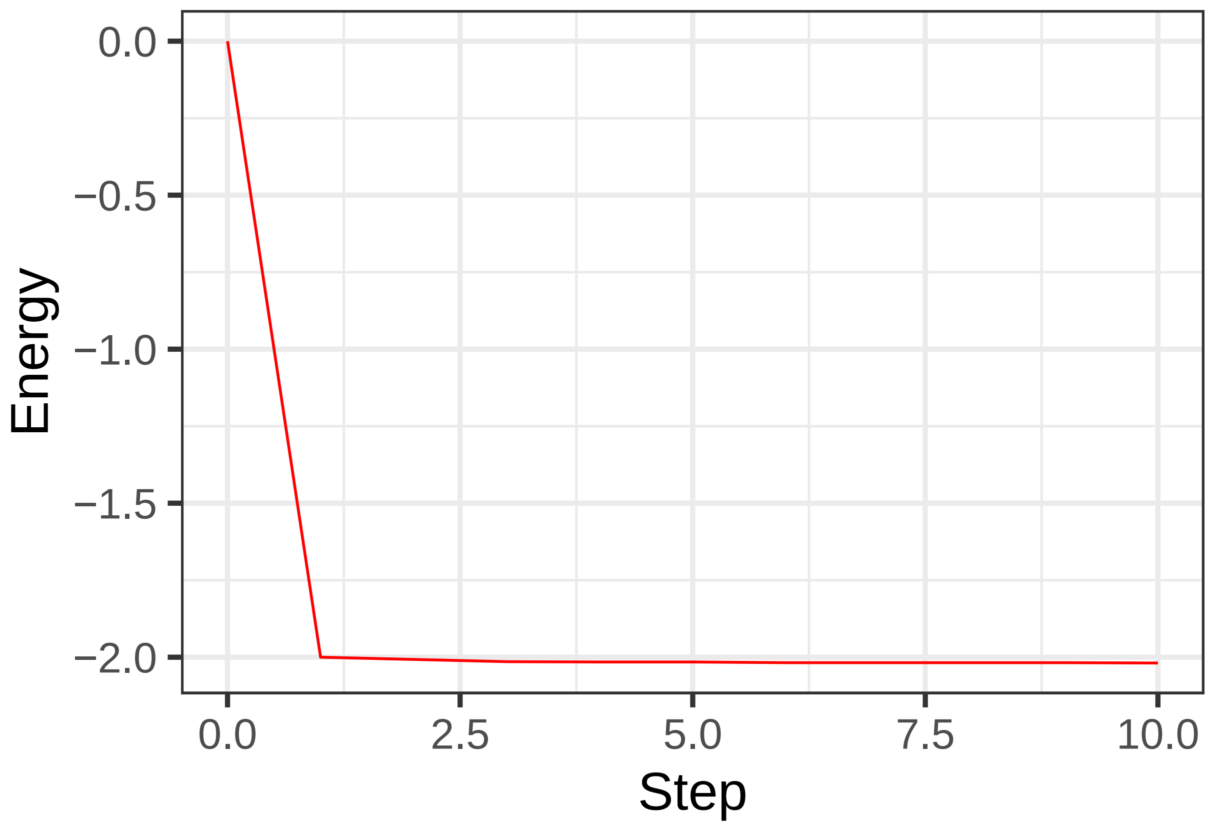
```
## [1]  1.00000000 -2.00000000 -0.02666667 -0.02667245  0.99948698
## [1]  2.00000000 -2.01444444 -0.02722222 -0.02722853  0.99946538
## [1]  3.00000000 -2.01444444 -0.02722222 -0.02722853  0.99946538
## [1]  4.00000000 -2.01555556 -0.02666667 -0.02667245  0.99948698
## [1]  5.00000000 -2.01555556 -0.02666667 -0.02667245  0.99948698
## [1]  6.00000000 -2.01777778 -0.02611111 -0.02611639  0.99950814
## [1]  7.00000000 -2.01777778 -0.02611111 -0.02611639  0.99950814
## [1]  8.00000000 -2.01777778 -0.02611111 -0.02611639  0.99950814
## [1]  9.00000000 -2.01777778 -0.02666667 -0.02667245  0.99948698
## [1] 10.00000000 -2.01888889 -0.02611111 -0.02611639  0.99950814
```

Plot the evolution of the important quantities (such as energy, mean magnetization, shannon entropy…)

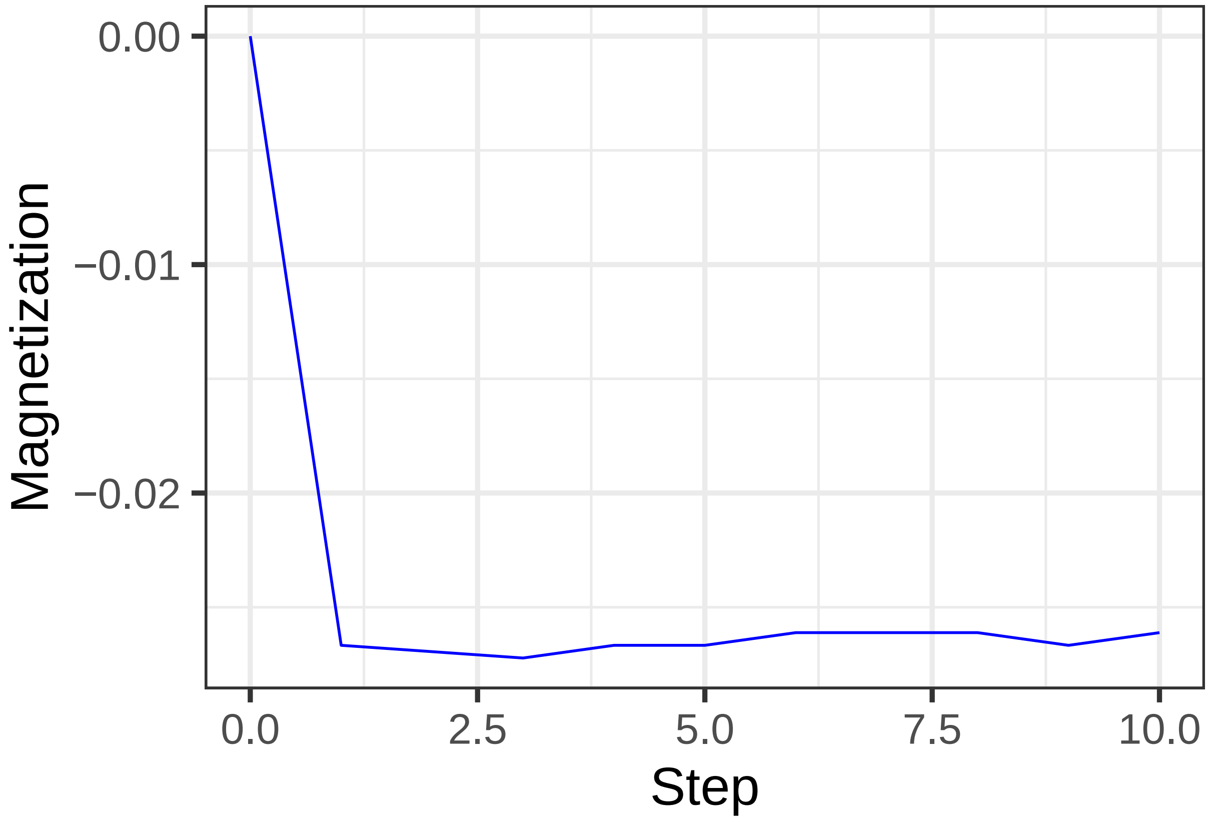
```
Ising_lattice_FINAL<-data.matrix(lattice)

plot(Ising_lattice_FINAL,breaks=c(-1,1),xaxt = "n",ylab='',xlab='',tick = FALSE)
```



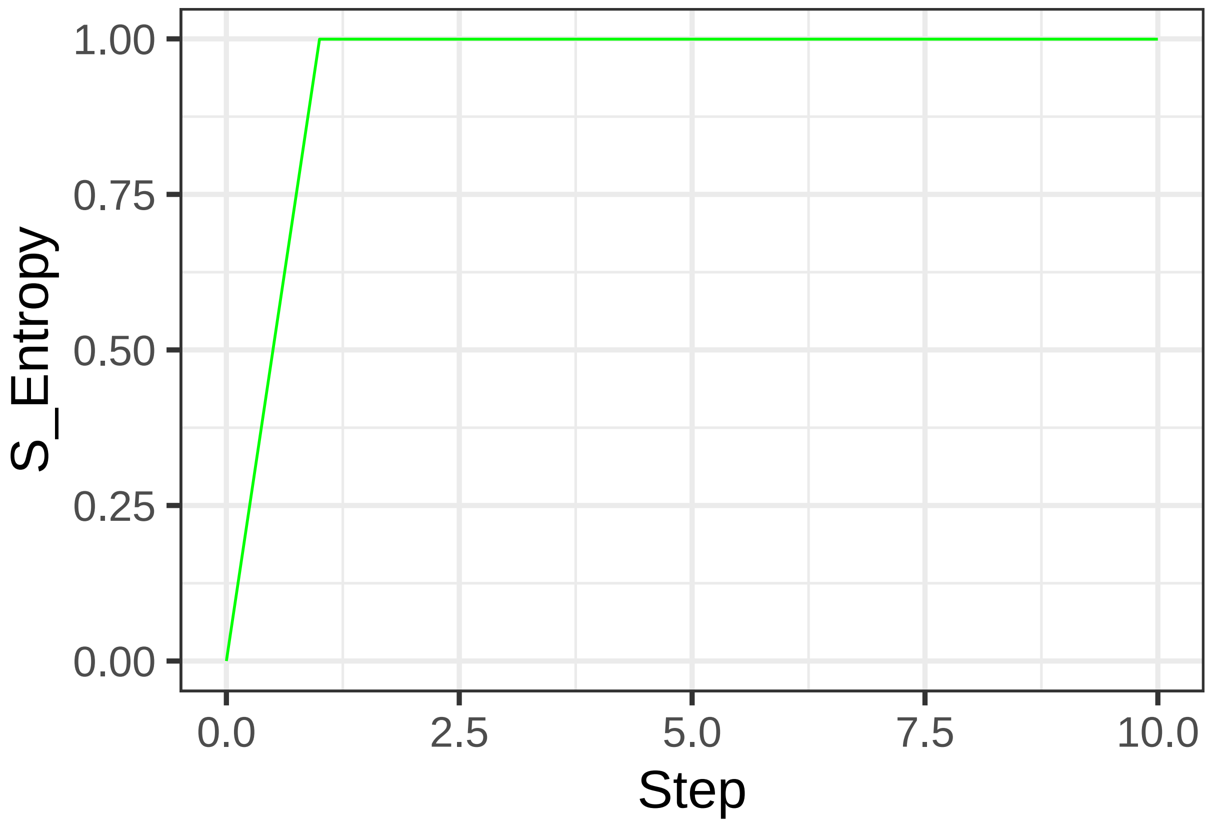
```r
#plot(energy_history)
energy_history<-energy_history[-2,]
ggplot(data= energy_history,mapping = aes(x = Step, y = Energy))+geom_line(colour="red")+theme_bw(base_
```



```
ggplot(data= energy_history,mapping = aes(x = Step, y = Magnetization))+geom_line(colour="blue")+theme_
```



```
ggplot(data= energy_history,mapping = aes(x = Step, y = S_Entropy))+geom_line(colour="green")+theme_bw(
```



Calculation and plot of the correlation function uf the 4-near-neighbor

```
ffn<-as.double(0)
sfn<-as.double(0)
tfn<-as.double(0)
fofn<-as.double(0)



for (i in 1:dimension) {
  for (j in 1:dimension) {



    i_right= ((i) %% dimension)+1
    i_left= ((i-2) %% dimension)+1
    j_down= ((j-2) %% dimension)+1
    j_up= ((j) %% dimension)+1


i_rright= ((i+1) %% dimension)+1
i_lleft= ((i-3) %% dimension)+1
j_ddown=  ((j-3) %% dimension)+1
j_uup= ((j+1) %% dimension)+1

i_rrright= ((i+2) %% dimension)+1
i_llleft= ((i-4) %% dimension)+1
```

```
j_dddown= ((j-4) %% dimension)+1
j_uuup= ((j+2) %% dimension)+1

i_rrrright= ((i+3) %% dimension)+1
i_lllleft= ((i-5) %% dimension)+1
j_ddddown= ((j-5) %% dimension)+1
j_uuuup= ((j+3) %% dimension)+1


ffn<-ffn+(lattice[i,j]*lattice[i_right,j]+lattice[i,j]*lattice[i_left,j]+lattice[i,j]*lattice[i,j_up]+la

sfn<-sfn+(lattice[i,j]*lattice[i_rright,j]+lattice[i,j]*lattice[i_lleft,j]+lattice[i,j]*lattice[i,j_ddow
         +lattice[i,j]*lattice[i_right,j_down] + lattice[i,j]*lattice[i_right,j_up]+lattice[i,j]*lattice

tfn<-tfn+(lattice[i,j]*lattice[i_rrright,j]+lattice[i,j]*lattice[i_llleft,j]+lattice[i,j]*lattice[i,j_dd

fofn<-fofn+(lattice[i,j]*lattice[i_rrrright,j]+lattice[i,j]*lattice[i_lllleft,j]+lattice[i,j]*lattice[i
          +lattice[i,j]*lattice[i_right,j_uuup]+lattice[i,j]*lattice[i_rright,j_uup]+lattice[i,j]*lattic
          +lattice[i,j]*lattice[i_rrright,j_down]+lattice[i,j]*lattice[i_rright,j_ddown]+lattice[i,j]*la
          +lattice[i,j]*lattice[i_llleft,j_down]+lattice[i,j]*lattice[i_lleft,j_ddown]+lattice[i,j]*lat
          +lattice[i,j]*lattice[i_llleft,j_up]+lattice[i,j]*lattice[i_lleft,j_uup]+lattice[i,j]*lattice
          )/16



  }
}
ffn
```

```
## [1] 36
```

```
sfn
```

```
## [1] 44
```

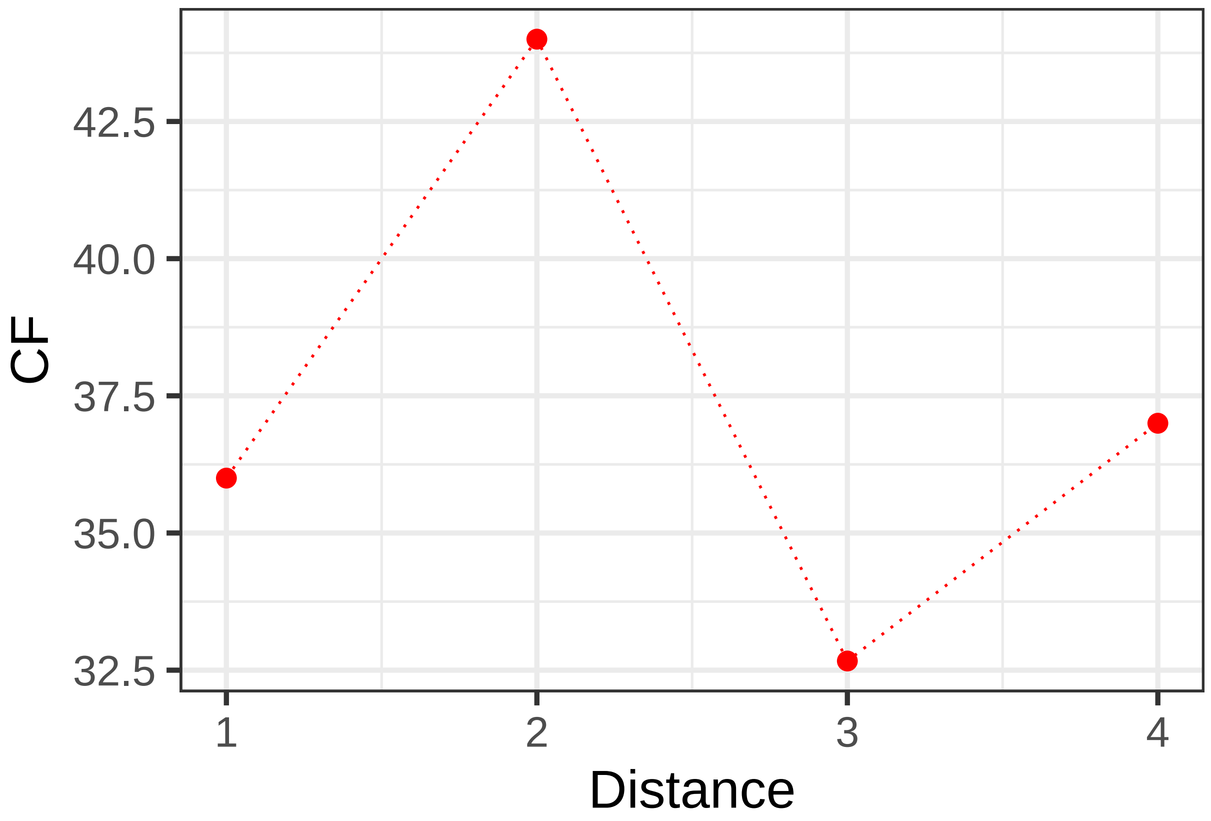
```
tfn
```

```
## [1] 32.66667
```

```
fofn
```

```
## [1] 37
```

```
order_parameter<-data.frame("Distance"=c(1,2,3,4),"CF"=c(ffn,sfn,tfn,fofn))
#fit<-lm(log(CF) ~ Distance, data= order_parameter)
#summary(fit)
ggplot(data= order_parameter,mapping = aes(x = Distance, y = CF))+geom_line(linetype="dotted",colour="re
```



Save the results

```
save(Ising_lattice_FINAL,file="J-1_60_2500_T=16_lattice.rda")
save(energy_history,file="J-1_60_2500_T=16_energy.rda")
```