011037 : TP4

BLANCHARD Simon BÔNE Constantin CHEN Zeyu

14 Mars 2019

Notes

- BLANCHARD Simon : 3; 3
- BÔNE Constantin : 3; 3
- CHEN Zeyu : 3; 3

```
library(ggplot2)
```

```
## Warning: package 'ggplot2' was built under R version 3.4.4
```

```
library(gridExtra)
```

Exercice 1 : Simulation du modèle d'Ising

Question 1

La fonction suivante permet de générer une configuration initiale S correspondant à un maillage de taille $N \times N$ avec une proportion p de spins d'état -1 et une proportion $1 - p$ de spins d'état 1 :

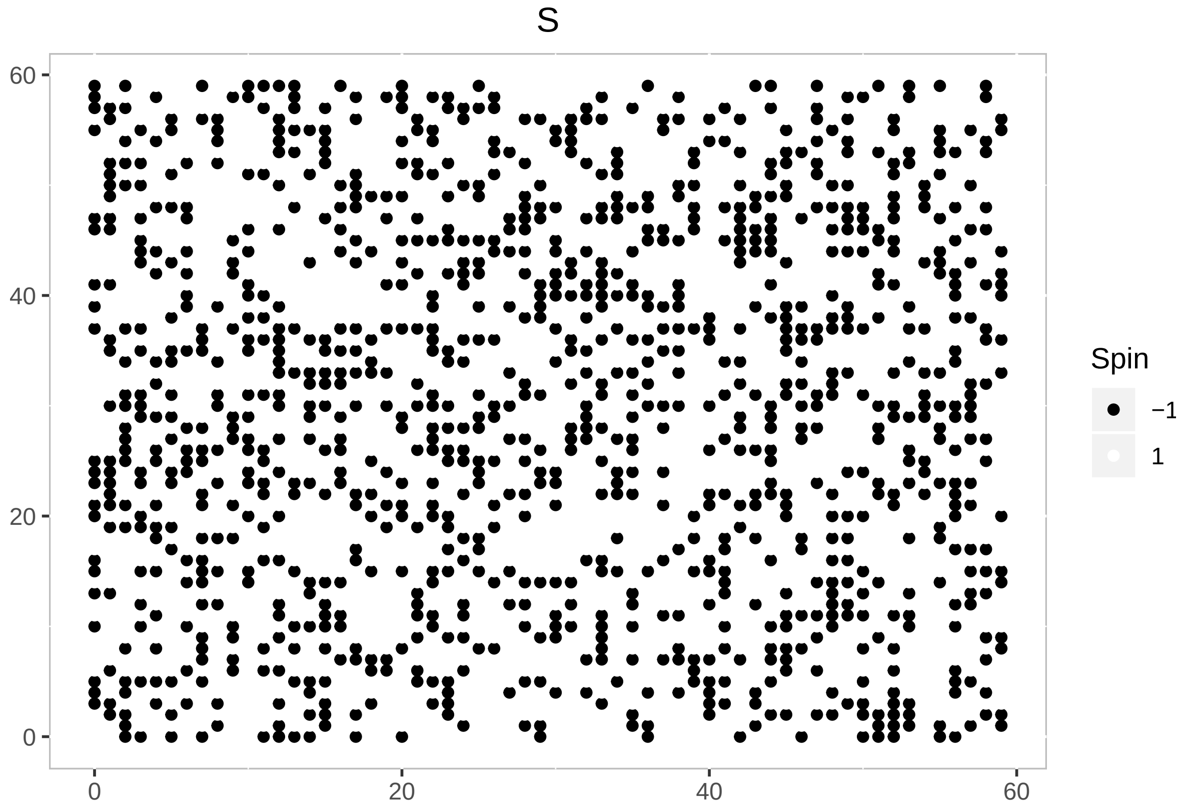
```
configuration_initiale_S <- function(N, p)
{
  I <- c()
  for(i in 0:(N - 1))
  {
    I <- c(I, rep(i, N))

  }
  J <- rep(0:(N - 1), N)
  S <- c()
  for(i in 1:(N * N))
  {
    u <- runif(1, 0, 1)
    S <- c(S, ifelse(u < p, -1, 1))
  }
  return(data.frame(I = I, J = J, S = S))
}
```

On génère alors une configuration initiale S :

```
S <- configuration_initiale_S(60, 0.3)
```

On affiche cette configuration initiale S :

```
S_plot <- ggplot(data = S, aes(x = I, y = J, color = factor(S), shape = factor(S))) + geom_point() + la
S_plot
```



La fonction suivante génère une réalisation d'une loi uniforme sur $\{0, ..., N-1\}$:

```
realisation_loi_uniforme_gamma <- function(N)
{
  u <- runif(1, 0, 1)
  p <- 1 / N
  i <- 0
  while(u > ((i + 1) * p) && i < (N - 1)) i <- i + 1
  return(ifelse(i == (N - 1), N - 1, i))
}
```

La fonction suivante calcule $\Delta H(S, S_{(x,y)})$ pour une configuration initiale S correspondant à un maillage de taille $N \times N$:

```
delta_H <- function(S, x, y, N)
{
  s <- 0

  # en (0, 0)
  if(x == 0 && y == 0) s <- S$S[2] + S$S[N + 1]

  # en (N-1, N-1)
  else if(x == (N - 1) && y == (N - 1)) s <- S$S[N * (N - 1)] + S$S[N * (N - 1) + N - 1]

  # en (0, i) pour i dans 1,...,N-2
  else if(x == 0 && y >= 1 && y <= (N - 2)) s <- S$S[y] + S$S[y + 2] + S$S[N + y + 1]

  # en (0, N-1)
  else if(x == 0 && y == (N - 1)) s <- S$S[y] + S$S[N + y + 1]
```

```
  # en (i, 0) pour i dans 1,...,N-2
  else if(x >= 1 && x <= (N - 2) && y == 0) s <- S$S[1] + S$S[(x + 1) * N + 1] + S$S[x * N + 2]

  # en (N-1, 0)
  else if(x == (N - 1) && y == 0) s <- S$S[(N - 2) * N + 1] + S$S[N * (N - 1) + 2]

  # en (i, N-1) pour i dans 1,...,N-2
  else if(x >= 1 && x <= (N - 2) && y == (N - 1)) s <- S$S[N * x] + S$S[N * (x + 2)] + S$S[N * (x + 1) 

  # en (N-1, i) pour i dans 1,...,N-2
  else if(x == (N - 1) && y >= 1 && y <= (N - 2)) s <- S$S[N * (N - 1) + y] + S$S[N * (N - 1) + y + 2] 

  # en (i, j) pour i,j dans 1,...,N-2
  else s <- S$S[N * x + y] + S$S[N * x + y + 2] + S$S[N * (x - 1) + y + 1] + S$S[N * (x + 1) + y + 1]

  return(2 * s * S$S[N * x + y + 1])
}
```

La fonction suivante implémente l'algorithme de Hastings-Metropolis pour une configuration initiale S correspondant à un maillage de taille $N \times N$, $\beta = B$ et M itérations :

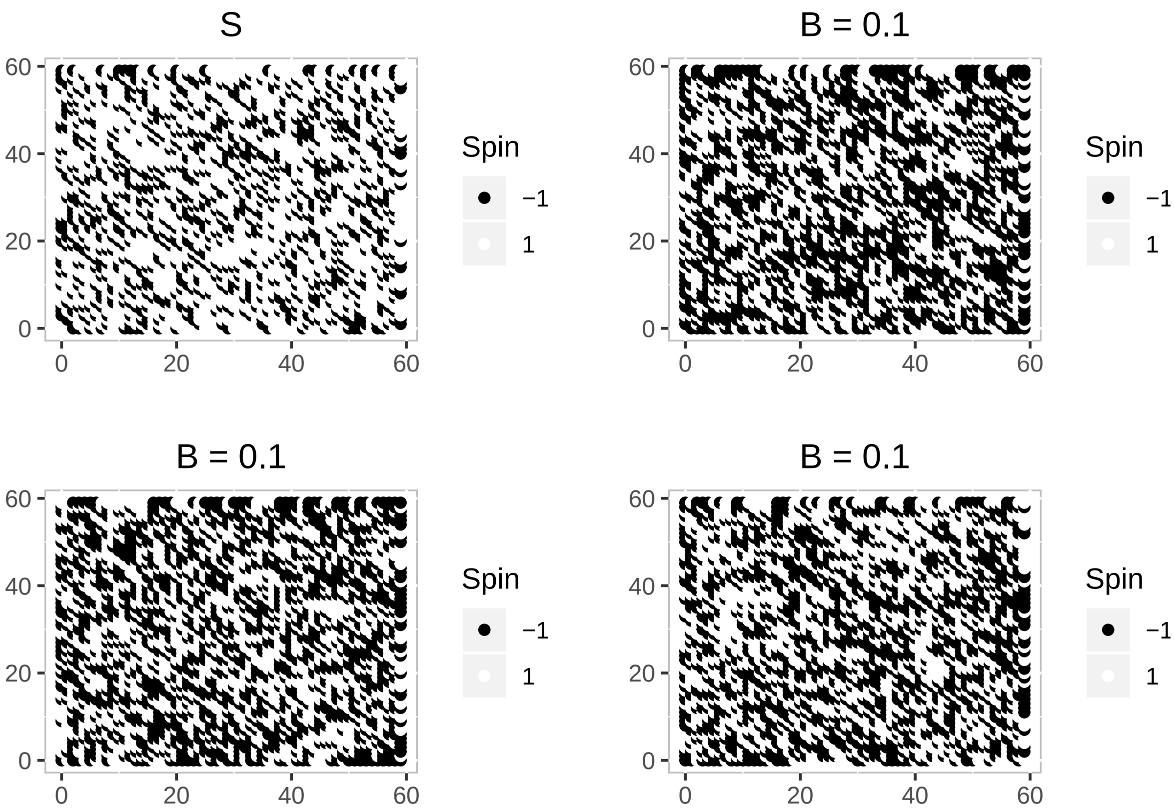
```
hastings_metropolis <- function(S, B, N, M)
{
  for(i in 1:M)
  {
    x <- realisation_loi_uniforme_gamma(N)
    y <- realisation_loi_uniforme_gamma(N)
    u <- runif(1, 0, 1)
    delta <- delta_H(S, x, y, N)
    S$S[N * x + y + 1] <- ifelse(u <= exp(-B * delta), - S$S[N * x + y + 1], S$S[N * x + y + 1])
  }
  return(S)
}
```

On génère alors trois modèles d'Ising de configuration initiale S avec $\beta = 0.1$ et $N = 60$:

```
Ising_01_1 <- ggplot(data = hastings_metropolis(S, 0.1, 60, 100000), aes(x = I, y = J, color = factor(S)
Ising_01_2 <- ggplot(data = hastings_metropolis(S, 0.1, 60, 100000), aes(x = I, y = J, color = factor(S)
Ising_01_3 <- ggplot(data = hastings_metropolis(S, 0.1, 60, 100000), aes(x = I, y = J, color = factor(S)
```

On affiche ces trois modèles d'Ising ainsi que la configuration initiale S :

```
grid.arrange(S_plot, Ising_01_1, Ising_01_2, Ising_01_3)
```



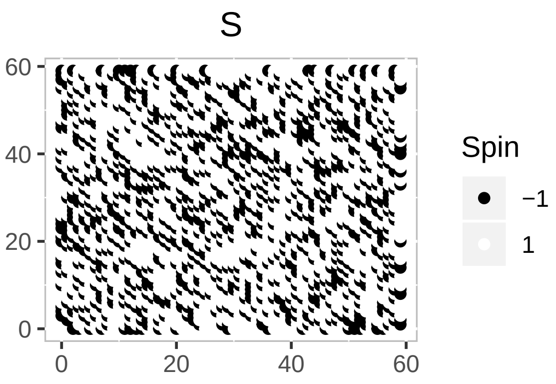
Question 2

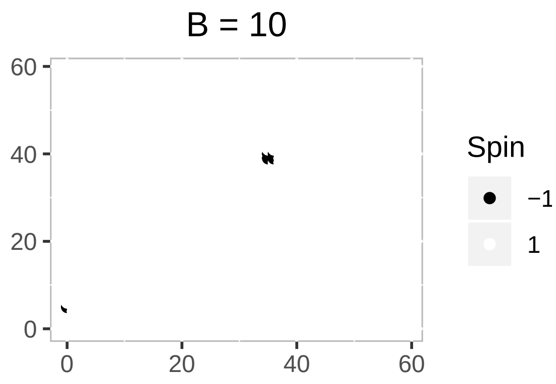
On génère trois modèles d'Ising de configuration initiale S avec $\beta = 10$ et $N = 60$:

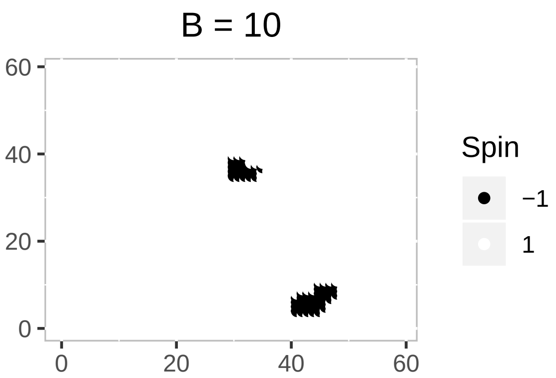
```
Ising_10_1 <- ggplot(data = hastings_metropolis(S, 10, 60, 100000), aes(x = I, y = J, color = factor(S)
Ising_10_2 <- ggplot(data = hastings_metropolis(S, 10, 60, 100000), aes(x = I, y = J, color = factor(S)
Ising_10_3 <- ggplot(data = hastings_metropolis(S, 10, 60, 100000), aes(x = I, y = J, color = factor(S)
```

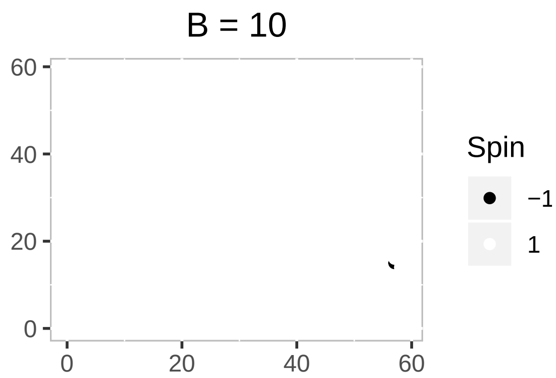
On affiche ces trois modèles d'Ising ainsi que la configuration initiale S :

```
grid.arrange(S_plot, Ising_10_1, Ising_10_2, Ising_10_3)
```









Question 3

On a :

$$\beta = \frac{1}{T}$$

On constate alors bien que lorsque β est faible (donc T élevée), les fluctuations thermiques dominent rendant le système désordonné. Au contraire, lorsque β est élevé (donc T faible), le système privilégie les configurations de basse énergie tendant à aligner les spins.

Question 4

On génère une nouvelle configuration initiale S :

```
S <- configuration_initiale_S(60, 0.7)
```

On affiche cette configuration initiale $S2$:

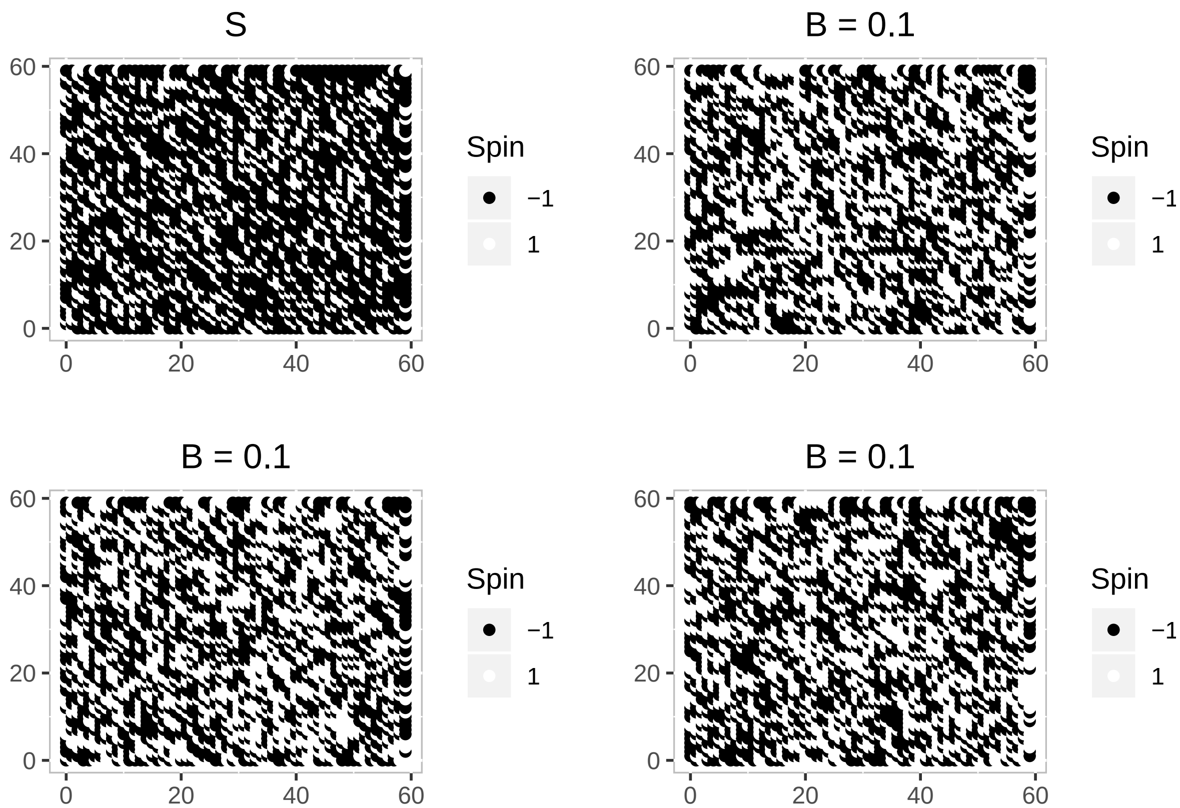
```
S_plot <- ggplot(data = S, aes(x = I, y = J, color = factor(S), shape = factor(S))) + geom_point() + la
S_plot
```

On génère alors trois nouveaux modèles d'Ising de configuration initiale S avec $\beta = 0.1$ et $N = 60$:

```
Ising_01_1 <- ggplot(data = hastings_metropolis(S, 0.1, 60, 100000), aes(x = I, y = J, color = factor(S
Ising_01_2 <- ggplot(data = hastings_metropolis(S, 0.1, 60, 100000), aes(x = I, y = J, color = factor(S
Ising_01_3 <- ggplot(data = hastings_metropolis(S, 0.1, 60, 100000), aes(x = I, y = J, color = factor(S
```

On affiche ces trois modèles d'Ising ainsi que la configuration initiale S :

```
grid.arrange(S_plot, Ising_01_1, Ising_01_2, Ising_01_3)
```

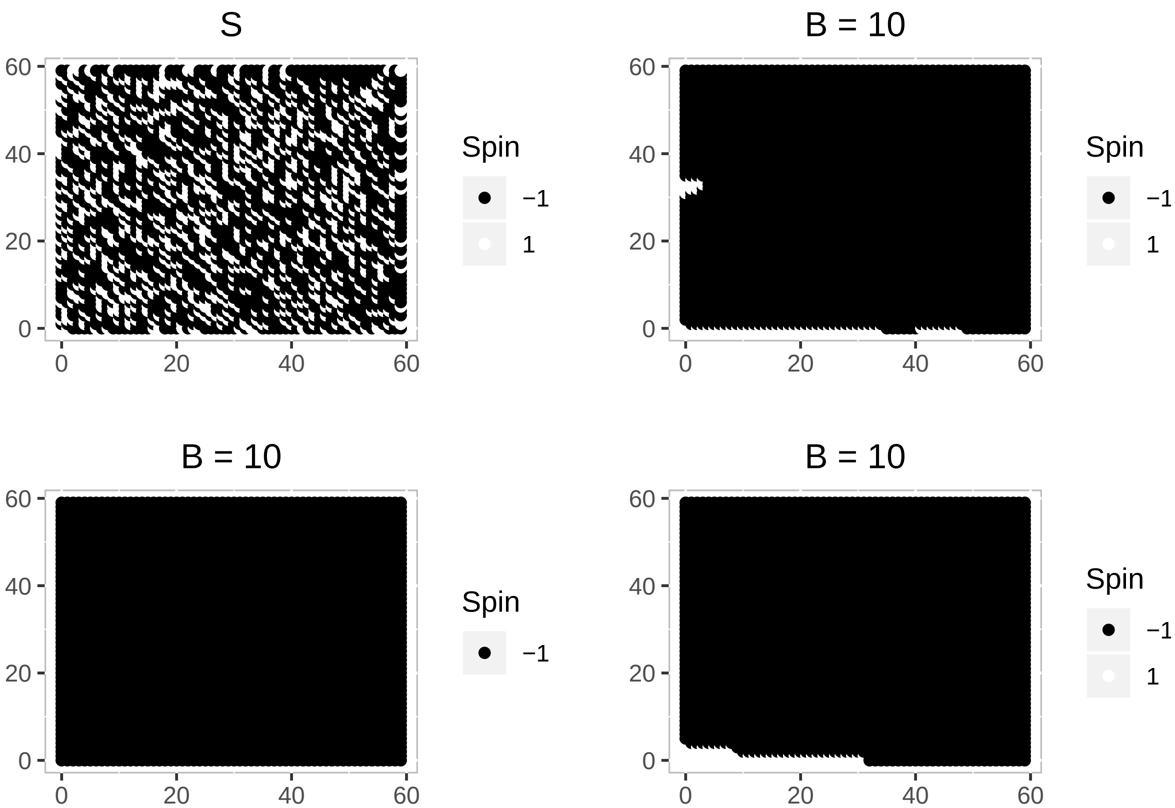


On génère également trois nouveaux modèles d'Ising de configuration initiale S avec $\beta = 10$ et $N = 60$:

```
Ising_10_1 <- ggplot(data = hastings_metropolis(S, 10, 60, 100000), aes(x = I, y = J, color = factor(S)
Ising_10_2 <- ggplot(data = hastings_metropolis(S, 10, 60, 100000), aes(x = I, y = J, color = factor(S)
Ising_10_3 <- ggplot(data = hastings_metropolis(S, 10, 60, 100000), aes(x = I, y = J, color = factor(S)
```

On affiche ces trois modèles d'Ising ainsi que la configuration initiale S :

```
grid.arrange(S_plot, Ising_10_1, Ising_10_2, Ising_10_3)
```



On constate également bien avec ces deux nouvelles séries de graphiques que lorsque β est faible (donc T élevée), les fluctuations thermiques dominent rendant le système désordonné. Au contraire, lorsque β est élevé (donc T faible), le système privilégie les configurations de basse énergie tendant à aligner les spins.

Exercice 2 : Voyageur de commerce

```r
set.seed(561)
cor <- read.csv("Coordonnees-Villes.csv",header = TRUE)
cor <- cbind(cor[,2],cor[,3])

numberVille <- dim(cor)[1]

#solution initiale
#s <- sample(numberVille)
s <- seq(1,50,1)

#The function to calculate the distance of one feasible solution
dist <- function(sol=s)
{
  distance = 0
  for(i in 1:(numberVille-1))
  {
    d <- sqrt((cor[sol[i],1]-cor[sol[i+1],1])^2+(cor[sol[i],2]-cor[sol[i+1],2])^2)
    distance<- distance + d
  }
  distance<- distance+sqrt((cor[sol[1],1]-cor[sol[numberVille],1])^2+(cor[sol[1],2]-cor[sol[numberVille
```

```r
  return (distance)
}

#Generate a neibor
neibor<-function(sol)
{
  n <- length(sol)
  x1 <- round(runif(1)*(n-1)+1)
  x2 <- round(runif(1)*(n-1)+1)
  while(x1==x2)
  {
  x1 <- round(runif(1)*(n-1)+1)
  x2 <- round(runif(1)*(n-1)+1)
  }
  temp <- sol[x1]
  sol[x1] <- sol[x2]
  sol[x2] <- temp
  return (sol)
}

#Applied the Simulated Annealing algothme
SA <- function(sol)
{
  i =1
  solution <- sol
  while(i<500)
  {
  mauvais=0
  j = 1
  b <- log(i*j+1)/8.3
  while(j<500)
  {
  u <- runif(1)
  oldSol <- sol
  oldDistance <- dist(oldSol)
  newSol <- neibor(oldSol)
  newDistance <- dist(newSol)
  if(u < exp(-b*(newDistance-oldDistance)))
    {
      sol <- newSol
      mauvais=0
    }else{
      mauvais=mauvais+1
      sol <- oldSol
    }
  j = j+1
  b <-log(i*j+1)/8.3
  if(mauvais>200)
    break;
  }
  if(dist(sol)<dist(solution))
      solution <- sol
  i=i+1
```

```
  }
  return(solution)
}
sol = SA(s)
```

Voila notre solution et distance de la solution

```
sol
```

```
##  [1] 17 16 42 48  5 33 44  6 46  7 41 24 28 11  8 20 32 19 18  1 10 22 47
## [24] 12 23 40 21  3 45 36 31 27 38 30 37 15 26 43 35 29 13 14  4 34 39  2
## [47]  9 49 50 25
```

```
dist(sol)
```

```
## [1] 247.091
```

on plot pour notre solution ,et on constate qu'il n'y aucun croisement

```
plot(cor[sol,1],cor[sol,2],main = "Longeur_Chemin=247.091")
lines(cor[sol,1],cor[sol,2],col="red")
text(cor[sol,1],cor[sol,2],labels= sol,cex=0.5,col="blue",pos=1)
```