# AMATO_CHEVAUX_PROJET_2

December 17, 2018

# 1 Projet 2 - Simulations Et Copules

## 1.1 Alexandre CHEVAUX

## 1.2 Virgile AMATO

# 2 Probleme de la Carte aux 4 couleurs (et en nuances de gris)

## 2.1 Initialisation de la carte

In [555]:
```
## Nombre de departements
N = 95

V = array(0,dim=c(N,N))

V[1,39]=1; V[1,74]=1; V[1,73]=1; V[1,38]=1; V[1,69]=1; V[1,71]=1;
V[2,8]=1; V[2,51]=1; V[2,77]=1; V[2,60]=1; V[2,80]=1; V[2,59]=1;
V[3,71]=1; V[3,42]=1; V[3,63]=1; V[3,23]=1; V[3,18]=1; V[3,58]=1;
V[4,5]=1; V[4,6]=1; V[4,83]=1; V[4,84]=1; V[4,26]=1;
V[5,26]=1; V[5,38]=1; V[5,73]=1;
V[6,83]=1;
V[7,26]=1; V[7,30]=1; V[7,38]=1; V[7,43]=1; V[7,48]=1;
V[7,42]=1; V[7,84]=1;
V[8,51]=1; V[8,55]=1;
V[9,11]=1; V[9,66]=1; V[9,31]=1;
V[10,21]=1; V[10,89]=1; V[10,77]=1; V[10,51]=1; V[10,52]=1;
V[11,66]=1; V[11,31]=1; V[11,81]=1; V[11,34]=1;
V[12,15]=1; V[12,48]=1; V[12,30]=1; V[12,34]=1; V[12,81]=1;
V[12,82]=1; V[12,46]=1;
V[13,83]=1; V[13,84]=1; V[13,30]=1;
V[14,50]=1; V[14,61]=1; V[14,27]=1;
V[15,19]=1; V[15,63]=1; V[15,43]=1; V[15,48]=1; V[15,46]=1;
V[16,17]=1; V[16,79]=1; V[16,86]=1; V[16,87]=1; V[16,24]=1;
V[17,85]=1; V[17,79]=1; V[17,24]=1; V[17,33]=1;
V[18,23]=1; V[18,36]=1; V[18,41]=1; V[18,45]=1; V[18,58]=1;
V[19,24]=1; V[19,87]=1; V[19,23]=1; V[19,63]=1; V[19,46]=1;
V[21,52]=1; V[21,70]=1; V[21,39]=1; V[21,71]=1; V[21,58]=1;
V[21,89]=1;
```

```
V[22,29]=1; V[22,56]=1; V[22,35]=1;
V[23,36]=1; V[23,63]=1; V[23,87]=1;
V[24,33]=1; V[24,87]=1; V[24,46]=1; V[24,47]=1;
V[25,39]=1; V[25,70]=1; V[25,90]=1; V[25,68]=1;
V[26,38]=1; V[26,84]=1;
V[27,28]=1; V[27,61]=1; V[27,76]=1; V[27,60]=1; V[27,95]=1;
V[27,78]=1;
V[28,61]=1; V[28,41]=1; V[28,45]=1; V[28,91]=1; V[28,78]=1;
V[28,72]=1;
V[29,56]=1;
V[30,34]=1; V[30,48]=1; V[30,84]=1;
V[31,32]=1; V[31,82]=1; V[31,81]=1; V[31,65]=1;
V[32,40]=1; V[32,47]=1; V[32,82]=1; V[32,65]=1; V[32,64]=1;
V[33,40]=1; V[33,47]=1;
V[34,81]=1;
V[35,53]=1; V[35,50]=1; V[35,44]=1; V[35,56]=1;
V[36,37]=1; V[36,41]=1; V[36,87]=1; V[36,86]=1;
V[37,41]=1; V[37,49]=1; V[37,72]=1; V[37,86]=1;
V[38,42]=1; V[38,69]=1; V[38,73]=1;
V[39,71]=1; V[39,70]=1;
V[40,47]=1; V[40,64]=1;
V[41,45]=1; V[41,72]=1;
V[42,63]=1; V[42,43]=1; V[42,63]=1; V[42,69]=1; V[42,71]=1;
V[43,63]=1; V[43,48]=1;
V[44,49]=1; V[44,85]=1; V[44,56]=1;
V[45,91]=1; V[45,77]=1; V[45,89]=1 ; V[45,58]=1;
V[46,47]=1; V[46,82]=1;
V[47,82]=1;
V[49,53]=1; V[49,72]=1; V[49,86]=1; V[49,79]=1; V[49,85]=1;
V[50,61]=1; V[50,53]=1;
V[51,52]=1; V[51,55]=1; V[51,77]=1;
V[52,55]=1; V[52,88]=1; V[52,70]=1;
V[53,61]=1; V[53,72]=1;
V[54,55]=1; V[54,57]=1; V[54,88]=1;
V[55,88]=1;
V[57,67]=1;
V[58,71]=1; V[58,89]=1;
V[59,62]=1; V[59,80]=1;
V[60,76]=1; V[60,95]=1; V[60,77]=1; V[60,80]=1;
V[61,72]=1;
V[62,80]=1;
V[64,65]=1;
V[67,68]=1; V[67,88]=1;
V[68,90]=1; V[68,88]=1;
V[69,71]=1;
V[70,90]=1; V[70,88]=1;
V[73,74]=1;
V[75,94]=1; V[75,93]=1; V[75,92]=1;
```

```
V[76,80]=1;
V[77,89]=1; V[77,91]=1; V[77,93]=1; V[77,94]=1; V[77,95]=1;
V[78,91]=1; V[78,92]=1; V[78,95]=1;
V[79,85]=1; V[79,86]=1;
V[81,82]=1;
V[83,84]=1;
V[86,87]=1;
V[88,90]=1;
V[91,92]=1; V[91,94]=1;
V[92,93]=1; V[92,94]=1; V[92,95]=1;
V[93,94]=1; V[93,95]=1;

V = t(V) + V
```

## 2.2 1) Carte Noir Et Blanc

### Initialisation des parametres

```
In [556]: ## Nombre de niveaux de gris L souhaite
          L = 60

          ## Nombre d'iterations de l'algorithme
          nmax = 3000

          beta = 1 # donné
          N = 95 # Nombre de département
```

### Fonction d'erreur totale H

```
In [557]: compute_error = function(G)
          {
              H = 0
              for (i in 1:(N-1))
              {
                  for (j in (i+1):N)
                  {
                      H = H + V[i, j] * abs(G[i] - G[j])
                  }
              }

              return(H)
          }
```

### Transition 1

```
In [558]: transition1 = function(g, H)
          {
            gnew = g
            # on tire une coordonnée au hasard
```

```
  k0 = sample(1:N, 1)
  # on calcule la valeur de la coordonée + 1, et la valeur de la coordonée - 1
  coord_plus_1 = g[k0] + 1
  coord_moins_1 = g[k0] - 1

  # L'ensemble des nombres qu'on peut tirer au hasard :
  random_ensemble = c(max(coord_moins_1, 1), min(coord_plus_1, L))

  gnew[k0] = sample(random_ensemble, 1)

  delta = compute_error(gnew) - compute_error(g)

  alpha = min(exp(beta * delta), 1)

  u = runif(1)

  # on accepte si u < alpha car on MINIMISE !
  if (u < alpha)
  {
    # on accepte la transition
    result = c(gnew, H + delta)
  }
  else
  {
    # on rejette la transition
    result = c(g, H)
  }

  return(result)
}
```

### Transition 2

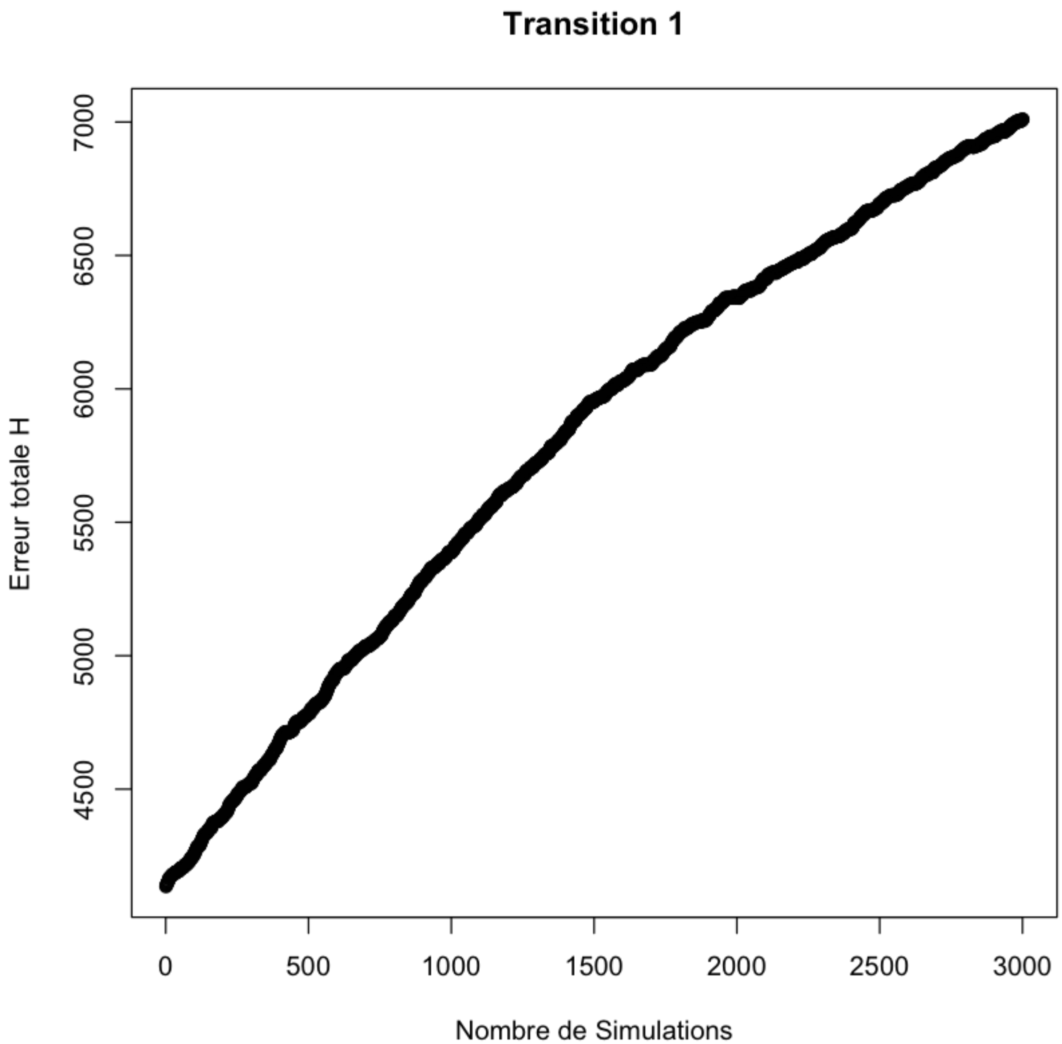
In [559]:
```
transition2 = function(g, H)
{
  gnew = g

  k0 = sample(1:N, 1)
  coord_plus_1 = g[k0] + 1
  coord_moins_1 = g[k0] - 1
  random_ensemble = c(max(coord_moins_1, 1), min(coord_plus_1, L))
  gnew[k0] = sample(random_ensemble, 1)

  pgibbs = rep(0, L)
  for (i_j in 1:L)
  {
    delta_num = 0
    for (i_l in 1:N)
```

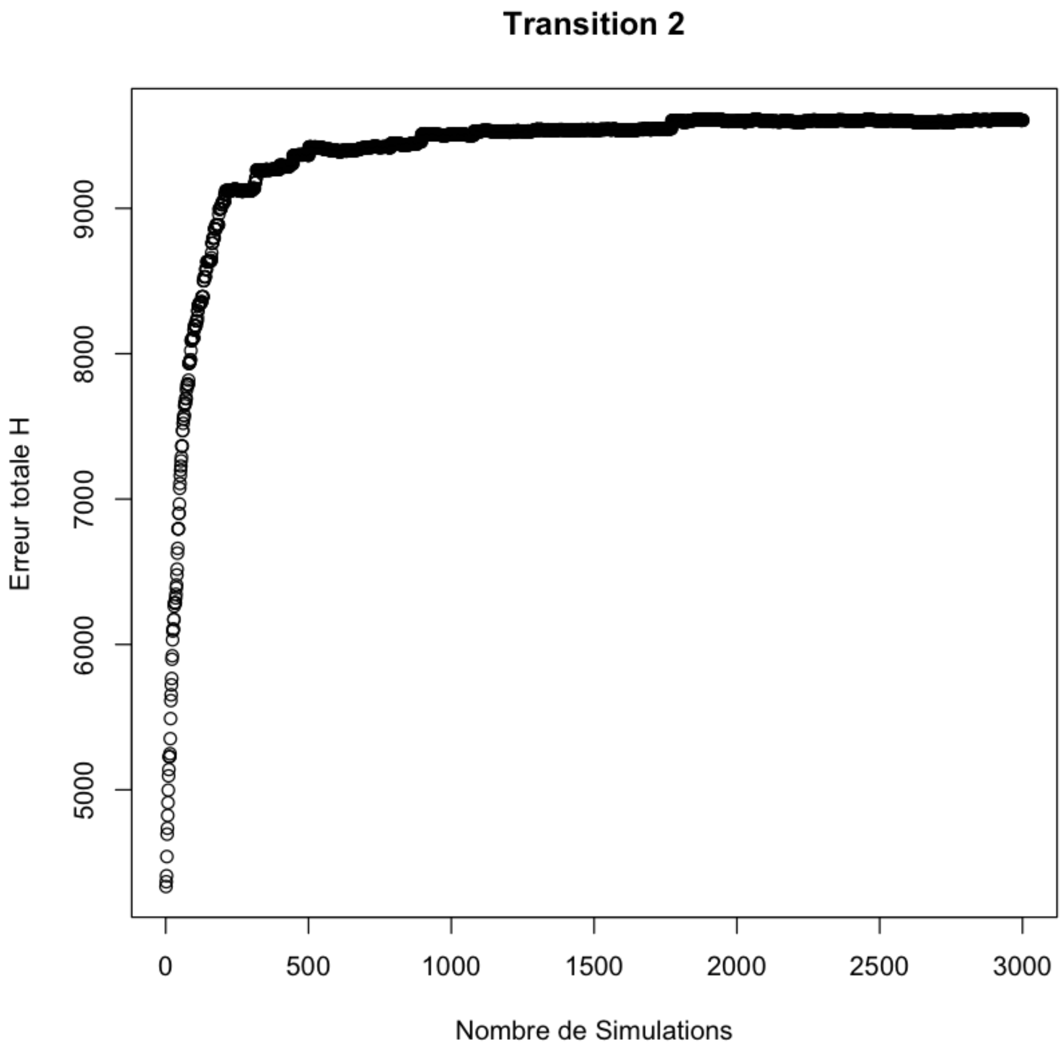
```
      {
        # On somme les différence car on MAXIMISE H !
        delta_num = delta_num + abs(i_j - g[i_l]) * V[k0, i_l]
      }
      pgibbs[i_j] = exp(beta * delta_num)
    }

    total = sum(pgibbs)
    pgibbs = pgibbs / total

    gnew[k0] = sample(1:L, prob=pgibbs)

    one_H = compute_error(gnew)

    result = c(gnew, one_H)

    return(result)
  }
```

**Calcul du G optimal et affichage de l'erreur total H**

```
In [560]: compute_G = function(G, nmax_simul, num_method)
          {
              transition_method = c(transition1, transition2)
              stopifnot(match(num_method, 1:2) > 0)

              H = rep(0, nmax_simul)
              H[1] = compute_error(G)

              for (i in 1:nmax_simul)
              {
                  #result = transition1(G, H[i])
                  result = transition_method[[num_method]](G, H[i])
                  G = result[1:N]
                  H[i+1] = result[N+1]
              }

              return(list("G"= G, "H"= H))
          }

          # Méthode Transition 1
          # Initialissation aleatoire des niveaux de gris des departements
          G = sample(1:L,N,replace=TRUE)

          result = compute_G(G, nmax, 1)

          G_opt1 = unlist(result["G"])
          H_opt1 = unlist(result["H"])
```

```r
plot(H_opt1,
    main="Transition 1",
    xlab="Nombre de Simulations",
    ylab="Erreur totale H")

# Méthode Transition 2
# Initialissation aleatoire des niveaux de gris des departements
G = sample(1:L,N,replace=TRUE)

# on enlève les warning car on en déclenche certains
old_warn_val = getOption("warn")
options(warn = -1)

result = compute_G(G, nmax, 2)

G_opt2 = unlist(result["G"])
H_opt2 = unlist(result["H"])

plot(H_opt2,
    main="Transition 2",
    xlab="Nombre de Simulations",
    ylab="Erreur totale H")

# On remet les warnings
options(warn = old_warn_val)
```


Transition 1
Erreur totale H
4500
5000
5500
6000
6500
7000
0
500
1000
1500
2000
2500
3000
Nombre de Simulations



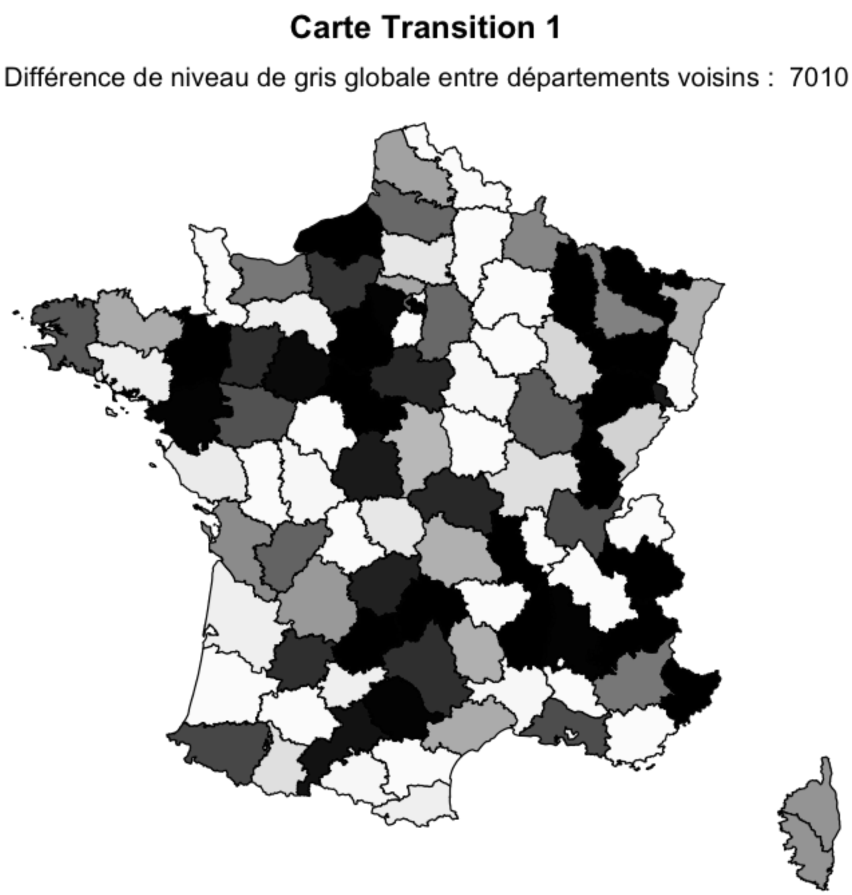

**Affichage de la carte**

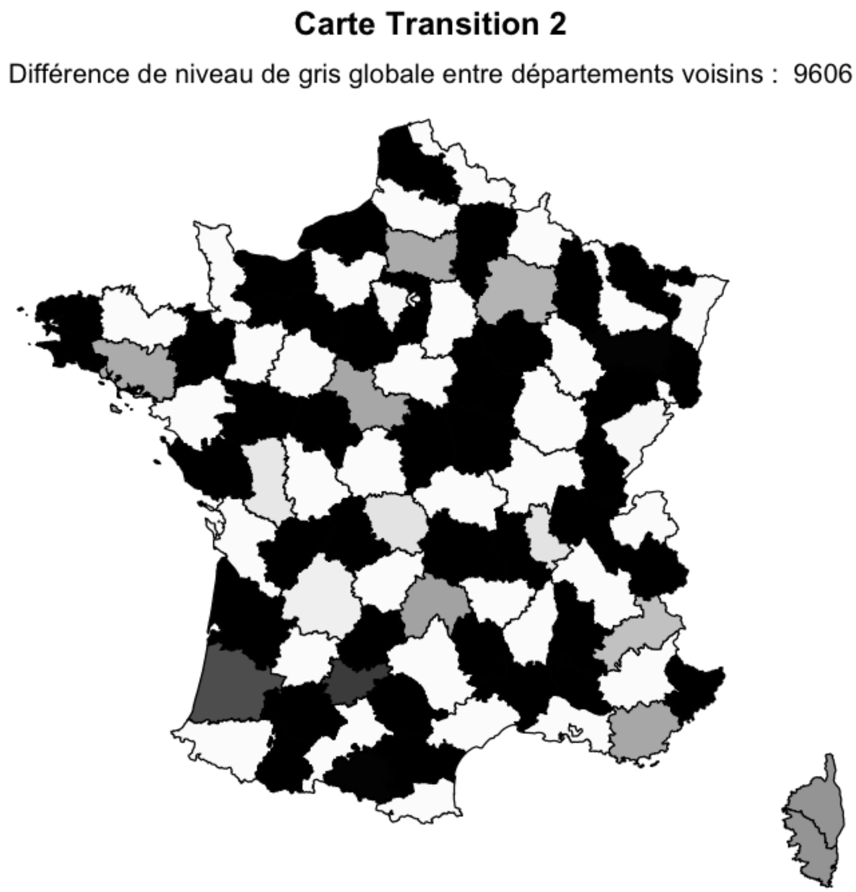
```
In [561]: # On change notre "Working Directory" afin de trouver les fichiers nécessaires
          setwd(dir=getwd())

          #### Méthode Transition 1
          ##############################

          ## Vecteur couleus des niveau de gris suivant G

          vect_couleurs= gray(1-G_opt1/L)

          ## recupertion du fond de carte et affichage
```

```
library(maptools)
fdc = readShapeSpatial("DEPARTEMENT")

A=fdc
A$CODE_DEPT=as.numeric(as.character(A$CODE_DEPT))
ligne=which(is.na(A$CODE_DEPT))
A$CODE_DEPT[ligne]=20
A = A[order(A$CODE_DEPT),]
fdc=A[order(A$CODE_DEPT),]

vect_couleurs=c(vect_couleurs[1:20],vect_couleurs[20],vect_couleurs[21:95])

plot(fdc,
     col=vect_couleurs,
    main="Carte Transition 1")
mtext(paste("Différence de niveau de gris globale entre départements voisins : ", as

#### Méthode Transition 2
##############################

## Vecteur couleus des niveau de gris suivant G

vect_couleurs= gray(1-G_opt2/L)

## recupertion du fond de carte et affichage

library(maptools)
fdc = readShapeSpatial("DEPARTEMENT")

A=fdc
A$CODE_DEPT=as.numeric(as.character(A$CODE_DEPT))
ligne=which(is.na(A$CODE_DEPT))
A$CODE_DEPT[ligne]=20
A = A[order(A$CODE_DEPT),]
fdc=A[order(A$CODE_DEPT),]

vect_couleurs=c(vect_couleurs[1:20],vect_couleurs[20],vect_couleurs[21:95])

plot(fdc,
     col=vect_couleurs,
    main="Carte Transition 2")
mtext(paste("Différence de niveau de gris globale entre départements voisins : ", as
```

```
Warning message:
readShapeSpatial is deprecated; use rgdal::readOGR or sf::st_readWarning message:
readShapePoly is deprecated; use rgdal::readOGR or sf::st_readWarning message in eval(expr, en
NAs introduced by coercionWarning message:
```

```
readShapeSpatial is deprecated; use rgdal::readOGR or sf::st_readWarning message:
readShapePoly is deprecated; use rgdal::readOGR or sf::st_readWarning message in eval(expr, env
NAs introduced by coercion
```






## 2.3 Carte Couleur

### Initialisation des pramètres

```
In [562]: # Nombre de couleur L souhaite
          L = 4
          # Nombre d'iterations de l'algorithme
          nmax = 10000
          # donnee
          beta = 1
          # Nombre de département
          N = 95
```

**Fonction d'erreur totale H**

```
In [563]: compute_error = function(G)
          {
              H = 0
              for (i in 1:(N-1))
              {
                  for (j in (i+1):N)
                  {
                      H = H + (G[i] == G[j]) * V[i, j]
                  }
              }

              return(H)
          }
```

**Transition 1**

```
In [564]: transition1 = function(g, H, beta)
          {
            gnew = g
            # on tire une coordonnée au hasard
            k0 = sample(1:N, 1)
            # on calcule la valeur de la coordonée + 1, et la valeur de la coordonée - 1
            coord_plus_1 = g[k0] + 1
            coord_moins_1 = g[k0] - 1

            # On tire un nombre au hasard : max(coord_moins_1, 1) OU min(coord_plus_1, L)
            gnew[k0] = sample(c(max(coord_moins_1, 1), min(coord_plus_1, L)), 1)

            # On calcul H (ou delta) : c'est la différence entre les anciennes et les nouvelle
            delta = compute_error(gnew) - compute_error(g)

            alpha = min(exp(beta * delta), 1)

            u = runif(1)

            # on accepte si u > alpha car on MAXIMISE !
            if (u > alpha)
            {
              # on accepte la transition
              result = c(gnew, H + delta)
            }
            else
            {
              # on rejette la transition
              result = c(g, H)
            }
```

```
    return(result)
}
```

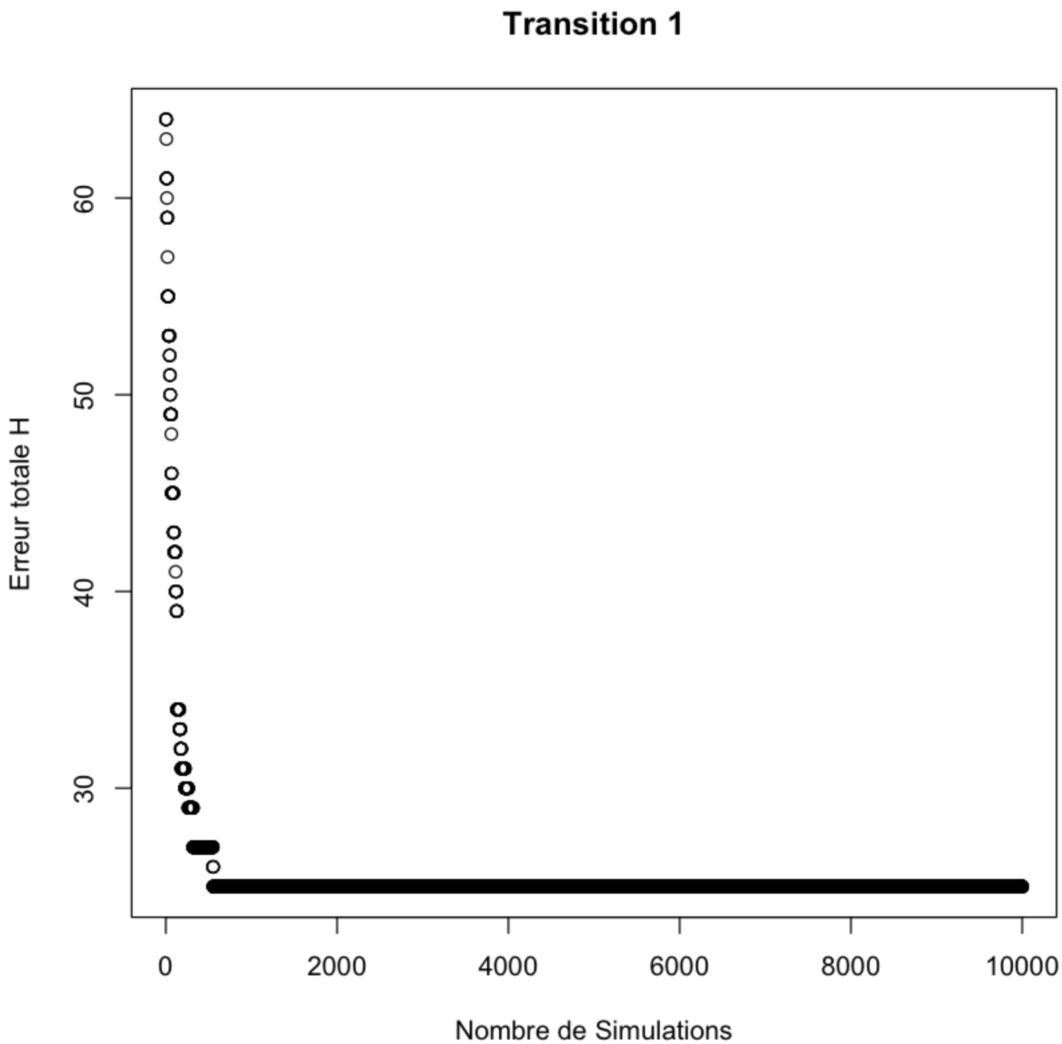
#### Transition 2

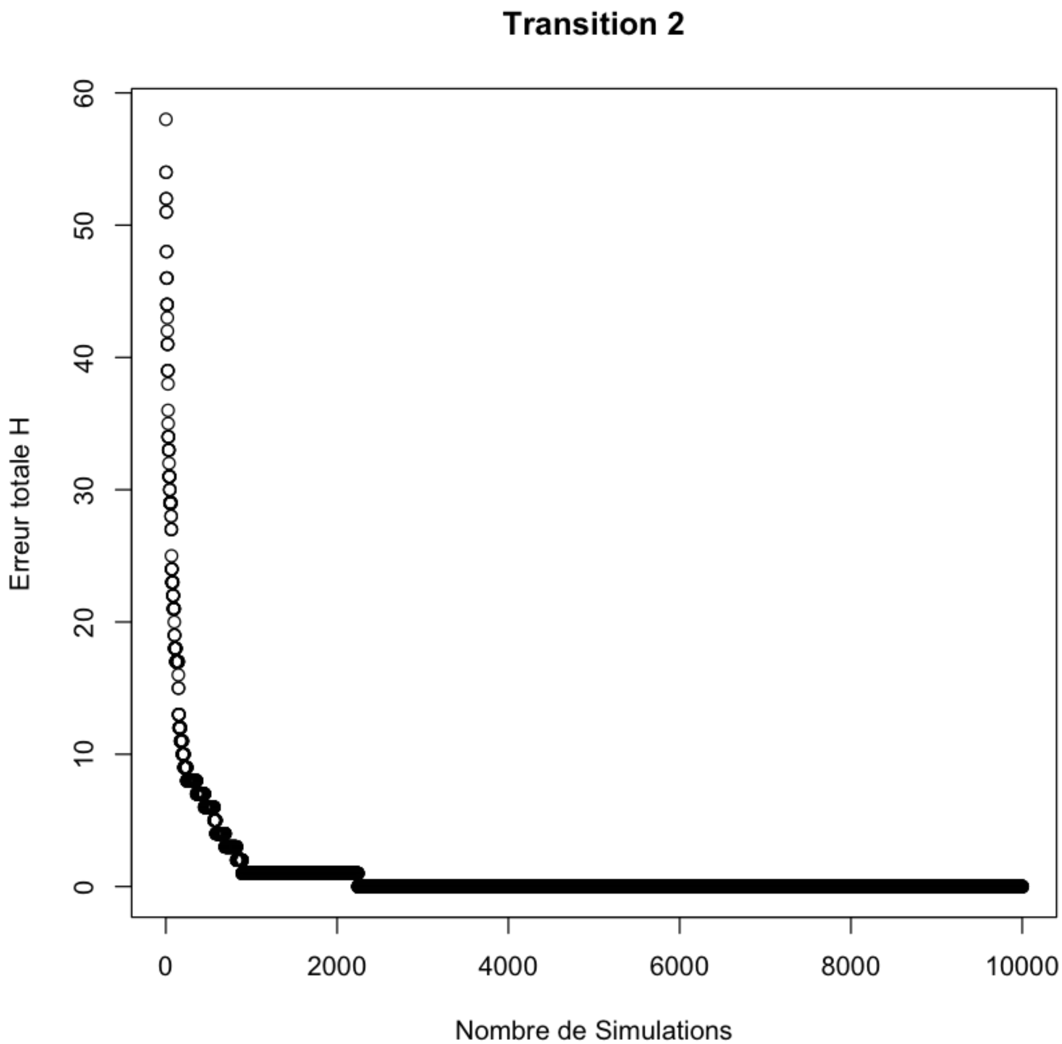
```
In [565]: transition2 = function(g, H, beta)
          {
            gnew = g

            k0 = sample(1:N, 1)
            coord_plus_1 = g[k0] + 1
            coord_moins_1 = g[k0] - 1
            random_ensemble = c(max(coord_moins_1, 1), min(coord_plus_1, L))
            gnew[k0] = sample(random_ensemble, 1)

            pgibbs = rep(0, L)
            for (i_j in 1:L)
            {
              delta_num = 0
              for (i_l in 1:N)
              {
                # On soustrait les différences car on MINIMISE H !
                delta_num = delta_num - abs(i_j == g[i_l]) * V[k0, i_l]
              }
              pgibbs[i_j] = exp(beta * delta_num)
            }

            total = sum(pgibbs)
            pgibbs = pgibbs / total

            gnew[k0] = sample(1:L, prob=pgibbs)

            one_H = compute_error(gnew)

            result = c(gnew, one_H)

            return(result)
          }
```

#### Calcul du G optimal et affichage de l'erreur total H

```
In [566]: compute_G = function(G, nmax_simul, num_method)
          {
              transition_method = c(transition1, transition2)
              stopifnot(match(num_method, 1:2) > 0)

              H = rep(0, nmax_simul)
```

```r
    H[1] = compute_error(G)

    beta_0 = 1
    for (i in 1:nmax_simul)
    {
        result = transition_method[[num_method]](G, H[i], beta)
        G = result[1:N]
        H[i+1] = result[N+1]
        beta = beta_0 * sqrt(i)
    }

    return(list("G"= G, "H"= H))
}

# Méthode Transition 1
# Initialissation aleatoire des couleurs des departements
G = sample(1:L,N,replace=TRUE)

result = compute_G(G, nmax, 1)

G_opt1 = unlist(result["G"])
H_opt1 = unlist(result["H"])

plot(H_opt1,
    main="Transition 1",
    xlab="Nombre de Simulations",
    ylab="Erreur totale H")

# Méthode Transition 2
# Initialissation aleatoire des couleurs des departements
G = sample(1:L,N,replace=TRUE)

# on enlève les warning car on en déclenche certains
old_warn_val = getOption("warn")
options(warn = -1)

result = compute_G(G, nmax, 2)

G_opt2 = unlist(result["G"])
H_opt2 = unlist(result["H"])

plot(H_opt2,
    main="Transition 2",
    xlab="Nombre de Simulations",
    ylab="Erreur totale H")

# On remet les warnings
options(warn = old_warn_val)
```


Transition 1
Erreur totale H
30
40
50
60
0
2000
4000
6000
8000
10000
Nombre de Simulations




**Affichage de la Carte**

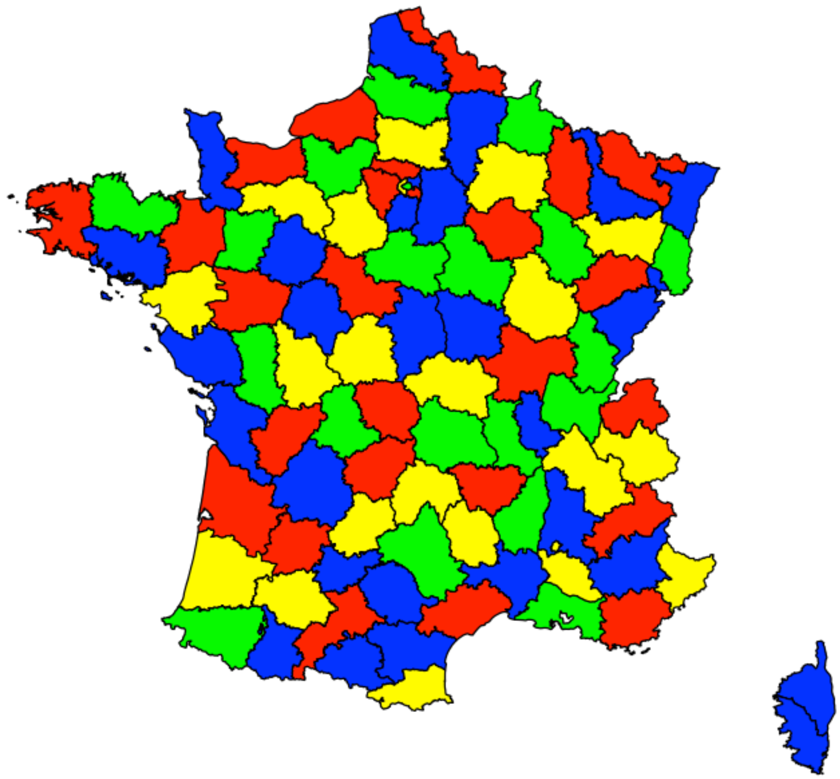
```
In [567]: # On change notre "Working Directory" afin de trouver les fichiers nécessaires
          setwd(dir=getwd())

          couleurs = c("blue", "red", "yellow", "green")

          # Carte Transition 1
          ####################

          vect_couleurs= couleurs[G_opt1]

          ## recupertion du fond de carte et affichage
```

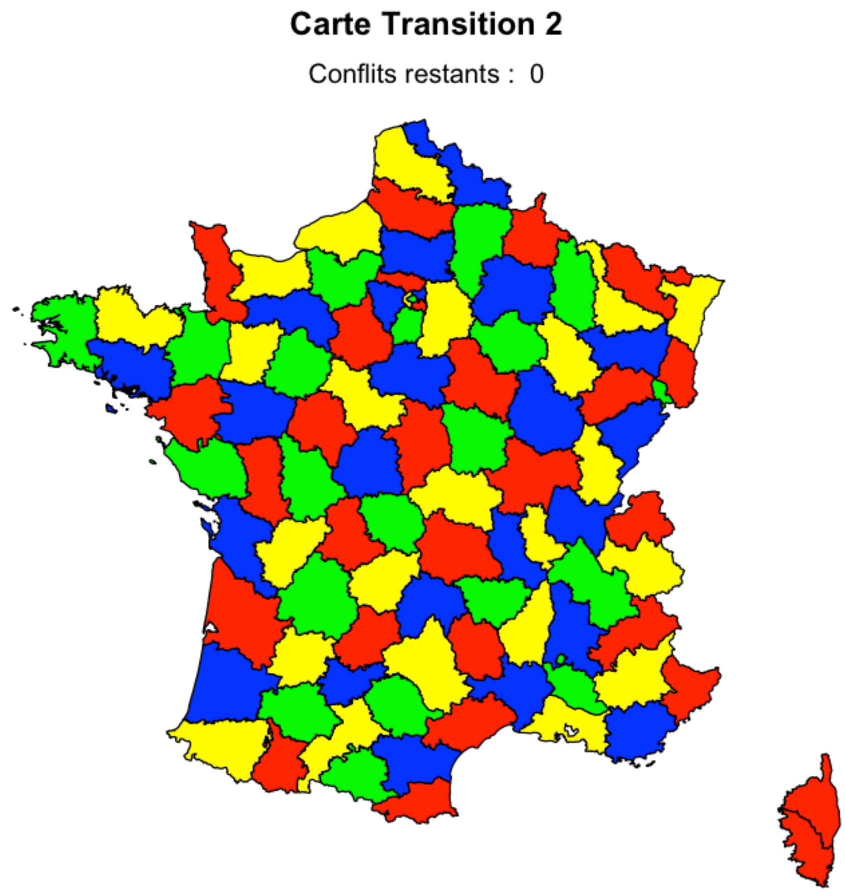
```r
library(maptools)
fdc = readShapeSpatial("DEPARTEMENT")

A=fdc
A$CODE_DEPT=as.numeric(as.character(A$CODE_DEPT))
ligne=which(is.na(A$CODE_DEPT))
A$CODE_DEPT[ligne]=20
A = A[order(A$CODE_DEPT),]
fdc=A[order(A$CODE_DEPT),]

vect_couleurs=c(vect_couleurs[1:20],vect_couleurs[20],vect_couleurs[21:95])

plot(fdc,
     col=vect_couleurs,
    main="Carte Transition 1")
mtext(paste("Conflits restants : ", as.character(H_opt1[nmax])))

# Carte Transition 2
####################

vect_couleurs= couleurs[G_opt2]

## recupertion du fond de carte et affichage
library(maptools)
fdc = readShapeSpatial("DEPARTEMENT")

A=fdc
A$CODE_DEPT=as.numeric(as.character(A$CODE_DEPT))
ligne=which(is.na(A$CODE_DEPT))
A$CODE_DEPT[ligne]=20
A = A[order(A$CODE_DEPT),]
fdc=A[order(A$CODE_DEPT),]

vect_couleurs=c(vect_couleurs[1:20],vect_couleurs[20],vect_couleurs[21:95])

plot(fdc,
     col=vect_couleurs,
    main="Carte Transition 2")
mtext(paste("Conflits restants : ", as.character(H_opt2[nmax])))
```

```
Warning message:
readShapeSpatial is deprecated; use rgdal::readOGR or sf::st_readWarning message:
readShapePoly is deprecated; use rgdal::readOGR or sf::st_readWarning message in eval(expr, en
NAs introduced by coercionWarning message:
readShapeSpatial is deprecated; use rgdal::readOGR or sf::st_readWarning message:
readShapePoly is deprecated; use rgdal::readOGR or sf::st_readWarning message in eval(expr, en
NAs introduced by coercion
```

**Carte Transition 1**

Conflits restants : 25






#### 2.3.1 Conclusion

- Avec transition 1 : on passe de 70 conflits environ à 20 conflits envirion (ie, H = 20)
- Avec transition 2 : on passe de 70 conflits environ à 0 conflits envirion (ie, H = 0)
- on veut trouver le H le plus faible possible –> La méthode de transition 2 est meilleur et permet de résoudre complétement le problème (H = 0, le problème est résolu)
- La méthode de transition 1 ne permet, elle, pas de résoudre complétement le problème (il reste des conflits)

## 3 Problème du Voyageur de Commerce

#### 3.0.1 Initialisation

In [1]:
```
## Algorithme de Recuit simule pour le probleme du voyageur de commerce
## Depart et retour a  Paris

N=40 ## Nombre de villes (Hors Paris)

## Matrice des distances ; les villes de 1 a 37 sont : Paris, Clermont,
## Poitiers, Gueret, Dijon, Macon, Tours, Le Mans, Auxerre, Angers,
## Nantes, Bordeaux, Bourges, Caen, Troyes, Rennes, Orleans, Lyon, Nancy,
## Strasbourg, Toulouse, Lille, Grenoble, Marseille, Carcassonne, Reims,
## Aurillac, Amiens, Nimes, La Rochelle, Pau, Mende, Nice, Limoges, Mulhouse.
## Agen, Rouen, Valence, Annecy, Brest, Sisteron

V =
array(0,dim=c(N+1,N+1))

V[1,2]=42; V[1,3]=38; V[1,4]=39; V[1,5]=33; V[1,6]=43; V[1,7]=26;
V[1,8]=24; V[1,9]=19; V[1,10]=34; V[1,11]=39; V[1,12]=57; V[1,13]=25;
V[1,14]=23; V[1,15]=17; V[1,16]=35; V[1,17]=13; V[1,18]=46; V[1,19]=30;
V[1,20]=44; V[1,21]=66; V[1,22]=22; V[1,23]=55; V[1,24]=78; V[1,25]=77;
V[1,26]=14; V[1,27]=58; V[1,28]=13; V[1,29]=71; V[1,30]=47; V[1,31]=78;
V[1,32]=56; V[1,33]=87; V[1,34]=38; V[1,35]=46; V[1,36]= 66; V[1,37]=13;
V[1,38]=56 ; V[1,39]=52; V[1,40]=59; V[1,41]=71;
V[2,3]=30; V[2,4]=13; V[2, 5]=29; V[2,6]=19; V[2,7]=33; V[2,8]=44;
V[2,9]=30; V[2,10]=44; V[2,11]=54; V[2,12]=37; V[2,13]=19; V[2,14]=59;
V[2,15]=37; V[2,16]=60; V[2,17]=30; V[2,18]=18; V[2,19]=46; V[2,20]=55;
V[2,21]= 36; V[2,22]=61; V[2,23]=28; V[2,24]=48; V[2,25]=43; V[2,26]=53;
V[2,27]=16; V[2,28]=52; V[2,29]=32; V[2,30]=40; V[2,31]=50; V[2,32]=17;
V[2,33]=56; V[2,34]=18; V[2,35]=45; V[2,36]=36; V[2,37]=51; V[2,38]=24;
V[2,39]=30; V[2,40]=83; V[2,41]=40;
V[3,4]=16; V[3,5]=43; V[3,6]=45; V[3,7]=12; V[3,8]=20; V[3,9]=36;
V[3,10]=16; V[3,11]=22; V[3,12]=23; V[3,13]=24; V[3,14]=36; V[3,15]=41;
V[3,16]=29; V[3,17]=22; V[3,18]=49; V[3,19]=58; V[3,20]=71; V[3,21]=41;
V[3,22]=55; V[3,23]=52; V[3,24]=79; V[3,25]=50; V[3,26]=47; V[3,27]=31;
V[3,28]=44; V[3,29]=58; V[3,30]=14; V[3,31]=42; V[3,32]=44; V[3,33]=83;
V[3,34]=12; V[3,35]=63; V[3,36]=33; V[3,37]=38; V[3,38]=50; V[3,39]=54;
V[3,40]=48; V[3,41]=71;
V[4,5]=35; V[4,6]=29; V[4,7]=21; V[4,8]=31; V[4,9]=29; V[4,10]=31;
V[4,11]=34; V[4,12]=31; V[4,13]=17; V[4,14]=52; V[4,15]=41; V[4,16]=44;
V[4,17]=27; V[4,18]=33; V[4,19]=51; V[4,20]=61; V[4,21]=34; V[4,22]=55;
V[4,23]=39; V[4,24]=64; V[4,25]=47; V[4,26]=56; V[4,27]=21; V[4,28]=46;
V[4,29]=47; V[4,30]=27; V[4,31]=44; V[4,32]=30; V[4,33]=68; V[4,34]=8;
V[4,35]=51; V[4,36]=36; V[4,37]=47; V[4,38]=36; V[4,39]=41; V[4,40]=62;
V[4,41]=57;
V[5,6]=15; V[5, 7]=42; V[5,8]=48; V[5,9]=15 ; V[5,10]=55 ; V[5,11]=64;
```

```
V[5,12]=67; V[5,13]=25; V[5,14]=55; V[5,15]=18; V[5,16]=62; V[5,17]=30;
V[5,18]=19; V[5,19]=20; V[5,20]=31; V[5,21]=61; V[5,22]=47; V[5,23]=28;
V[5,24]=51; V[5,25]=64; V[5,26]=28; V[5,27]=45; V[5,28]=41; V[5,29]=44;
V[5,30]=57; V[5,31]=75; V[5,32]=41; V[5,33]=59; V[5,34]=41; V[5,35]=21;
V[5,36]=64; V[5,37]=44; V[5,38]=29; V[5,39]=23; V[5,40]=86; V[5,41]=44;
V[6,7]=44; V[6,8]=25; V[6,9]=25; V[6,10]=56; V[6,11]=60 ; V[6,12]=58;
V[6,13]=25; V[6,14]=63; V[6,15]=31; V[6,16]=70; V[6,17]=38; V[6,18]=7;
V[6,19]=32;V[6,20]=40; V[6,21]=54; V[6,22]=59; V[6,23]=17; V[6,24]=38;
V[6,25]=52; V[6,26]=42; V[6,27]=37; V[6,28]=52; V[6,29]=31; V[6,30]=53;
V[6,31]=69; V[6,32]=28; V[6,33]=49; V[6,34]=34; V[6,35]=29; V[6,36]=59;
V[6,37]=52; V[6,38]=17 ; V[6,39]=14; V[6,40]=89; V[6,41]=32;
V[7,8]=10; V[7,9]=28; V[7,10]=12; V[7,11]=21; V[7,12]=33; V[7,13]=16;
V[7,14]=26; V[7,15]=31; V[7,16]=26; V[7,17]=12; V[7,18]=43;V[7,19]=49;
V[7,20]=63; V[7,21]=49; V[7,22]=45; V[7,23]=53; V[7,24]=80; V[7,25]=61;
V[7,26]=37; V[7,27]=41; V[7,28]=35; V[7,29]=62; V[7,30]=23; V[7,31]=52;
V[7,32]=47; V[7,33]=85; V[7,34]=20; V[7,35]=59; V[7,36]=44; V[7,37]=28;
V[7,38]=52; V[7,39]=53; V[7,40]=50; V[7,41]=73;
V[8,9]=23; V[8,10]=11; V[8,11]=18; V[8,12]=43; V[8,13]=26; V[8,14]=17;
V[8,15]=34; V[8,16]=16; V[8,17]=14; V[8,18]=50; V[8,19]=50; V[8,20]=64;
V[8,21]=57; V[8,22]=42; V[8,23]=60; V[8,24]=90; V[8,25]=70; V[8,26]=34;
V[8,27]=51; V[8,28]=31; V[8,29]=70; V[8,30]=27; V[8,31]=59; V[8,32]=56;
V[8,33]=93; V[8,34]=29; V[8,35]=62; V[8,36]=54; V[8,37]=21; V[8,38]=59;
V[8,39]=60; V[8,40]=40; V[8,41]=83;
V[9,10]=40; V[9,11]=49; V[9,12]=60; V[9,13]=15; V[9,14]=40; V[9,15]=8;
V[9,16]=48; V[9,17]=15; V[9,18]=30; V[9,19]=26; V[9,20]=39; V[9,21]=61;
V[9,22]=38; V[9,23]=39; V[9,24]=61; V[9,25]=71; V[9,26]=21; V[9,27]=43;
V[9,28]=30; V[9,29]=54; V[9,30]=46; V[9,31]=69; V[9,32]=43; V[9,33]=70;
V[9,34]=32; V[9,35]=35; V[9,36]=60; V[9,37]=30; V[9,38]=40; V[9,39]=36;
V[9,40]=72; V[9,41]=55;
V[10,11]=9; V[10,12]=38; V[10,13]=27; V[10,14]=25; V[10,15]=43; V[10,16]=13;
V[10,17]=25; V[10,18]=52; V[10,19]=58; V[10,20]=72; V[10,21]=54; V[10,22]=51;
V[10,23]=63; V[10,24]=91; V[10,25]=64; V[10,26]=43; V[10,27]=47; V[10,28]=39;
V[10,29]=72; V[10,30]=18; V[10,31]=53; V[10,32]=57; V[10,33]=95; V[10,34]=25;
V[10,35]=70; V[10,36]=48; V[10,37]=30; V[10,38]=61; V[10,39]=63; V[10,40]=37;
V[10,41]=84;
V[11,12]=33; V[11,13]=36; V[11,14]=29; V[11,15]=52; V[11,16]=11; V[11,17]=33;
V[11,18]=60; V[11,19]=67; V[11,20]=81; V[11,21]=55; V[11,22]=60; V[11,23]=70;
V[11,24]=97; V[11,25]=66; V[11,26]=52; V[11,27]=54; V[11,28]=48; V[11,29]=76;
V[11,30]=14; V[11,31]=51; V[11,32]=61; V[11,33]=101; V[11,34]=30; V[11,35]=79;
V[11,36]=46; V[11,37]=39; V[11,38]=69; V[11,39]=72; V[11,40]=30; V[11,41]=93;
V[12,13]=42 ; V[12,14]=59; V[12,15]=65; V[12,16]=45; V[12,17]=45; V[12,18]=53;
V[12,19]=78; V[12,20]=91; V[12,21]=24; V[12,22]=77; V[12,23]=64; V[12,24]=65;
V[12,25]=34; V[12,26]=73; V[12,27]=31; V[12,28]=67; V[12,29]=51; V[12,30]=18;
V[12,31]=19; V[12,32]=41; V[12,33]=78; V[12,34]=22; V[12,35]=80;V[12,36]=14;
V[12,37]=65; V[12,38]=57; V[12,39]=66; V[12,40]=62; V[12,41]=73;
V[13,14]=42; V[13,15]=23; V[13,16]=41; V[13,17]=12; V[13,18]=28; V[13,19]=40;
V[13,20]=53; V[13,21]=47; V[13,22]=44; V[13,23]=39; V[13,24]=65; V[13,25]=57;
V[13,26]=35; V[13,27]=34; V[13,28]=36; V[13,29]=50; V[13,30]=32; V[13,31]=56;
```

```
V[13,32]=35; V[13,33]=70; V[13,34]=19; V[13,35]=45; V[13,36]=43; V[13,37]=33;
V[13,38]=37; V[13,39]=38; V[13,40]=64; V[13,41]=55;
V[14,15]=41; V[14,16]=19; V[14,17]=32; V[14,18]=65; V[14,19]=52; V[14,20]=66;
V[14,21]=73; V[14,22]=35; V[14,23]=75; V[14,24]=101; V[14,25]=87; V[14,26]=38;
V[14,27]=67; V[14,28]=24; V[14,29]=86; V[14,30]=40; V[14,31]=75; V[14,32]=71;
V[14,33]=107; V[14,34]=44; V[14,35]=68; V[14,36]=74; V[14,37]=13; V[14,38]=74;
V[14,39]=73; V[14,40]=37; V[14,41]=93;
V[15,16]=48; V[15,17]=19; V[15,18]=41; V[15,19]=19; V[15,20]=32; V[15,21]=68;
V[15,22]=32; V[15,23]=43; V[15,24]=69; V[15,25]=80; V[15,26]=13; V[15,27]=51;
V[15,28]=26; V[15,29]=58; V[15,30]=53; V[15,31]=76; V[15,32]=51; V[15,33]=75;
V[15,34]=40; V[15,35]=30; V[15,36]=69; V[15,37]=31; V[15,38]=44; V[15,39]=39;
V[15,40]=73; V[15,41]=62;
V[16,17]=30; V[16,18]=64; V[16,19]=64; V[16,20]=78; V[16,21]=66; V[16,22]=53;
V[16,23]=75; V[16,24]=106; V[16,25]=78; V[16,26]=49; V[16,27]=60; V[16,28]=42;
V[16,29]=83; V[16,30]=25; V[16,31]=62; V[16,32]=68; V[16,33]=106; V[16,34]=38;
V[16,35]=77; V[16,36]=58; V[16,37]=31; V[16,38]=73; V[16,39]=74; V[16,40]=24;
V[16,41]=97;
V[17,18]=39; V[17,19]=37; V[17,20]=51; V[17,21]=54; V[17,22]=34; V[17,23]=49;
V[17,24]=76; V[17,25]=65; V[17,26]=27; V[17,27]=45; V[17,28]=25; V[17,29]=61;
V[17,30]=35; V[17,31]=62; V[17,32]=46; V[17,33]=81; V[17,34]=26; V[17,35]=48;
V[17,36]=54; V[17,37]=22; V[17,38]=47; V[17,39]=48; V[17,40]=55; V[17,41]=69;
V[18,19]=39; V[18,20]=45; V[18,21]=47; V[18,22]=66; V[18,23]=10; V[18,24]=32;
V[18,25]=45; V[18,26]=49; V[18,27]=30; V[18,28]=59; V[18,29]=24; V[18,30]=55;
V[18,31]=65; V[18,32]=22; V[18,33]=42; V[18,34]=33; V[18,35]=34; V[18,36]=54;
V[18,37]=59; V[18,38]=10; V[18,39]=14; V[18,40]=97; V[18,41]=24;
V[19,20]=14; V[19,21]=81; V[19,22]=38; V[19,23]=47; V[19,24]=72; V[19,25]=85;
V[19,26]=21; V[19,27]=66; V[19,28]=35; V[19,29]=63; V[19,30]=71; V[19,31]=95;
V[19,32]=60; V[19,33]=78; V[19,34]=58; V[19,35]=17; V[19,36]=85; V[19,37]=45;
V[19,38]=49; V[19,39]=40; V[19,40]=88; V[19,41]=65;
V[20,21]=90; V[20,22]=48; V[20,23]=51; V[20,24]=81; V[20,25]=94; V[20,26]=35;
V[20,27]=75; V[20,28]=49; V[20,29]=69; V[20,30]=85; V[20,31]=104; V[20,32]=67;
V[20,33]=76; V[20,34]=58; V[20,35]=11; V[20,36]=94; V[20,37]=62; V[20,38]=55;
V[20,39]=41; V[20,40]=107; V[20,41]=67;
V[21,22]=88; V[21,23]=52; V[21,24]=41; V[21,25]=9; V[21,26]=81; V[21,27]=25;
V[21,28]=80; V[21,29]=28; V[21,30]=41; V[21,31]=19; V[21,32]=24; V[21,33]=55;
V[21,34]=28; V[21,35]=80; V[21,36]=11; V[21,37]=76; V[21,38]=43; V[21,39]=60;
V[21,40]=86; V[21,41]=49;
V[22,23]=75; V[22,24]=100; V[22,25]=97; V[22,26]=20; V[22,27]=79; V[22,28]=12;
V[22,29]=90; V[22,30]=68; V[22,31]=96; V[22,32]=77; V[22,33]=106; V[22,34]=60;
V[22,35]=55; V[22,36]=88; V[22,37]=25; V[22,38]=76; V[22,39]=69; V[22,40]=75;
V[22,41]=93;
V[23,24]=28; V[23,25]=44; V[23,26]=59; V[23,27]=39; V[23,28]=68; V[23,29]=24;
V[23,30]=65; V[23,31]=71; V[23,32]=27; V[23,33]=32; V[23,34]=44; V[23,35]=40;
V[23,36]=60; V[23,37]=70; V[23,38]=9; V[23,39]=10; V[23,40]=108; V[23,41]=14;
V[24,25]=31; V[24,26]=80; V[24,27]=43; V[24,28]=90; V[24,29]=12; V[24,30]=76;
V[24,31]=58; V[24,32]=26; V[24,33]=18; V[24,34]=57; V[24,35]=65; V[24,36]=52;
V[24,37]=90; V[24,38]=21; V[24,39]=37; V[24,40]=136; V[24,41]=13;
V[25,26]=90; V[25,27]=26; V[25,28]=89; V[25,29]=19; V[25,30]=50; V[25,31]=28;
```

```
V[25,32]=27; V[25,33]=46; V[25,34]=38; V[25,35]=78; V[25,36]=21; V[25,37]=85;
V[25,38]=34; V[25,39]=51; V[25,40]=96; V[25,41]=40;
V[26,27]=68; V[26,28]=16; V[26,29]=71; V[26,30]=59; V[26,31]=87; V[26,32]=63;
V[26,33]=87; V[26,34]=50; V[26,35]=36; V[26,36]=80; V[26,37]=24; V[26,38]=56;
V[26,39]=50; V[26,40]=73; V[26,41]=73;
V[27,28]=67; V[27,29]=29; V[27,30]=38; V[27,31]=37; V[27,32]=14; V[27,33]=57;
V[27,34]=17; V[27,35]=60; V[27,36]=22; V[27,37]=64; V[27,38]=27; V[27,39]=42;
V[27,40]=77; V[27,41]=43;
V[28,29]=83; V[28,30]=57; V[28,31]=86; V[28,32]=69; V[28,33]=100; V[28,34]=51;
V[28,35]=52; V[28,36]=80; V[28,37]=12; V[28,38]=69; V[28,39]=64; V[28,40]=62;
V[28,41]=84;
V[29,30]=66; V[29,31]=47; V[29,32]=14; V[29,33]=28; V[29,34]=46; V[29,35]=59;
V[29,36]=40; V[29,37]=84; V[29,38]=14; V[29,39]=32; V[29,40]=108; V[29,41]=20;
V[30,31]=38; V[30,32]=53; V[30,33]=93; V[30,34]=23; V[30,35]=77; V[30,36]=32;
V[30,37]=49; V[30,38]=64; V[30,39]=68; V[30,40]=44; V[30,41]=86;
V[31,32]=43; V[31,33]=74; V[31,34]=37; V[31,35]=94; V[31,36]=17; V[31,37]=85;
V[31,38]=61; V[31,39]=78; V[31,40]=82; V[31,41]=67;
V[32,33]=42; V[32,34]=32; V[32,35]=56; V[32,36]=32; V[32,37]=67; V[32,38]=18;
V[32,39]=35; V[32,40]=95; V[32,41]=31;
V[33,34]=73; V[33,35]=65; V[33,36]=67; V[33,37]=106; V[33,38]=35; V[33,39]=42;
V[33,40]=142; V[33,41]=18;
V[34,35]=60; V[34,36]=24; V[34,37]=48; V[34,38]=41; V[34,39]=47; V[34,40]=60;
V[34,41]=65;
V[35,36]=83; V[35,37]=61; V[35,38]=44; V[35,39]=30; V[35,40]=102; V[35,41]=57;
V[36,37]=75; V[36,38]=48; V[36,39]=64; V[36,40]=76; V[36,41]=60;
V[37,38]=68; V[37,39]=64; V[37,40]=50; V[37,41]=83;
V[38,39]=17; V[38,40]=107; V[38,41]=17;
V[39,40]=104; V[39,41]=25;
V[40,41]=122;

V=V+t(V)
```

#### 3.0.2 Fonction d'erreur totale H

In [2]:
```
compute_error = function(G)
{
  H = V[1, G[1] + 1]
  for (i in 1:(N-1))
  {
    H = H + V[G[i] + 1, G[i+1] + 1]
  }
  H = H + V[G[N] + 1, 1]

  return(H)
}
```

### 3.0.3 Transition 1

```
In [3]: transition1 = function(G, beta)
        {
          gnew = G

          k_rnd = sample(1:N, 2, replace=FALSE)
          k01 = k_rnd[1]
          k02 = k_rnd[2]

          new_city_1 = gnew[k01]
          new_city_2 = gnew[k02]

          gnew[k01] = new_city_2
          gnew[k02] = new_city_1

          current_error = compute_error(G)
          new_error = compute_error(gnew)

          delta = new_error - current_error

          #alpha = min(exp(beta * delta), 1)
          # TEST
          alpha = min(exp(-beta * delta), 1)

          u = runif(1)

          #if (alpha < u)
          if (alpha > u)
          {
            return(gnew)
          }
          else
          {
            return(G)
          }
        }
```

### 3.0.4 Transition 2

```
In [4]: transition2 = function(G, beta)
        {
          gnew = G

          k_rnd = sample(1:N, 2, replace=FALSE)
          k01 = k_rnd[1]
          k02 = k_rnd[2]
```

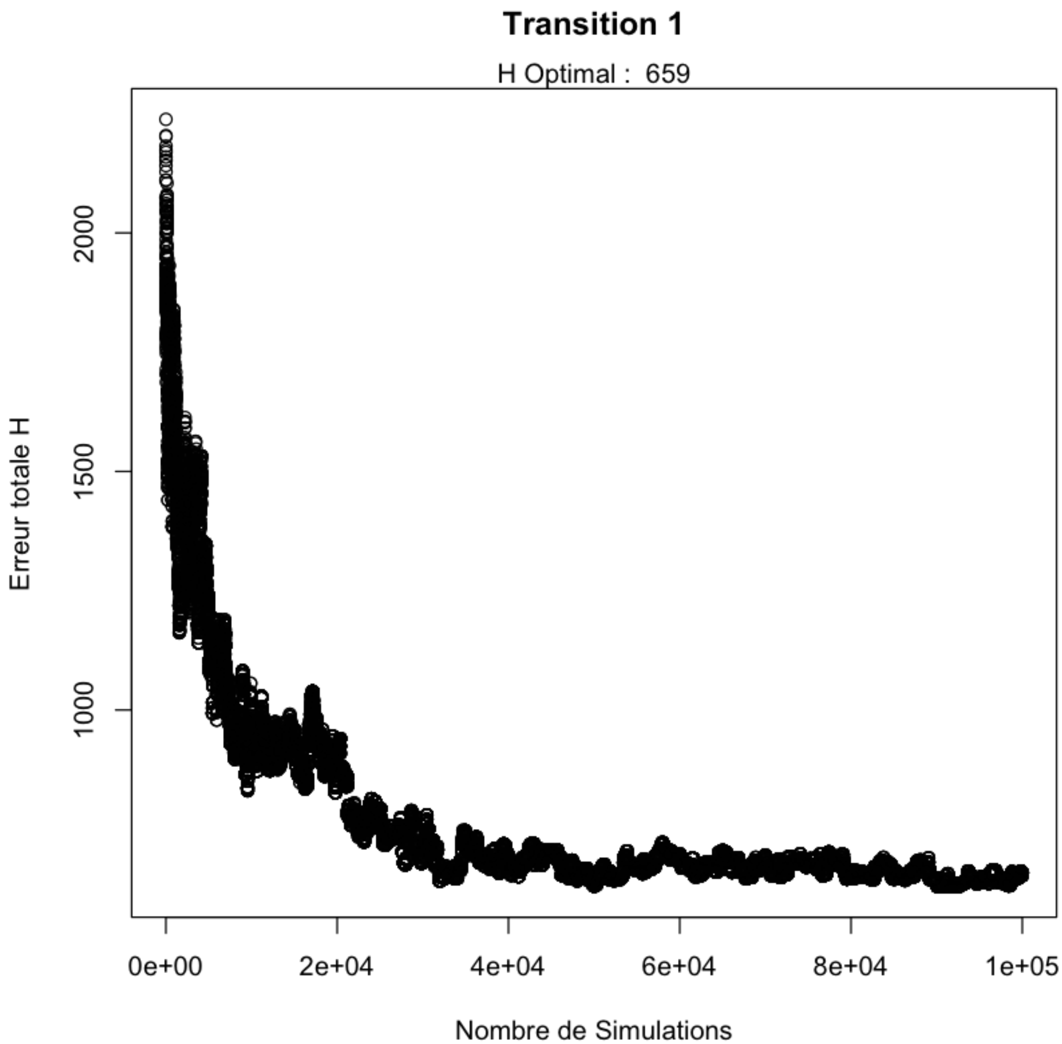
```
  current_vector = gnew[k01:k02]
  reversed_vector = rev(current_vector)
  gnew[k01:k02] = reversed_vector

  current_error = compute_error(G)
  new_error = compute_error(gnew)

  delta = new_error - current_error

  #alpha = min(exp(beta * delta), 1)
  # TEST
  alpha = min(exp(-beta * delta), 1)

  u = runif(1)

  #if (alpha < u)
  if (alpha > u)
  {
    return(gnew)
  }
  else
  {
    return(G)
  }
}
```

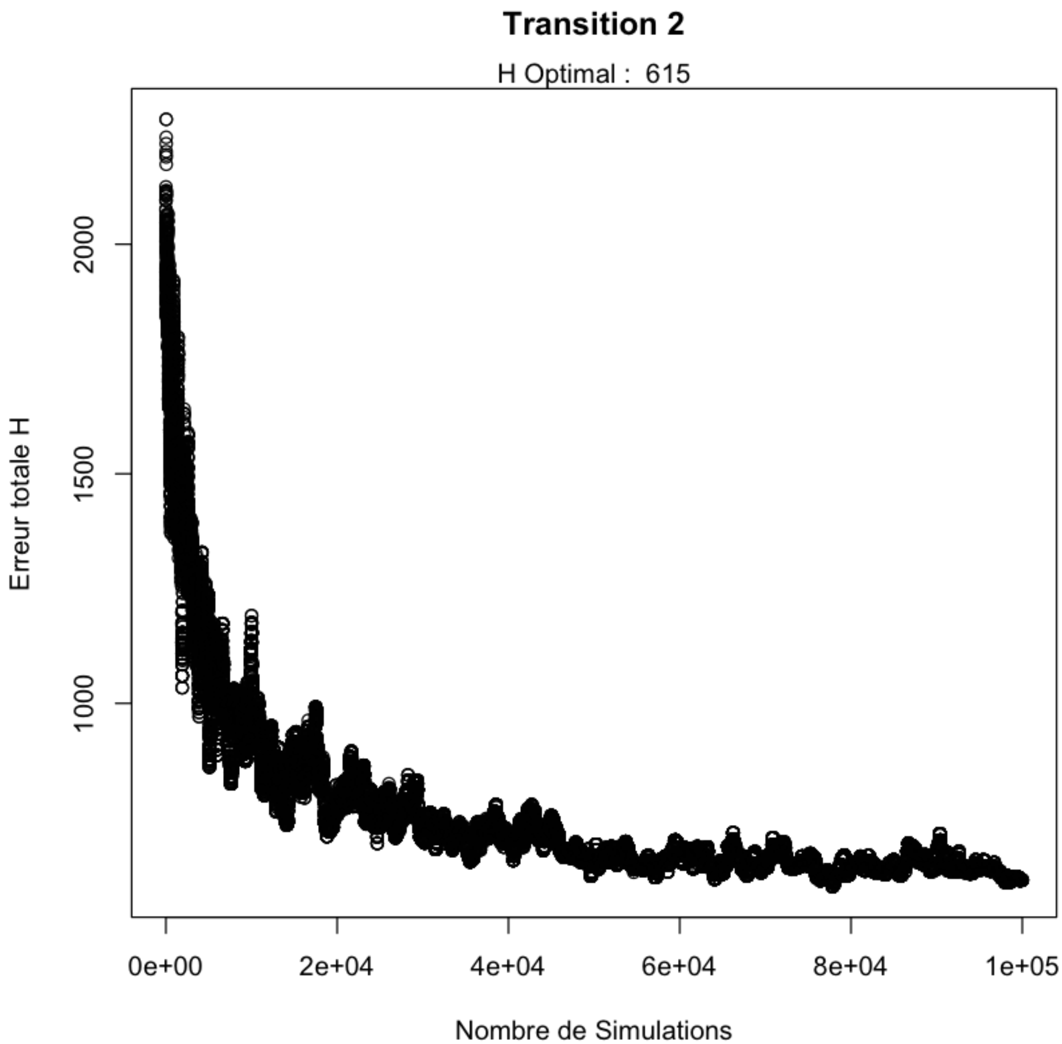
#### 3.0.5 Calcul du G optimal et affichage de l'erreur total H

In [6]:
```
compute_G = function(G, beta, num_method)
{
  transitions = c(transition1, transition2)
  transition = transitions[[num_method]]
  H = rep(0, nmax)
  H[1] = compute_error(G)

  for (i in 1:nmax)
  {
    G = transition(G, beta)
    H[i+1] = compute_error(G)
    beta = beta_0 * sqrt(i)
  }

  return(list("G"= G, "H"= H))
}
```

In [7]:
```
beta_0 = 0.001
nmax = 100000
##Initialisation du trajet --> On passe par chaque ville dans l'ordre d'indexation
```

```r
G0=1:N

# Méthode Transition 1
# Initialissation aleatoire des niveaux de gris des departements

result = compute_G(G0, beta_0, 1)

G_opt1 = unlist(result["G"])
H_opt1 = unlist(result["H"])

plot(H_opt1,
     main="Transition 1",
     xlab="Nombre de Simulations",
     ylab="Erreur totale H")
mtext(paste("H Optimal : ", H_opt1[nmax]))

# Méthode Transition 2
# Initialissation aleatoire des niveaux de gris des departements

result = compute_G(G0, beta_0, 2)

G_opt2 = unlist(result["G"])
H_opt2 = unlist(result["H"])

plot(H_opt2,
     main="Transition 2",
     xlab="Nombre de Simulations",
     ylab="Erreur totale H")
mtext(paste("H Optimal : ", H_opt2[nmax]))
```


Transition 1
H Optimal : 659
Erreur totale H
2000
1500
1000
0e+00
2e+04
4e+04
6e+04
8e+04
1e+05
Nombre de Simulations




### 3.0.6 Affichage de la carte

In [8]:
```
## Trace de la carte de France
frontiere <- array(0,dim=c(93,2))

frontiere[1,]<-c(-1.890011344864835, 43.39994027717378)
frontiere[2,]<-c(-1.526088473383384, 43.46652614992962)
frontiere[3,]<-c(-1.203687377362188, 44.56128139764979)
frontiere[4,]<-c(-1.163166811337313, 45.52731326071159)
frontiere[5,]<-c(-0.6736004848015615, 45.04545630005487)
frontiere[6,]<-c(-0.7066948874486197, 45.32945421754)
frontiere[7,]<-c(-1.118203419105098, 45.62396892639908)
frontiere[8,]<-c(-1.105714715458872, 45.94493198008892)
```

```
frontiere[9,]<-c(-1.12838239609037, 46.26455005881584)
frontiere[10,]<-c(-1.6645304311116, 46.43530693449198)
frontiere[11,]<-c(-2.088893784042677, 46.84723574177161)
frontiere[12,]<-c(-2.073119073683606, 47.05048478216584)
frontiere[13,]<-c(-2.504608707324863, 47.41535164398242)
frontiere[14,]<-c(-4.037633513427355, 47.83792429729075)
frontiere[15,]<-c(-4.254849703174842, 47.80261641004851)
frontiere[16,]<-c(-4.645123541547223, 48.0242045085484)
frontiere[17,]<-c(-4.331355831033263, 48.05282984881239)
frontiere[18,]<-c(-4.488867816785802, 48.21453280425273)
frontiere[19,]<-c(-4.306560642090731, 48.24810156186573)
frontiere[20,]<-c(-4.340892758039394, 48.34350662170984)
frontiere[21,]<-c(-4.694048576126732, 48.31323817501202)
frontiere[22,]<-c(-4.725527732010061, 48.49427630181697)
frontiere[23,]<-c(-4.018198196631981, 48.67506746252908)
frontiere[24,]<-c(-3.642873987186096, 48.67979462275454)
frontiere[25,]<-c(-3.566124212239391, 48.79243751363897)
frontiere[26,]<-c(-3.104118485435778, 48.83571472087023)
frontiere[27,]<-c(-2.674218125158933, 48.53771138275979)
frontiere[28,]<-c(-2.352572190734006, 48.62188260125732)
frontiere[29,]<-c(-1.546511804018612, 48.63608706535828)
frontiere[30,]<-c(-1.575065288393238, 49.26317482005418)
frontiere[31,]<-c(-1.92568701686213, 49.7155241070171)
frontiere[32,]<-c(-1.600292703431177, 49.66362701893297)
frontiere[33,]<-c(-1.228254313376111, 49.685464046314)
frontiere[34,]<-c(-1.232046250908218, 49.53935500087057)
frontiere[35,]<-c(-1.101752196986852, 49.34843695552533)
frontiere[36,]<-c(-0.2825782177554256, 49.2579067570399)
frontiere[37,]<-c(0.4773543128856985, 49.42809916390489)
frontiere[38,]<-c(0.1253004614904157, 49.47090909815103)
frontiere[39,]<-c(0.2208001384982297, 49.70276214800806)
frontiere[40,]<-c(1.224354325238743, 50.00722197783535)
frontiere[41,]<-c(1.639721900869359, 50.38926155112484)
frontiere[42,]<-c(1.642664904961495, 50.89415356758433)
frontiere[43,]<-c(2.571993635011597, 51.0338944589171)
frontiere[44,]<-c(2.57332611518341, 50.82346702706909)
frontiere[45,]<-c(2.913760909892759, 50.68788879417939)
frontiere[46,]<-c(3.103244146802385, 50.78631626136612)
frontiere[47,]<-c(3.659733848071119, 50.30529841369466)
frontiere[48,]<-c(4.016034143556764, 50.33988863284699)
frontiere[49,]<-c(4.175178735191895, 50.08210320100684)
frontiere[50,]<-c(4.616530720615451, 49.96826023474988)
frontiere[51,]<-c(4.795025965615682, 50.18583431157266)
frontiere[52,]<-c(4.827977702557021, 49.74699255642302)
frontiere[53,]<-c(5.425392647421083, 49.54439130531281)
frontiere[54,]<-c(5.739730685124953, 49.55103807649246)
frontiere[55,]<-c(6.06729438912769, 49.4840577316069)
frontiere[56,]<-c(6.491281445078217, 49.46264377200747)
```

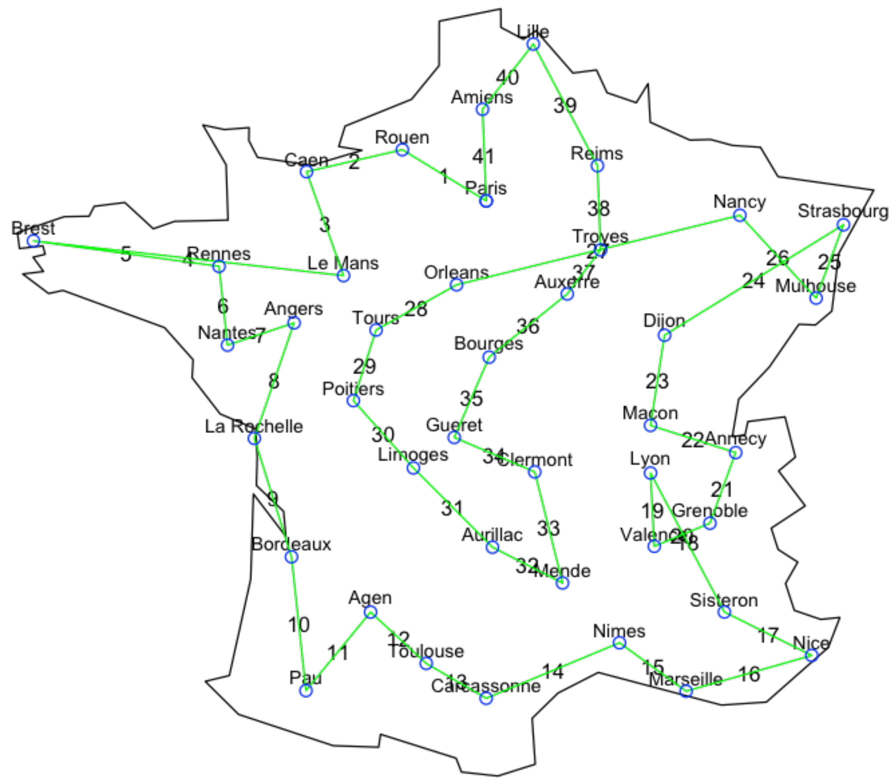
```
frontiere[57,]<-c(6.75571327386961, 49.12927322687535)
frontiere[58,]<-c(8.201796478649424, 48.97591791225366)
frontiere[59,]<-c(7.673676139531496, 48.25851563893939)
frontiere[60,]<-c(7.56845146508074, 47.60344963683364)
frontiere[61,]<-c(7.32024367405695, 47.44513955544842)
frontiere[62,]<-c(7.071360242064299, 47.45358049632104)
frontiere[63,]<-c(6.62451807999329, 46.96823803276949)
frontiere[64,]<-c(6.162029217094378, 46.60231996612718)
frontiere[65,]<-c(6.066610775760656, 46.17856306229678)
frontiere[66,]<-c(6.257498364113462, 46.19666431426239)
frontiere[67,]<-c(6.301872403502051, 46.34938687654174)
frontiere[68,]<-c(6.861463339274339, 46.40245337472187)
frontiere[69,]<-c(7.017929770616087, 45.94556231010345)
frontiere[70,]<-c(6.764943836752621, 45.78857058089388)
frontiere[71,]<-c(7.166601474155536, 45.39282263517097)
frontiere[72,]<-c(6.653761799998819, 45.13911105218517)
frontiere[73,]<-c(6.757651740197312, 44.89710726010914)
frontiere[74,]<-c(7.054043217502413, 44.75396034087427)
frontiere[75,]<-c(6.869325931607312, 44.51087114631039)
frontiere[76,]<-c(7.002616551493348, 44.20607287444226)
frontiere[77,]<-c(7.400805105055442, 44.10380889958807)
frontiere[78,]<-c(7.689753826772605, 44.12552535018468)
frontiere[79,]<-c(7.515061875183162, 43.7852964894461)
frontiere[80,]<-c(6.579999636832232, 43.16489068063933)
frontiere[81,]<-c(5.902270950224193, 43.13098616039171)
frontiere[82,]<-c(4.043101528472381, 43.50153398287527)
frontiere[83,]<-c(3.436014403081625, 43.2677934567814)
frontiere[84,]<-c(3.044082836994732, 42.98072339226199)
frontiere[85,]<-c(3.091056730618972, 42.45969888755423)
frontiere[86,]<-c(2.526608645753571, 42.32581477421949)
frontiere[87,]<-c(1.978104539472978, 42.37088197097692)
frontiere[88,]<-c(1.895870371966853, 42.52820887653542)
frontiere[89,]<-c(0.7560594524244203, 42.79943908951059)
frontiere[90,]<-c(0.7269731355005962, 42.65314077026987)
frontiere[91,]<-c(0.04248034029632037, 42.66991843927905)
frontiere[92,]<-c(-1.386126483719303, 43.02508241329305)
frontiere[93,]<-c(-1.890011344864835, 43.39994027717378)

## Localisation des villes
position <- array(0,dim=c(N+2,2))
position[1,]<-c(2.351805180704266, 48.85400969127046)
position[N+2,]<-c(2.351805180704266, 48.85400969127046)
position[2,]<-c(3.083998339641749, 45.77768075368159)
position[3,]<-c(0.3459766164476508, 46.58727225103882)
position[4,]<-c(1.868830388128449, 46.16906317274702)
position[5,]<-c(5.043326151926627, 47.3293717391258)
position[6,]<-c(4.831812627327309, 46.30566433896024)
position[7,]<-c(0.6891061057189397, 47.3882990242959)
```

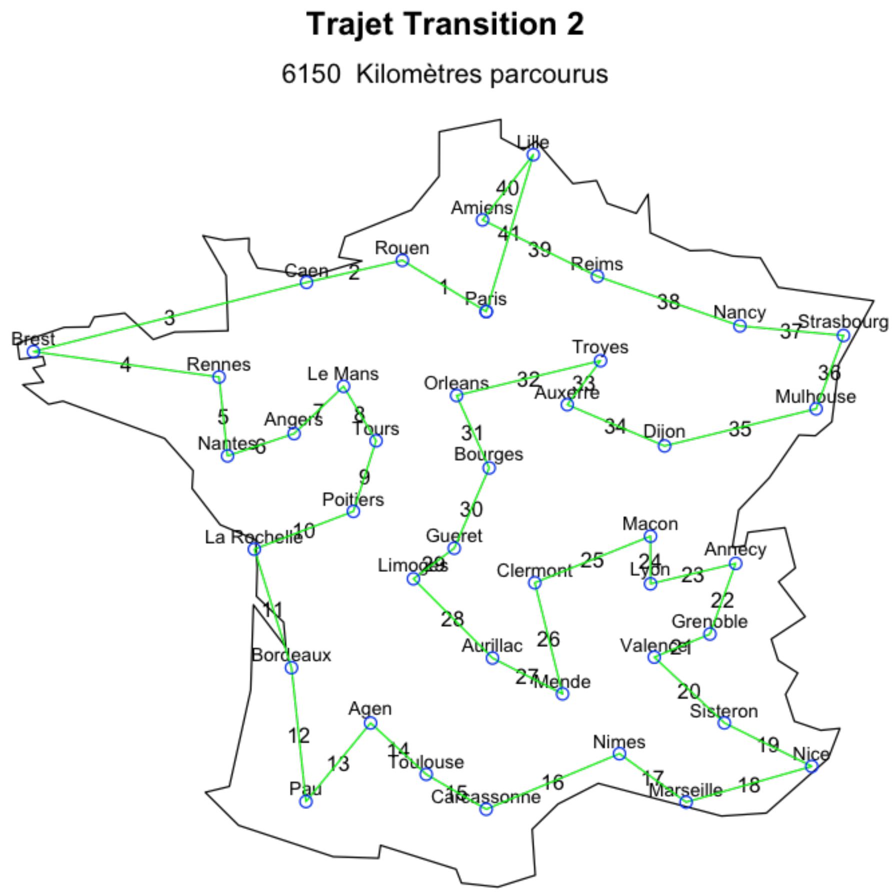
```r
position[8,]<-c(0.1977390085072901, 48.00585356747929)
position[9,]<-c(3.575159833022831, 47.79764093593785)
position[10,]<-c(-0.5487355455807347, 47.46921855555196)
position[11,]<-c(-1.553835556601499, 47.21645709538476)
position[12,]<-c(-0.5880694804310389, 44.81346413012534)
position[13,]<-c(2.39608025982174, 47.08025796745675)
position[14,]<-c(-0.3610639287786248, 49.18501591578796)
position[15,]<-c(4.07734043442664, 48.29643496506808)
position[16,]<-c(-1.681486463481057, 48.11161801319862)
position[17,]<-c(1.904310346007765, 47.90183788966329)
position[18,]<-c(4.831844579009798, 45.76537622222675)
position[19,]<-c(6.180761941163706, 48.6911991567444)
position[20,]<-c(7.74454877733517, 48.58312694472249)
position[21,]<-c(1.445872778203823, 43.60435748484851)
position[22,]<-c(3.062386996469666, 50.63446933632193)
position[23,]<-c(5.728730310031624, 45.19549293594445)
position[24,]<-c(5.368670, 43.288566)
position[25,]<-c(2.351024, 43.208041)
position[26,]<-c(4.030531, 49.253933)
position[27,]<-c(2.446033, 44.922935)
position[28,]<-c(2.295631455224776, 49.89397910222291)
position[29,]<-c(4.363240651994897, 43.83825200968894)
position[30,]<-c(-1.148867197790304, 46.1580868456743)
position[31,]<-c(-0.3697895951897412, 43.2951664398616)
position[32,]<-c(3.503532370791358, 44.51696519841824)
position[33,]<-c(7.26579664737482, 43.69476935505691)
position[34,]<-c(1.252155, 45.82246)
position[35,]<-c(7.331065, 47.750431)
position[36,]<-c(0.605669, 44.18648)
position[37,]<-c(1.085770, 49.435238)
position[38,]<-c(4.8883, 44.9333)
position[39,]<-c(6.1170, 45.9966)
position[40,]<-c(-4.48333, 48.4)
position[41,]<-c(5.94623, 44.18758)

## Nom des villes
P=c('Paris','Clermont','Poitiers','Gueret','Dijon','Macon','Tours',
'Le Mans','Auxerre','Angers','Nantes','Bordeaux','Bourges','Caen',
'Troyes','Rennes','Orleans','Lyon','Nancy','Strasbourg', 'Toulouse',
'Lille', 'Grenoble', 'Marseille', 'Carcassonne', 'Reims', 'Aurillac',
'Amiens', 'Nimes', 'La Rochelle', 'Pau', 'Mende', 'Nice', 'Limoges',
'Mulhouse', 'Agen', 'Rouen', 'Valence', 'Annecy','Brest','Sisteron')

## Trace du trajet G_opt1
coord_trajet=position
for (i in 1:N)
  #{coord_trajet[i+1,]=position[G_best[i]+1,]}
  {coord_trajet[i+1,]=position[G_opt1[i]+1,]}
```

```
#x11()
plot(frontiere[,1],frontiere[,2],type="l", bg="grey",col="black", xlab="", ylab="", axe
mtext(paste(H_opt1[nmax] * 10, " Kilomètres parcourus"))

for (i in 1:(N+1))
  {
  text(position[i,1],position[i,2]+0.15,P[i], cex=0.7)
  x=(coord_trajet[i,1]+coord_trajet[i+1,1])/2
  y=(coord_trajet[i,2]+coord_trajet[i+1,2])/2
  text(x,y,i,cex=0.8)        }

lines(coord_trajet[,1],coord_trajet[,2],type="l", col="green")
lines(coord_trajet[,1],coord_trajet[,2],type="p", col="blue")

## Trace du trajet G_opt2

coord_trajet=position
for (i in 1:N)
  #{coord_trajet[i+1,]=position[G_best[i]+1,]}
{coord_trajet[i+1,]=position[G_opt2[i]+1,]}

plot(frontiere[,1],frontiere[,2],type="l", bg="grey",col="black", xlab="", ylab="", axe
mtext(paste(H_opt2[nmax] * 10, " Kilomètres parcourus"))

for (i in 1:(N+1))
{
  text(position[i,1],position[i,2]+0.15,P[i], cex=0.7)
  x=(coord_trajet[i,1]+coord_trajet[i+1,1])/2
  y=(coord_trajet[i,2]+coord_trajet[i+1,2])/2
  text(x,y,i,cex=0.8)        }

lines(coord_trajet[,1],coord_trajet[,2],type="l", col="green")
lines(coord_trajet[,1],coord_trajet[,2],type="p", col="blue")
```

**Trajet Transition 1**

6590 Kilomètres parcourus






# 4 Problème de la Clique Maximum

On dispose d'un Graphe aléatoire G, constitué d'un ensemble de N points, reliés aléatoirements les uns aux autres, chaque point étant reliés au plus une fois avec un autre point (pas de double lien). Une clique, dans un graphe, est un ensemble de point, tous reliés les uns aux autres. Dans une clique de 4 élements, chaque point de la clique est reliés aux 3 autres. On veut connaitre la Clique maximum de ce graphe G. Ce problème est NP-complet, car on peut le réduire à un Problème SAT, connu comme étant NP-complet.

- On vas créer une matrice X de dimension (d, 2).
- chaque ligne représente un lien entre un point et un autre du graphe
- d est le nombre total de liens entre les points.

- d est supérieur à 0, et au maximum égal à N!/2!(N-2)!, qui correspond au cas où chaque point du graphe sont reliés les unx aux autres, et la clique maximum est donc evidemment égal à N.
- Enfaite, d = 2 * N!/2!(N-2)! = N!/(N-2)!, car dans notre tableau, un lien est écrit deux fois, par exemple le lien (2, 4) ci dessous, apparait aussi sur la ligne de (4, 2)...
- Dans la pratique, on définira d comme au moins égal à N
- La 1ère colonne contient l'identifiant i du point, i = 1, . . . , N
- la 2ème colonne indique l'identifiant j du point reliés, j = 1, . . . , N
- la matrice X est donc, par exemple, et pour N = 12 de la forme :

```
In [1]: #|   id   |   lien   |
         #_____________________
         #|   1    |    3     |
         #|   1    |    8     |
         #|   1    |    12    |
         #|   2    |    4     |
         #|   2    |    9     |
         #|   3    |    1     |
         #|   4    |    2     |
         #|   4    |    6     |
         #|   4    |    11    |
         #|  ...   |   ...    |
         #|   12   |    1     |
         #|   12   |    6     |
         #_____________________
```

#### 4.0.1 Simulation du graphe aléatoire

```
In [2]: Nb_pts = 17
        max_rnd_link = Nb_pts/40 * factorial(Nb_pts) / (2*factorial(Nb_pts-2))
        min_rnd_link = 3 * Nb_pts
        nb_links = sample(min_rnd_link:max_rnd_link, 1)
        ID = rep(0, nb_links)
        LINK = rep(0, nb_links)
        X_first = rep(0, nb_links)

        for (i in 1:nb_links)
        {
            rnd_id1 = sample(1:Nb_pts, 1)
            rnd_id2 = sample(1:Nb_pts, 1)
            ID[i] = rnd_id1
            LINK[i] = rnd_id2
        }

        # On trie le vecteur d'ID
        ID = sort(ID)

        # On merge les vecteurs ID et LINK
```

```r
X_first = cbind(ID, LINK)

# A chaque ligne (x, j), on ajoute au vecteur X la ligne (j, x)
#(en effet, si, par exemple, le point 1 est lié au point 4, on a la ligne (1,4) ... et
# la ligne (4, 1) ! )
lenX = length(X_first[, 1])
X_extend = rep(1, lenX)
X_extend = cbind(X_extend, X_extend)
for (i in 1:lenX)
{
    row = X_first[i, ]
    X_extend[i, ] = c(row[2], row[1])
}

# On trie X_extend
X_extend = X_extend[order(X_extend[, 1]), ]

# On supprime les lignes en doublon
X_first = X_first[!duplicated(X_first), ]
X_extend = X_extend[!duplicated(X_extend), ]

# On superpose X et X_extend afin d'obtenir la matrix finale
X = rbind(X_first, X_extend)

# On supprime les lignes en doublon de la matrice finale
X = X[!duplicated(X), ]

# On trie la matrice finale
X = X[order(X[, 1]), ]

# On suprime les lignes de la forme (i, j) où i = j
X = X[!(X[, 1] == X[, 2]), ]

head(X, 10)
```

| ID | LINK |
|---|---|
| 1 | 8 |
| 1 | 17 |
| 1 | 15 |
| 1 | 9 |
| 1 | 2 |
| 1 | 5 |
| 1 | 6 |
| 2 | 13 |
| 2 | 14 |
| 2 | 10 |

- Construction de la matrice de voisinage

```
In [3]: df = data.frame(matrix(0, nrow=Nb_pts, ncol=Nb_pts))

        for (i in 1:length(X[, 1]))
        {
            line_axis = X[i, 1]
            column_axis  = X[i, 2]

            df[line_axis, column_axis] = 1
        }

        rownames(df) = colnames(df)

        head(df, length(df[1, ]))
```

| | X1 | X2 | X3 | X4 | X5 | X6 | X7 | X8 | X9 | X10 | X11 | X12 | X13 | X14 | X15 | X16 | X17 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| X1 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 |
| X2 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 |
| X3 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 1 |
| X4 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| X5 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 0 |
| X6 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| X7 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 1 |
| X8 | 1 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 |
| X9 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| X10 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 1 | 0 | 1 |
| X11 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| X12 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| X13 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| X14 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| X15 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| X16 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 |
| X17 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 0 |

```
In [4]: if (!isSymmetric(as.matrix(df)))
        {
            print("There's a problem with the simulated link table : it's not symmetric, wherea
        } else
        {
            print("Simulation is OK")
        }

[1] "Simulation is OK"
```

### 4.0.2 Graphe avec une clique de taille maximum N fixée

```
In [2]: Nb_pts = 40
        range_pts = 1:Nb_pts
        desired_clique_size = 10
```

```
range_desired_clique = sample(1:Nb_pts, desired_clique_size, replace=FALSE)

clique = expand.grid(range_desired_clique, range_desired_clique)

# Remove link to self : Point 1 --> Point 1
good_links = rownames(clique[which(!(clique[,1] == clique[, 2])),])
clique = clique[good_links,]

random_links = data.frame()

for (i in seq(1, (45*desired_clique_size), by=2))
{
  already_picked=list()
  rnd_id1 = sample(1:Nb_pts, 1)
  rnd_id2 = sample(1:Nb_pts, 1)
  while(is.element(rnd_id2, range_desired_clique) || rnd_id1==rnd_id2){
       rnd_id2 = sample(1:Nb_pts, 1)
  }
  already_picked= c(already_picked,rnd_id2)
  #print(already_picked)
  random_links[i, 1] = rnd_id1
  random_links[i, 2] = rnd_id2

  random_links[i + 1, 1] = rnd_id2
  random_links[i + 1, 2] = rnd_id1
}


colnames(random_links) = colnames(clique)

X = rbind(clique, random_links)

X = X[order(X["Var1"]), ]
X=unique(X)
rownames(X) = 1:length(X[, 1])
print(head(X))
#head(X, 10)
```

```
  Var1 Var2
1    1   10
2    1   17
3    1   20
4    1   11
5    1   15
6    1   22
```

- Construction de la matrice de voisinage

```
In [3]: df = data.frame(matrix(0, nrow=Nb_pts, ncol=Nb_pts))

        for (i in 1:length(X[, 1]))
        {
            line_axis = X[i, 1]
            column_axis  = X[i, 2]

            df[line_axis, column_axis] = 1
        }

        rownames(df) = colnames(df)

        head(df, 10)
```

| | X1 | X2 | X3 | X4 | X5 | X6 | X7 | X8 | X9 | X10 | ... | X31 | X32 | X33 | X34 | X35 | X36 | X37 | X |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| X1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | ... | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| X2 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 0 | 0 | ... | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| X3 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | ... | 1 | 1 | 1 | 0 | 1 | 0 | 1 | 1 |
| X4 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | ... | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| X5 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | ... | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| X6 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | ... | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 |
| X7 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | ... | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 1 |
| X8 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | ... | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| X9 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | ... | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 1 |
| X10 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | ... | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |

```
In [4]: if (!isSymmetric(as.matrix(df)))
        {
            print("There's a problem with the simulated link table : it's not symmetric, wherea
        } else
        {
            print("Simulation is OK")
        }
```

```
[1] "Simulation is OK"
```

#### 4.0.3 Algorithme de Metropolis pour trouver la Clique Maximum

… Algorithme de metroplois donnant le vecteur clique_star, composé des id des points de l'ensembe composant la clique maximum. On construit à partir de ce vecteur max_clique, composé des id de tous les points du graph, et un 1 pour ceux faisant partie de la clique, 0 sinon

- Initialisation

```
In [5]: ## Nombre d'iterations de l'algorithme
        nmax = 500
```

- Fonction d'erreur H

In [6]:
```
# L'erreur H vas être la somme de lien entre les points de l'ensemble
# On vas donc vouloir maximiser H
# (maximiser le nombre de liens entre les points de l'ensemble candidat)
# H vaut donc, au plus :
#    - si l'ordre ne compte pas (la ligne (i, j) et (j, i) sont les mêmes):
#        max de H = fact(N)/(2*fact(N-2))
#    - si l'ordre compte pas (la ligne (i, j) et (j, i) ne sont pas les mêmes):
#        max de H = fact(N)/(fact(N-2))

compute_error = function(G, links_table)
{
    H = 0
    for (i in 1:length(links_table[, 1]))
    {
        if (G[links_table[i, 1]] == 1 & G[links_table[i, 2]] == 1)
        {
            H = H + 1
        }
    }
    return(H)
}
```

- Fonction de transition 1

In [7]:
```
transition1 = function(g, H, links_table, beta)
{
  # g est l'ensemble candidat clique maximum à K élements
  gnew = g
  clique_sz = sum(g)
  max_link = factorial(clique_sz)/factorial(clique_sz-2)
  #clique_elements = g[find(g == 1)]
  clique_elements = which(g == 1)
  non_clique_elements = which(g == 0)
  k0 = sample(clique_elements, 1)
  k1 = sample(non_clique_elements, 1)
  gnew[k0] = 0
  gnew[k1] = 1

  # On calcul H (ou delta) : c'est la différence entre les anciennes et les nouvelles
  delta = compute_error(gnew, links_table) - compute_error(g, links_table)
  alpha = max(min(exp(beta * (delta/max_link - 1)), 1), 0)

  u = runif(1)

  if (u < alpha)
  {
    # on accepte la transition
    result = c(gnew, compute_error(gnew, links_table))
```

```
    }
    else
    {
      # on rejette la transition
      result = c(g, H)
    }

    return(result)
  }
```

- Fonction de transition 2

```
In [8]: transition2 = function(g, H, links_table, beta)
  {
    gnew = g
    clique_sz = sum(g)
    max_link = factorial(clique_sz)/factorial(clique_sz-2)
    if (max_link>200){
        print(g)
        print(max_link)
    }
    current_error = compute_error(g, links_table)

    clique_elements = which(g == 1)
    k0 = sample(clique_elements, 1)
    gnew[k0] = 0

    all_probas = rep(0, Nb_pts)
    for (i_j in 1:Nb_pts)
    {
      if (!(g[i_j] == 1))
      {
          gnew = g
          gnew[k0] = 0
          gnew[i_j] = 1
          delta_num = 0
          delta = compute_error(gnew, links_table) - current_error
          alpha = max(min(exp(beta * (delta/max_link - 1)), 1), 0)

          all_probas[i_j] = alpha
      }
      else
      {
          all_probas[i_j] = 0
      }

    }
```

```
    best_point = which.max(all_probas)

    gnew = g
    gnew[k0] = 0
    gnew[best_point] = 1

    Hnew = compute_error(gnew, links_table)

    return(c(gnew, Hnew))
  }
```

- L'algorithme de Metropolis : Calcul du G optimal

```
In [9]: compute_G = function(G, nmax_simul, num_method, links_table)
        {
            transition_method = c(transition1, transition2)
            clique_sz = sum(G)
            max_link = factorial(clique_sz)/factorial(clique_sz-2)
            nb_iteration=0
            stopifnot(match(num_method, 1:4) > 0)

            H = rep(0, nmax_simul)
            H[1] = compute_error(G, links_table)

            beta_fixed = 1
            beta_0 = 0.01
            for (i in 1:nmax_simul)
            {
                if (is.element(num_method, c(3, 4)))
                {
                    result = transition_method[[num_method - 2]](G, H[i], links_table, beta_0
                }
                else
                {
                    result = transition_method[[num_method]](G, H[i], links_table, beta_fixed)
                }

                #G = result[1:clique_sz]
                G = result[1:Nb_pts]
                H[i+1] = result[Nb_pts+1]
                nb_iteration=i

                if (H[i+1] == max_link)
                {
                    print(max_link)
                    H[i+2:length(H)] = H[i+1]

                    break
```

```
            }
            # TEST
            if (H[i+1] > max_link)
            {
                print(G)
                print(compute_error(G, links_table))
            }
        }

        return(list("G"= G, "H"= H,"nb_iter"=nb_iteration))
    }
```

- Boucle Principale

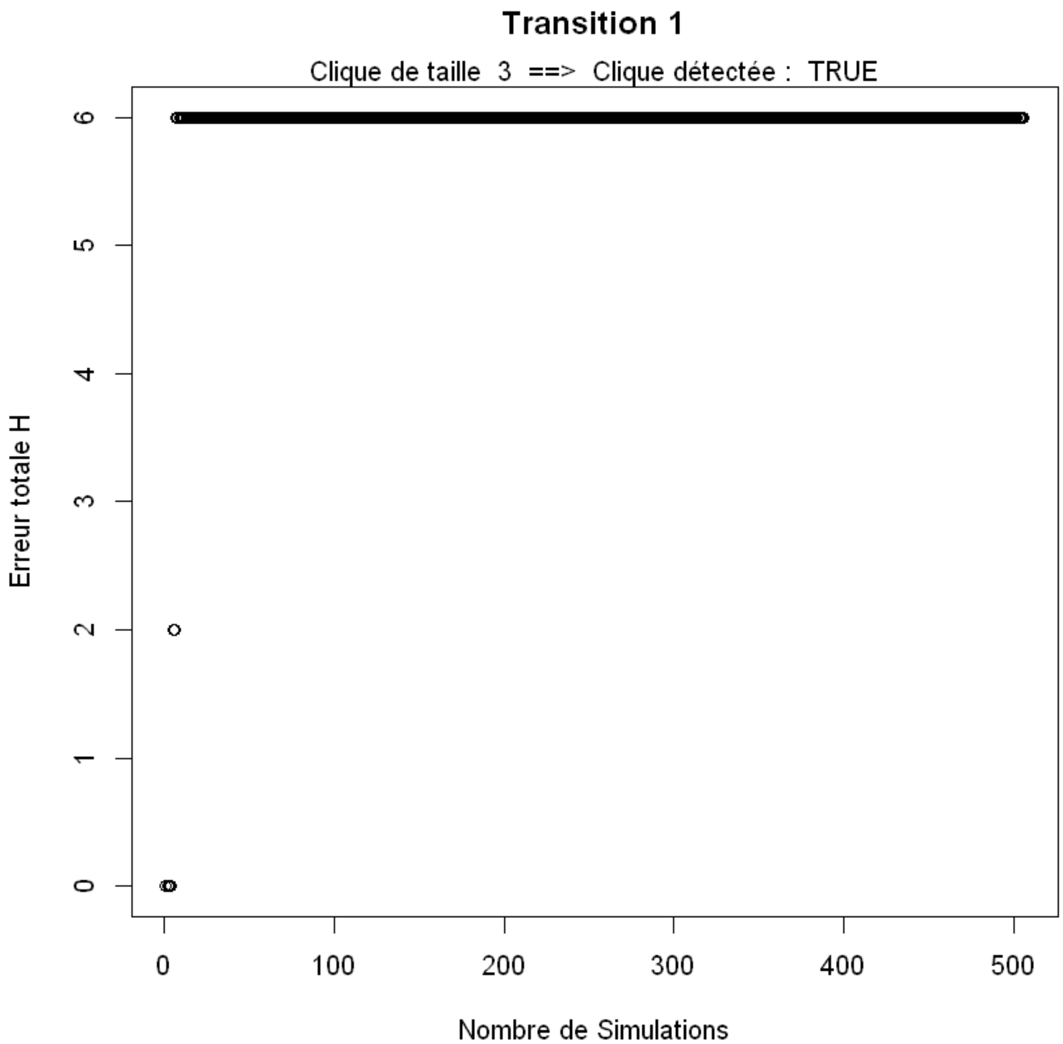
```
In [10]: old_warn_val = getOption("warn")
         options(warn = -1)

         start_clique_sz = 3

         nb_max_clique1 = 0
         nb_max_clique2 = 0
         G_opt1_list = list()
         H_opt1_list = list()
         G_opt2_list = list()
         H_opt2_list = list()
         G_opt3_list = list()
         H_opt3_list = list()
         G_opt4_list = list()
         H_opt4_list = list()
         i_row = 1
         clique1_is_def = FALSE
         clique2_is_def = FALSE
         clique3_is_def = FALSE
         clique4_is_def = FALSE
         nbr_iteration_list=list()
         G0 = rep(0, Nb_pts)
         #random_index = sample(1:Nb_pts, i_clique_sz, replace=FALSE)
         #G0[random_index] = 1
         #print(G0)
         for (i_clique_sz in start_clique_sz:Nb_pts)
         {
             # Méthode Transition 1
             # Initialissation aleatoire de G0
             #G0 = sample(1:Nb_pts,i_clique_sz,replace=FALSE)
             G0[1:i_clique_sz]=1
             #result = compute_G(G0, nmax, 1, X)
             result = compute_G(G0, nmax, 1, X)
             nb_iteration_transition1=result["nb_iter"]
```

```
G_opt1_list[i_row] = result["G"]
H_opt1_list[i_row] = result["H"]

H_opt1 = unlist(H_opt1_list[i_row])
clique_exists1 = H_opt1[nmax] == factorial(i_clique_sz)/factorial(i_clique_sz-2)

plot(H_opt1,
     main="Transition 1",
     xlab="Nombre de Simulations",
     ylab="Erreur totale H")
mtext(paste("Clique de taille ", i_clique_sz,
            " ==> ", "Clique détectée : ", clique_exists1))

if (!clique_exists1 && !clique1_is_def)
{
    nb_max_clique1 = i_clique_sz - 1
    clique1_is_def = TRUE
}

## Méthode Transition 2
## on enlève les warning car on en déclenche certains
old_warn_val = getOption("warn")
options(warn = -1)

#result = compute_G(G0, nmax, 2, X)
result = compute_G(G0, nmax, 2, X)
nb_iteration_transition2=result["nb_iter"]
G_opt2_list[i_row] = result["G"]
H_opt2_list[i_row] = result["H"]

H_opt2 = unlist(H_opt2_list[i_row])
clique_exists2 = H_opt2[nmax] == factorial(i_clique_sz)/factorial(i_clique_sz-2)

plot(H_opt2,
     main="Transition 2",
     xlab="Nombre de Simulations",
     ylab="Erreur totale H")
mtext(paste("Clique de taille ", i_clique_sz,
            " ==> ", "Clique détectée : ", clique_exists2))

## On remet les warnings
options(warn = old_warn_val)

if (!clique_exists2 && !clique2_is_def)
{
    nb_max_clique2 = i_clique_sz - 1
    clique2_is_def = TRUE
}
```

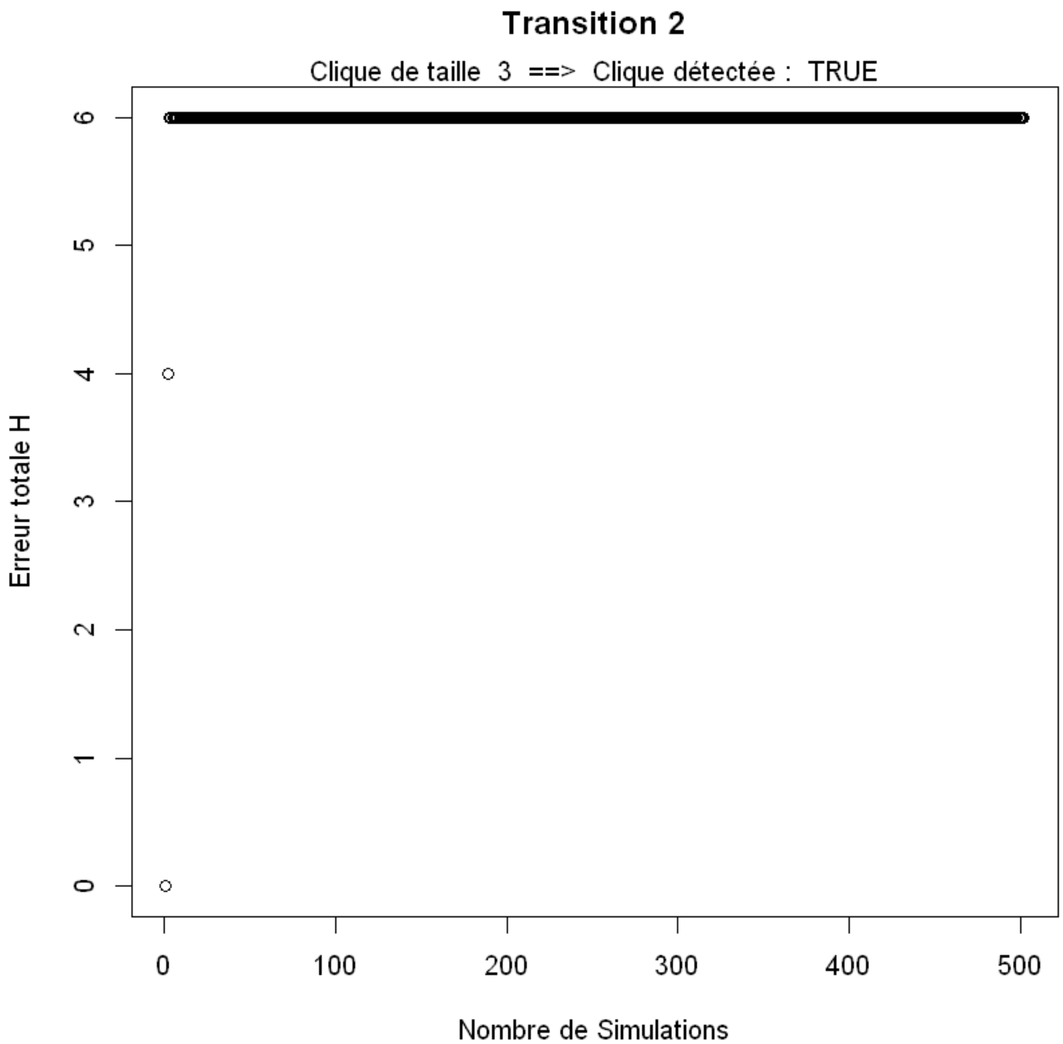
```
# Méthode Transition 3
#result = compute_G(G0, nmax, 3, X)
result = compute_G(G0, nmax, 3, X)
nb_iteration_transition3=result["nb_iter"]
G_opt3_list[i_row] = result["G"]
H_opt3_list[i_row] = result["H"]

H_opt3 = unlist(H_opt3_list[i_row])
clique_exists3 = H_opt3[nmax] == factorial(i_clique_sz)/factorial(i_clique_sz-2)

plot(H_opt3,
     main="Transition 3",
     xlab="Nombre de Simulations",
     ylab="Erreur totale H")
mtext(paste("Clique de taille ", i_clique_sz,
            " ==> ", "Clique détectée : ", clique_exists3))

if (!clique_exists3 && !clique3_is_def)
{
    nb_max_clique3 = i_clique_sz - 1
    clique3_is_def = TRUE
}

# Méthode Transition 4
#result = compute_G(G0, nmax, 4, X)
result = compute_G(G0, nmax, 4, X)
nb_iteration_transition4=result["nb_iter"]
G_opt4_list[i_row] = result["G"]
H_opt4_list[i_row] = result["H"]

H_opt4 = unlist(H_opt4_list[i_row])
clique_exists4 = H_opt4[nmax] == factorial(i_clique_sz)/factorial(i_clique_sz-2)

plot(H_opt4,
     main="Transition 4",
     xlab="Nombre de Simulations",
     ylab="Erreur totale H")
mtext(paste("Clique de taille ", i_clique_sz,
            " ==> ", "Clique détectée : ", clique_exists4))

if (!clique_exists4 && !clique4_is_def)
{
    nb_max_clique4 = i_clique_sz - 1
    clique4_is_def = TRUE
}

nbr_iteration_list[[i_row]]=list(i_row+2,nb_iteration_transition1,nb_iteration_tra
```

```
        #print(nbr_iteration_list[i_row])
        if (clique1_is_def && clique2_is_def && clique3_is_def && clique4_is_def)
        {
            break
        }

        i_row = i_row + 1
    }
    install.packages("xtable")
    library("xtable")
    output=NULL
    output = matrix(unlist(nbr_iteration_list), ncol = 5, byrow = TRUE)
    colnames(output)=c(" ","Transition 1", "Transition 2","Transition 3","Transition 4")

    options(warn = old_warn_val)
```

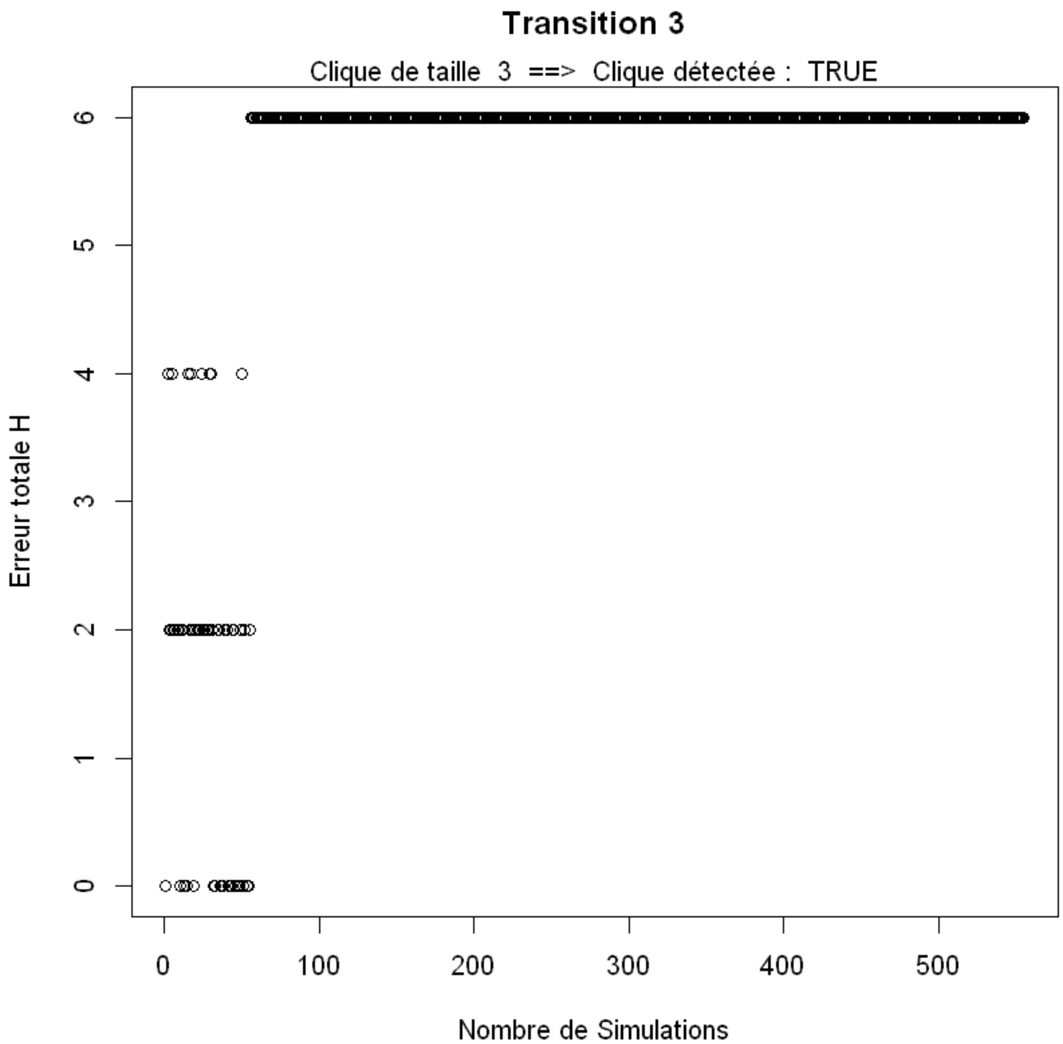
```
[1] 6
[1] 6
```

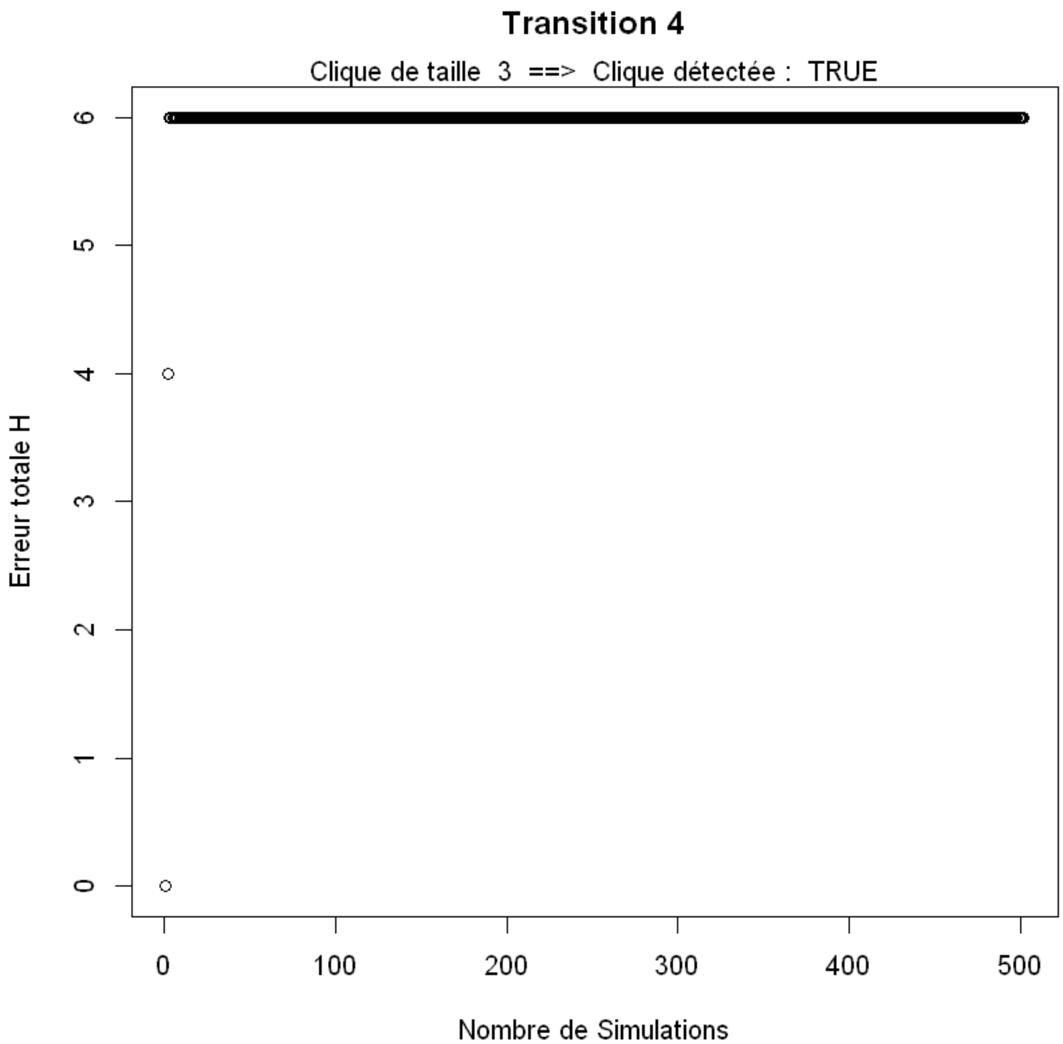

Transition 1
Clique de taille  3  ==>  Clique détectée :  TRUE
Erreur totale H
Nombre de Simulations
0
1
2
3
4
5
6
0
100
200
300
400
500

```
[1] 6
```

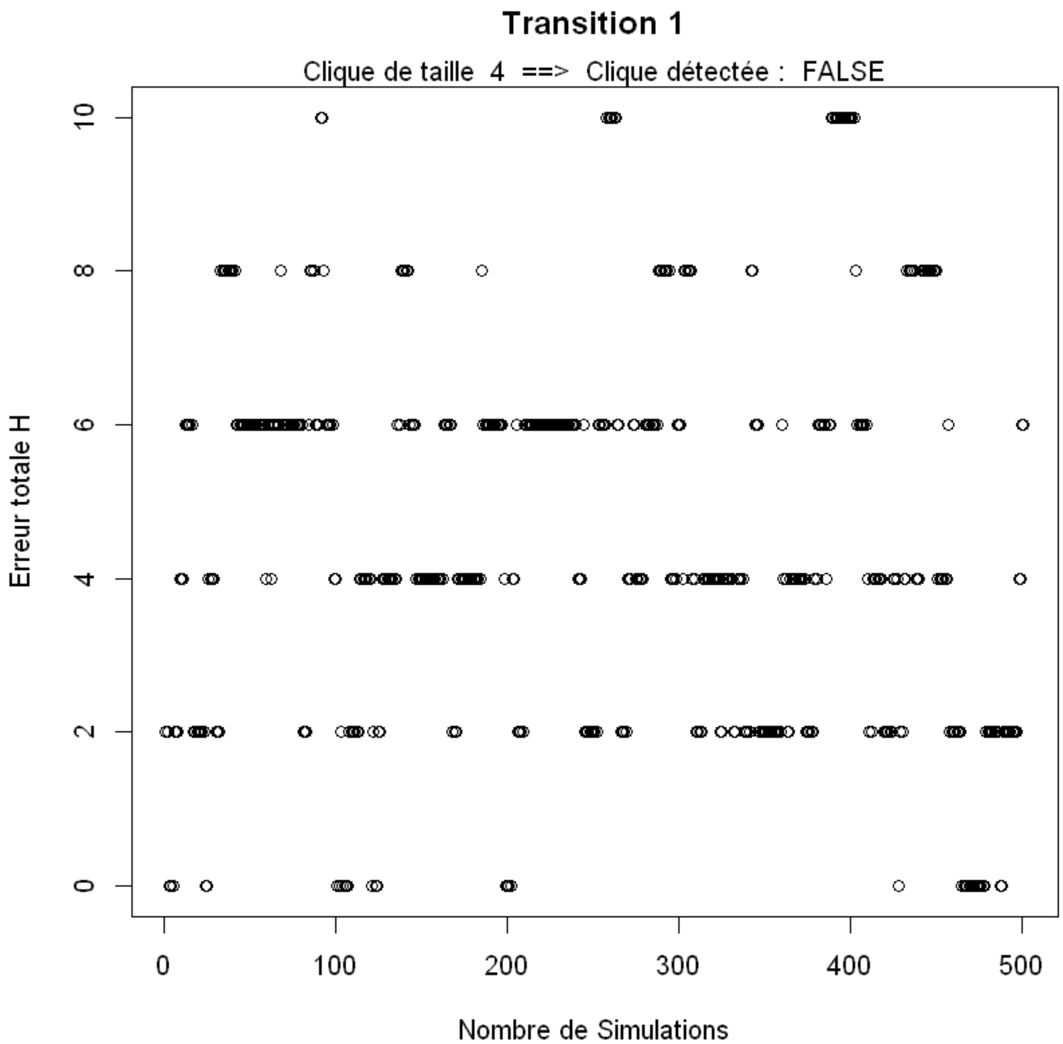

Transition 2
Clique de taille 3 ==> Clique détectée : TRUE
Erreur totale H
Nombre de Simulations
0
1
2
3
4
5
6
0
100
200
300
400
500

```
[1] 6
```

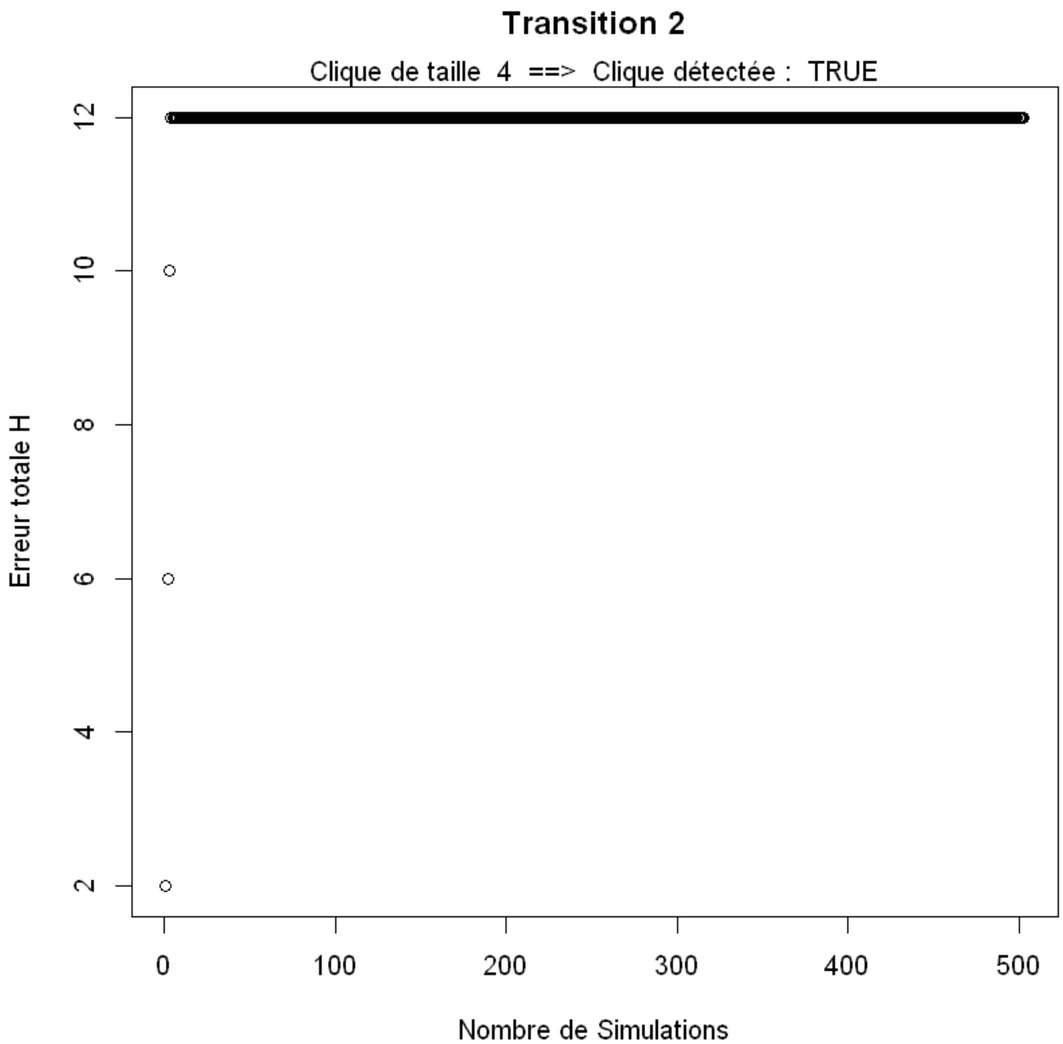

Transition 3
Clique de taille 3 ==> Clique détectée : TRUE
6
5
4
3
2
1
0
Erreur totale H
0
100
200
300
400
500
Nombre de Simulations


Transition 4
Clique de taille 3 ==> Clique détectée : TRUE
Erreur totale H
0
1
2
3
4
5
6
0
100
200
300
400
500
Nombre de Simulations

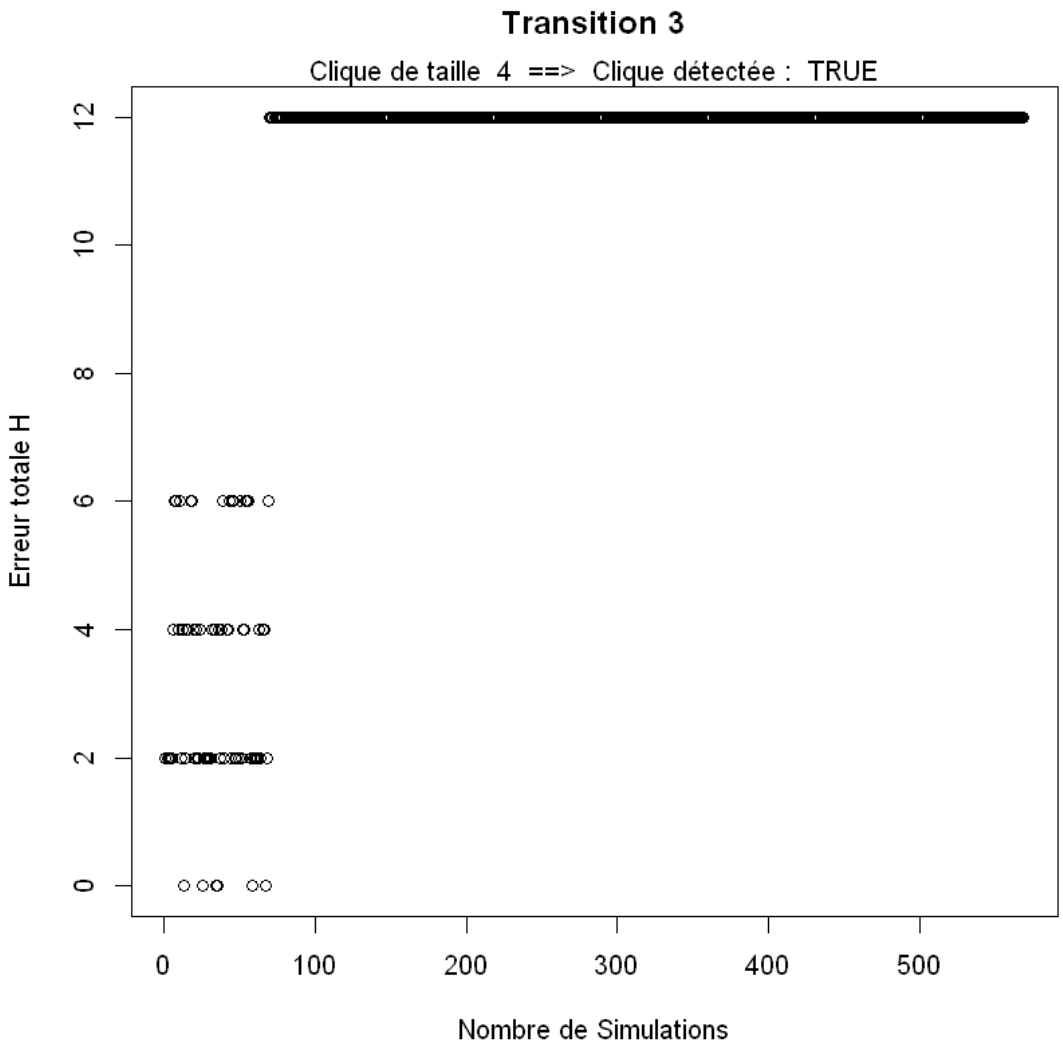
```
[1] 12
```

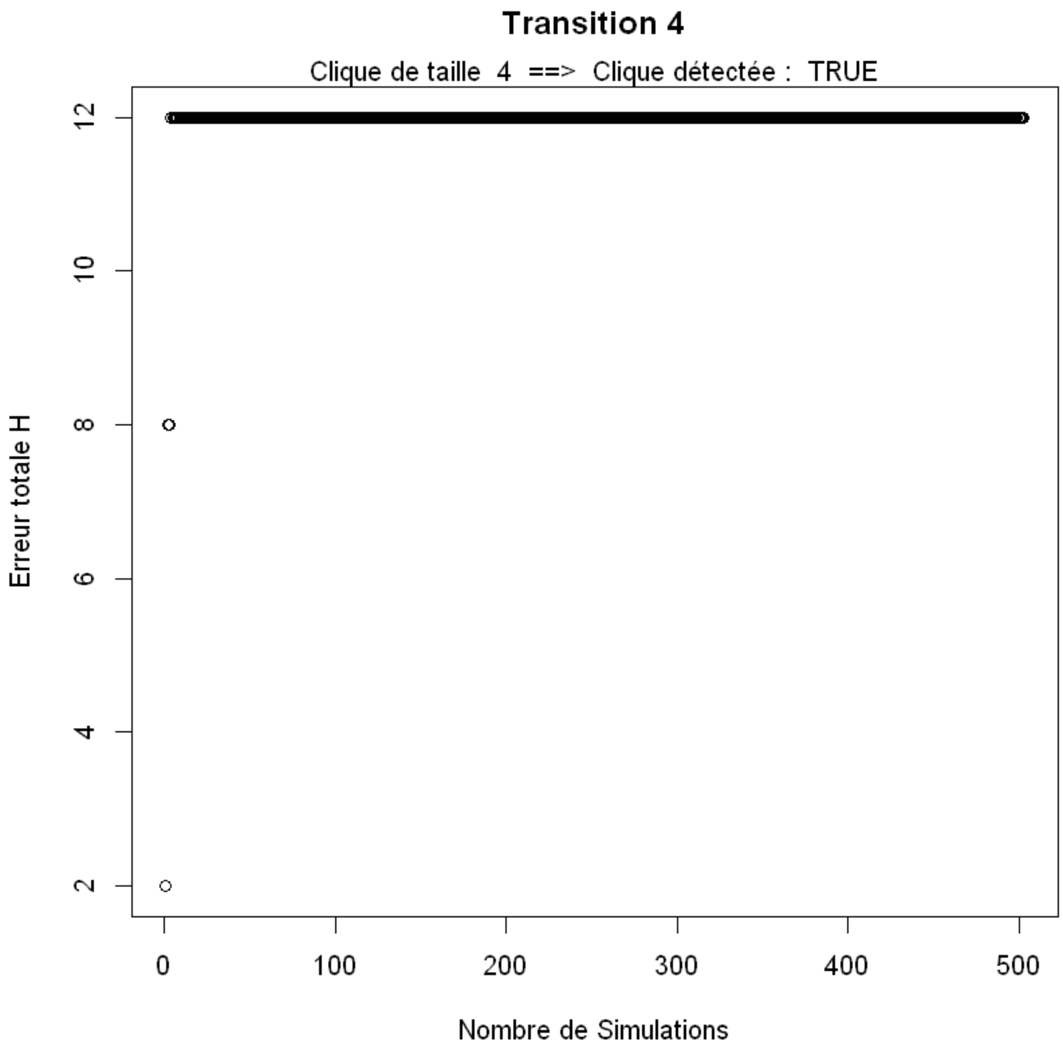

Transition 1
Clique de taille 4 ==> Clique détectée : FALSE
Erreur totale H
Nombre de Simulations
0
2
4
6
8
10
0
100
200
300
400
500

```
[1] 12
```

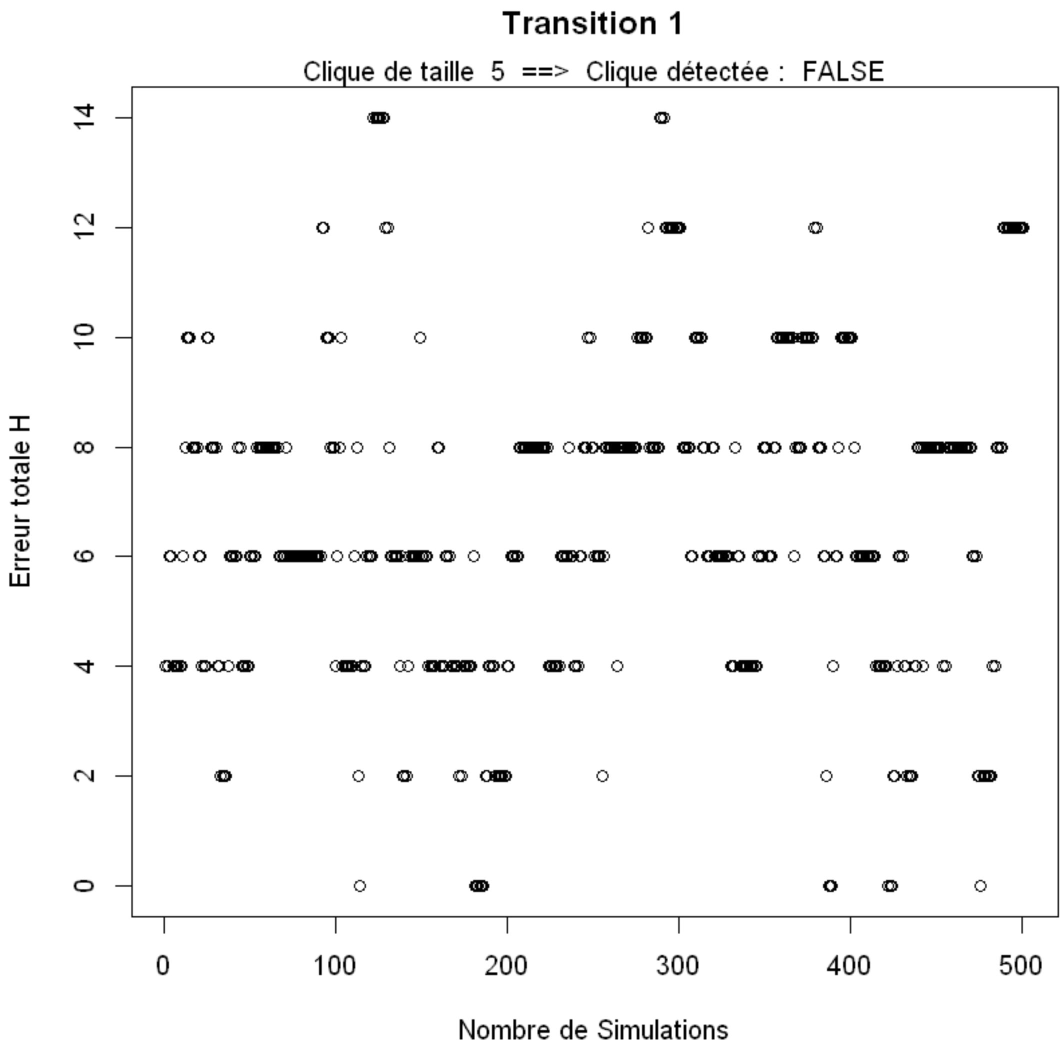

Transition 2
Clique de taille 4 ==> Clique détectée : TRUE
Erreur totale H
2
4
6
8
10
12
0
100
200
300
400
500
Nombre de Simulations

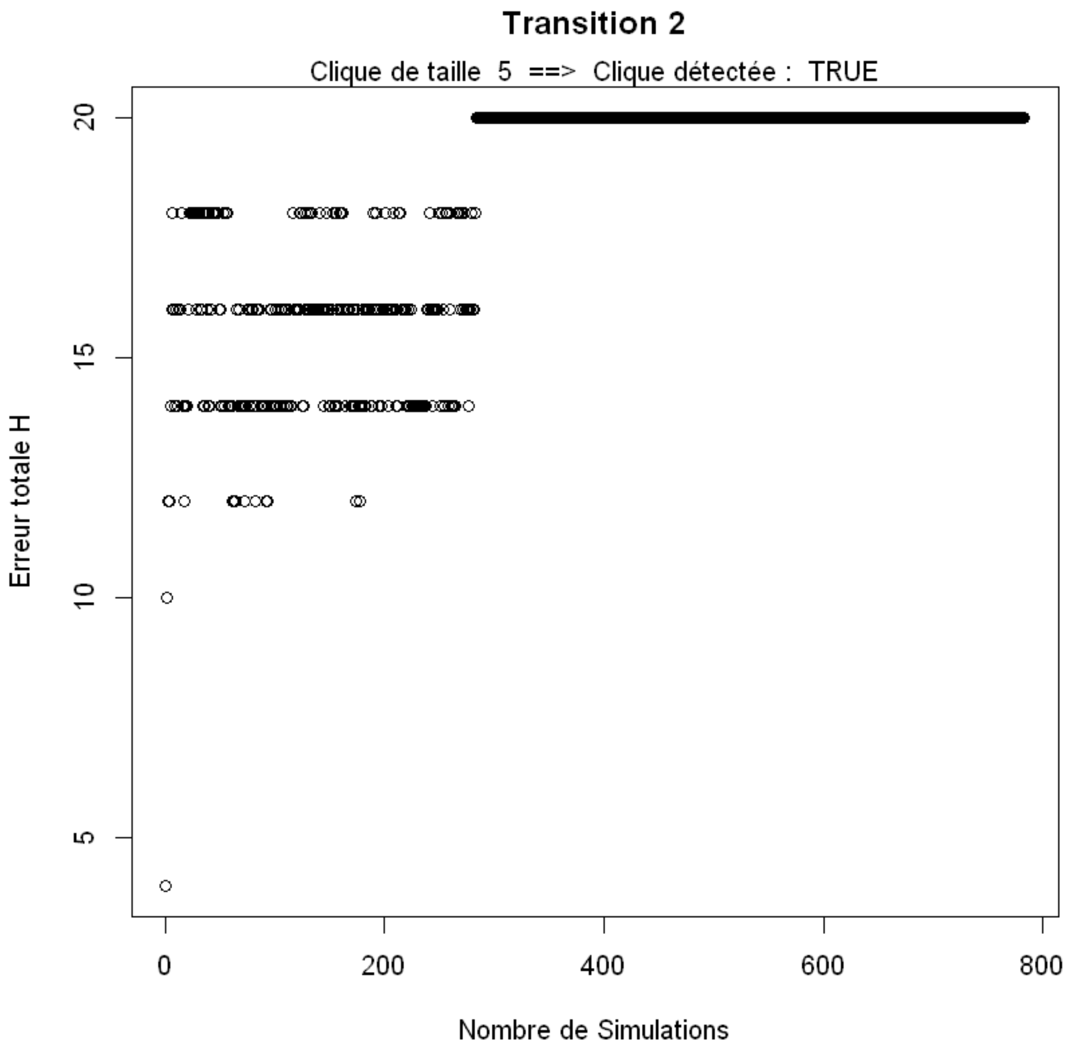
```
[1] 12
```


Transition 3
Clique de taille 4 ==> Clique détectée : TRUE
Erreur totale H
0
2
4
6
8
10
12
0
100
200
300
400
500
Nombre de Simulations

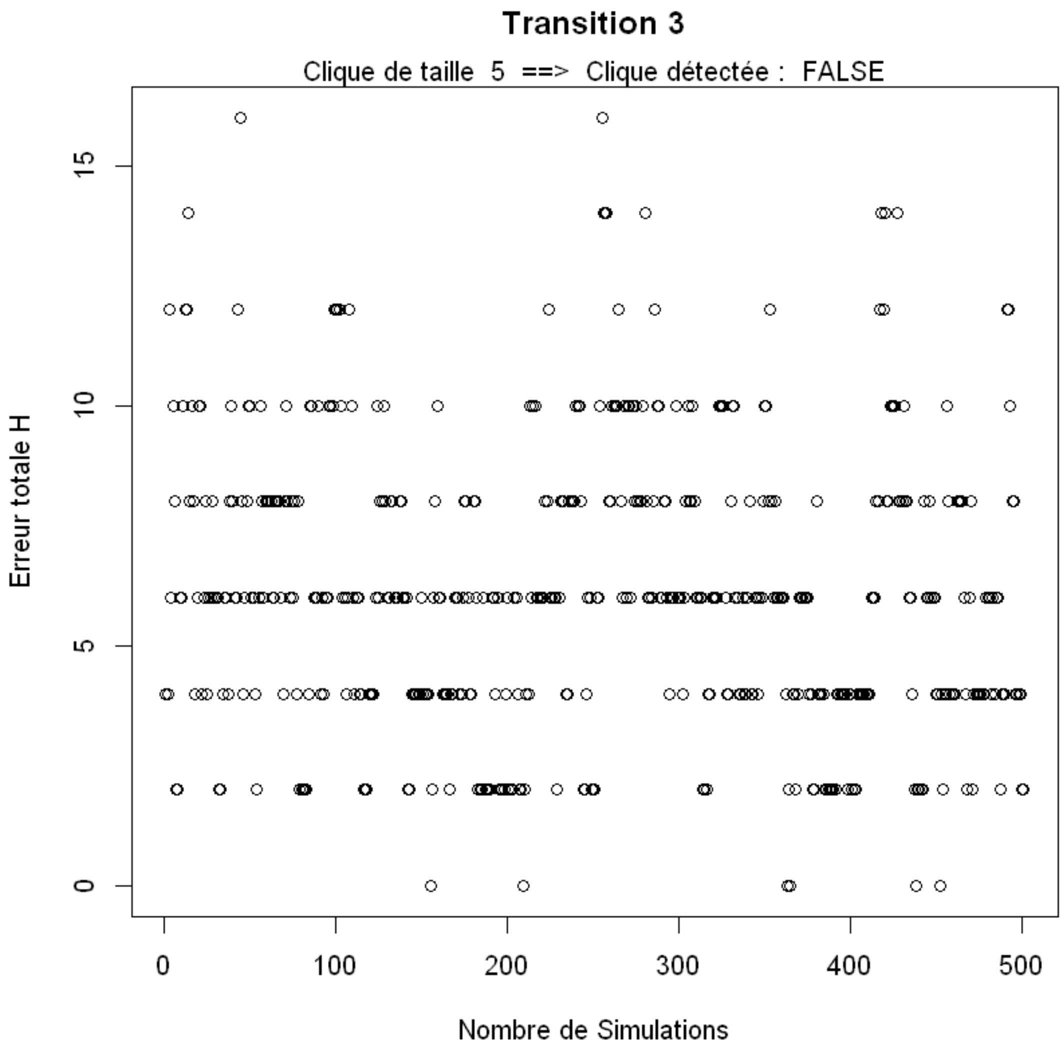

Transition 4
Clique de taille 4 ==> Clique détectée : TRUE
Erreur totale H
2
4
6
8
10
12
0
100
200
300
400
500
Nombre de Simulations

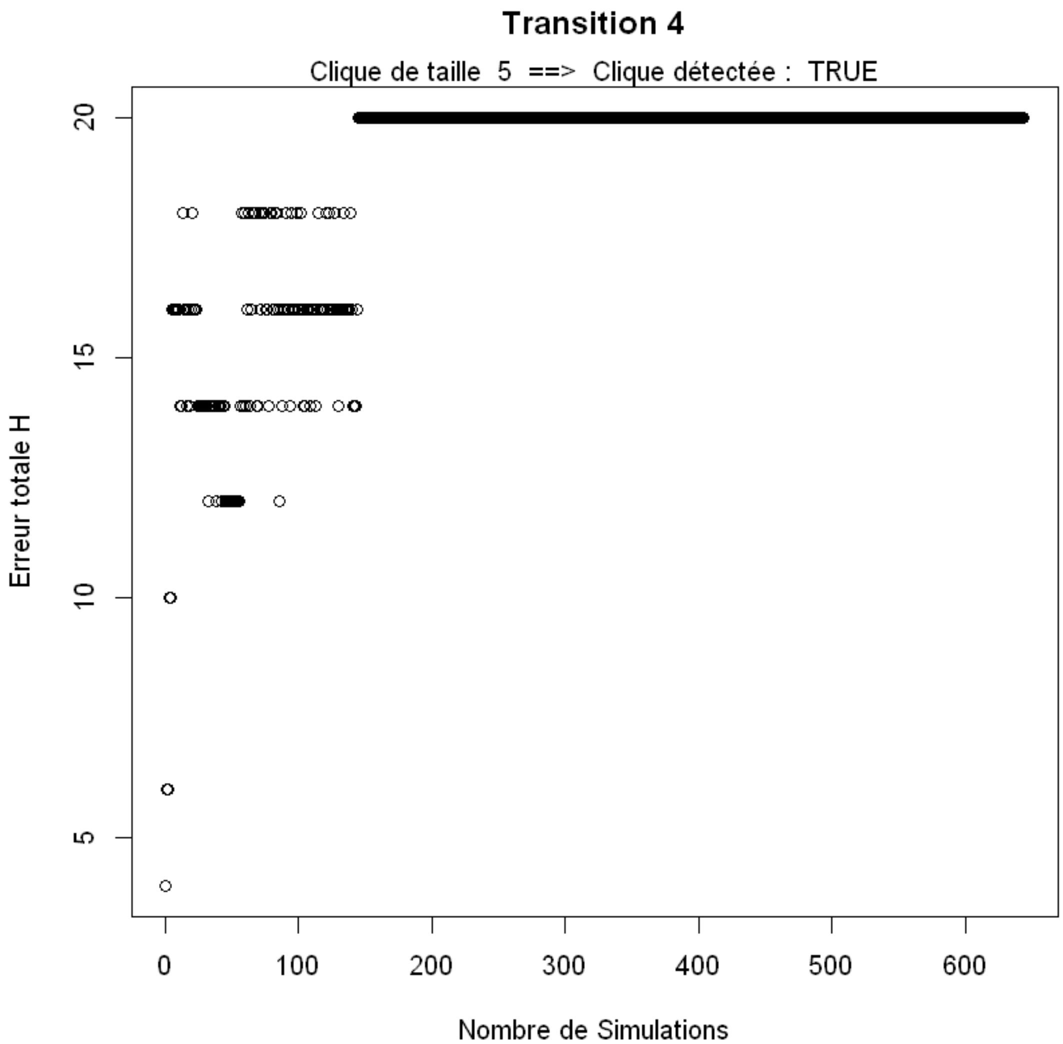
```
[1] 20
```


Transition 1
Clique de taille 5 ==> Clique détectée : FALSE
Erreur totale H
Nombre de Simulations
0
2
4
6
8
10
12
14
0
100
200
300
400
500

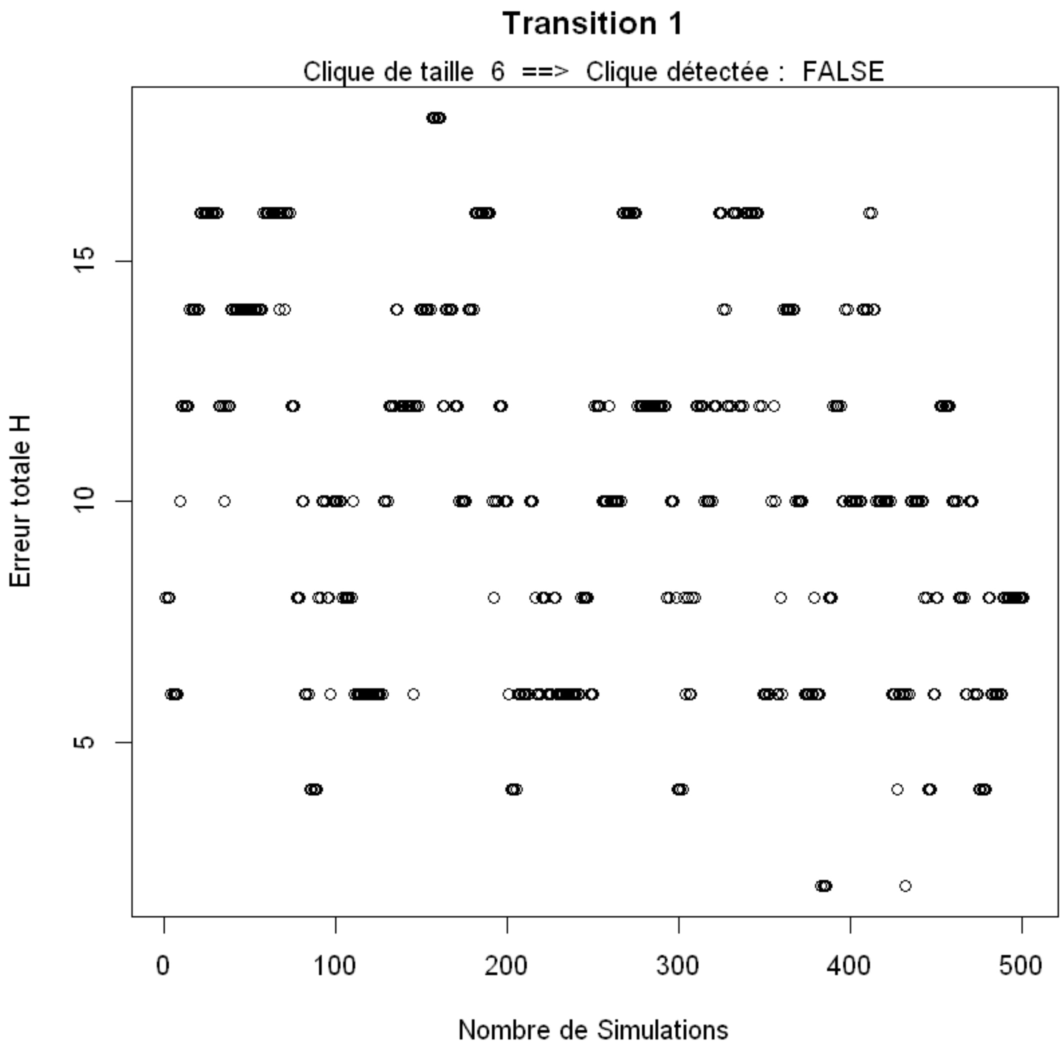

Transition 2
Clique de taille 5 ==> Clique détectée : TRUE
Erreur totale H
Nombre de Simulations
5
10
15
20
0
200
400
600
800

[1] 20

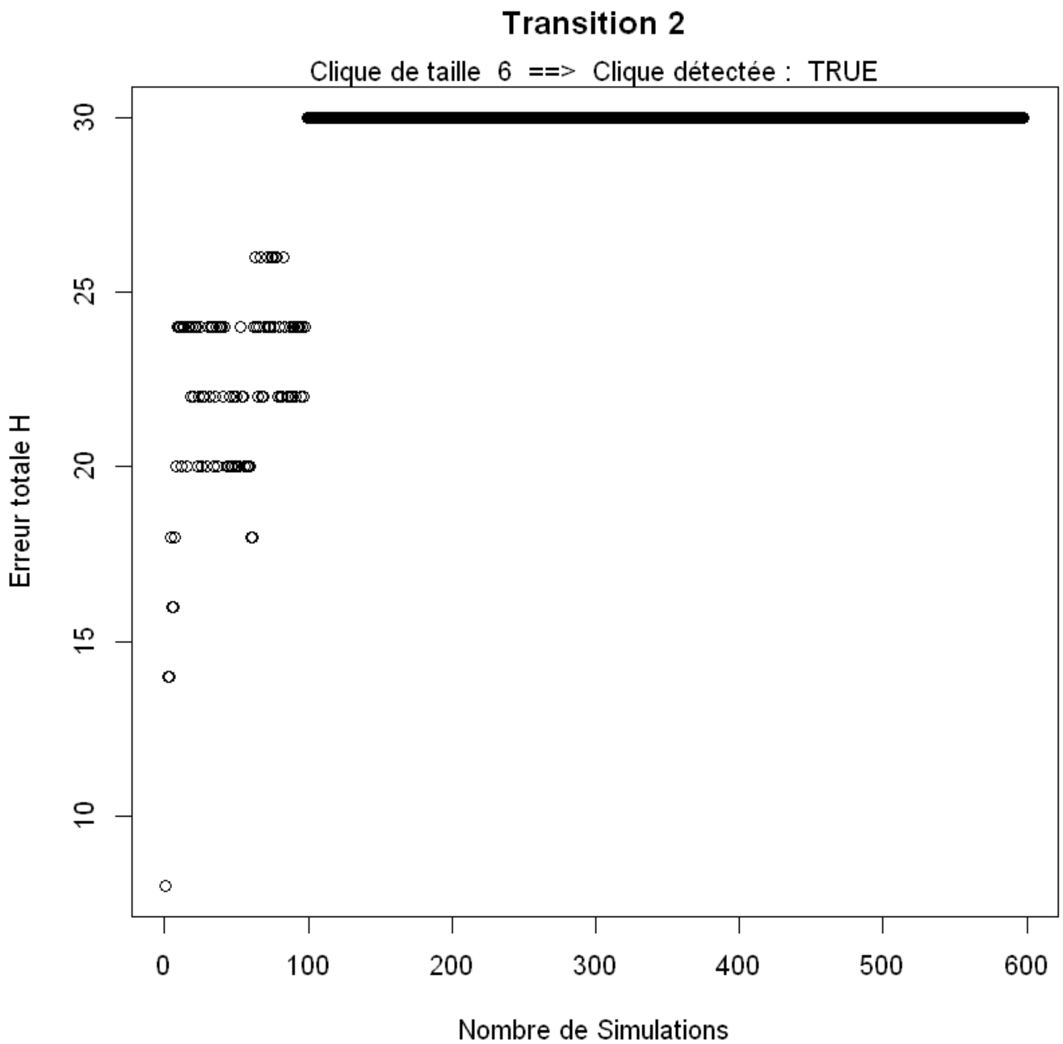

Transition 3
Clique de taille 5 ==> Clique détectée : FALSE
Erreur totale H
0
5
10
15
0
100
200
300
400
500
Nombre de Simulations


Transition 4
Clique de taille 5 ==> Clique détectée : TRUE
Erreur totale H
20
15
10
5
0
100
200
300
400
500
600
Nombre de Simulations

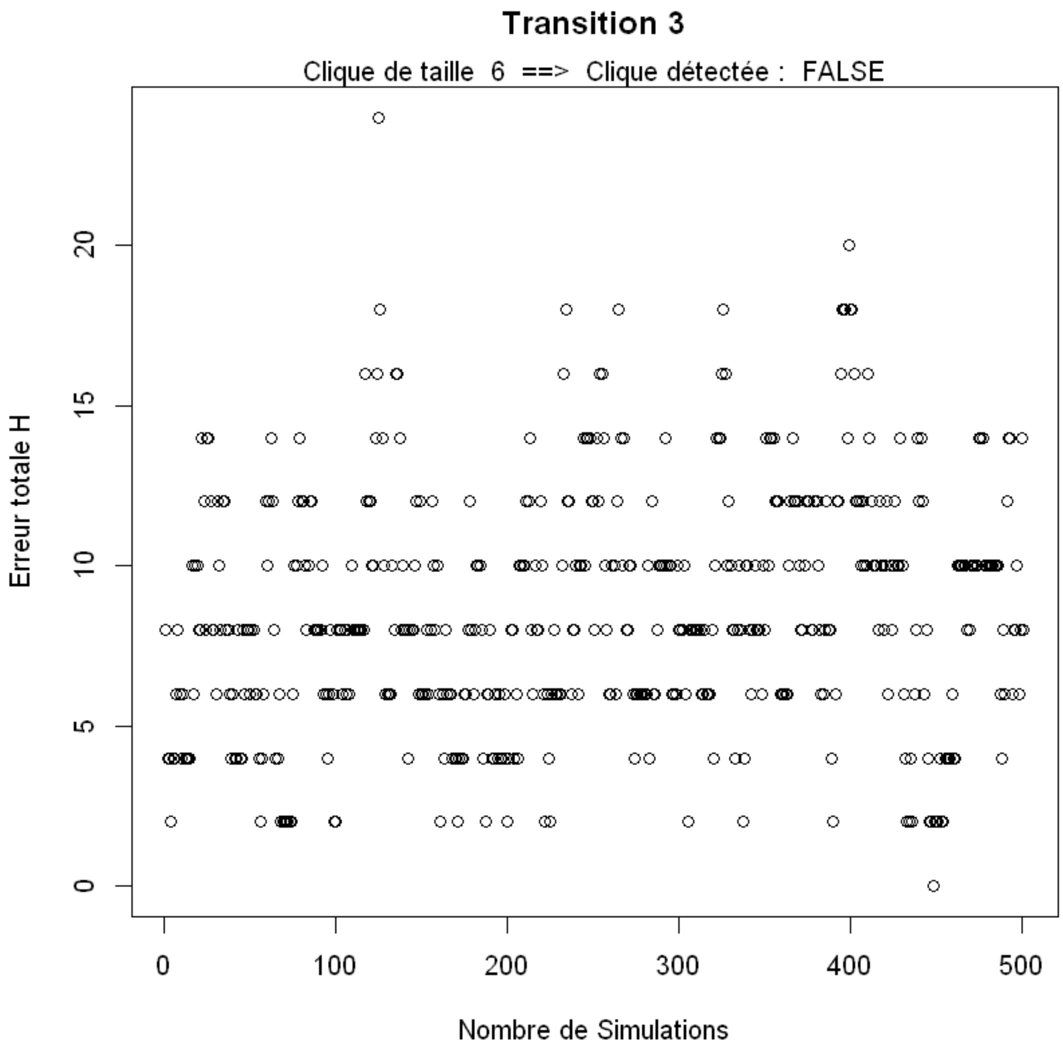
```
[1] 30
```

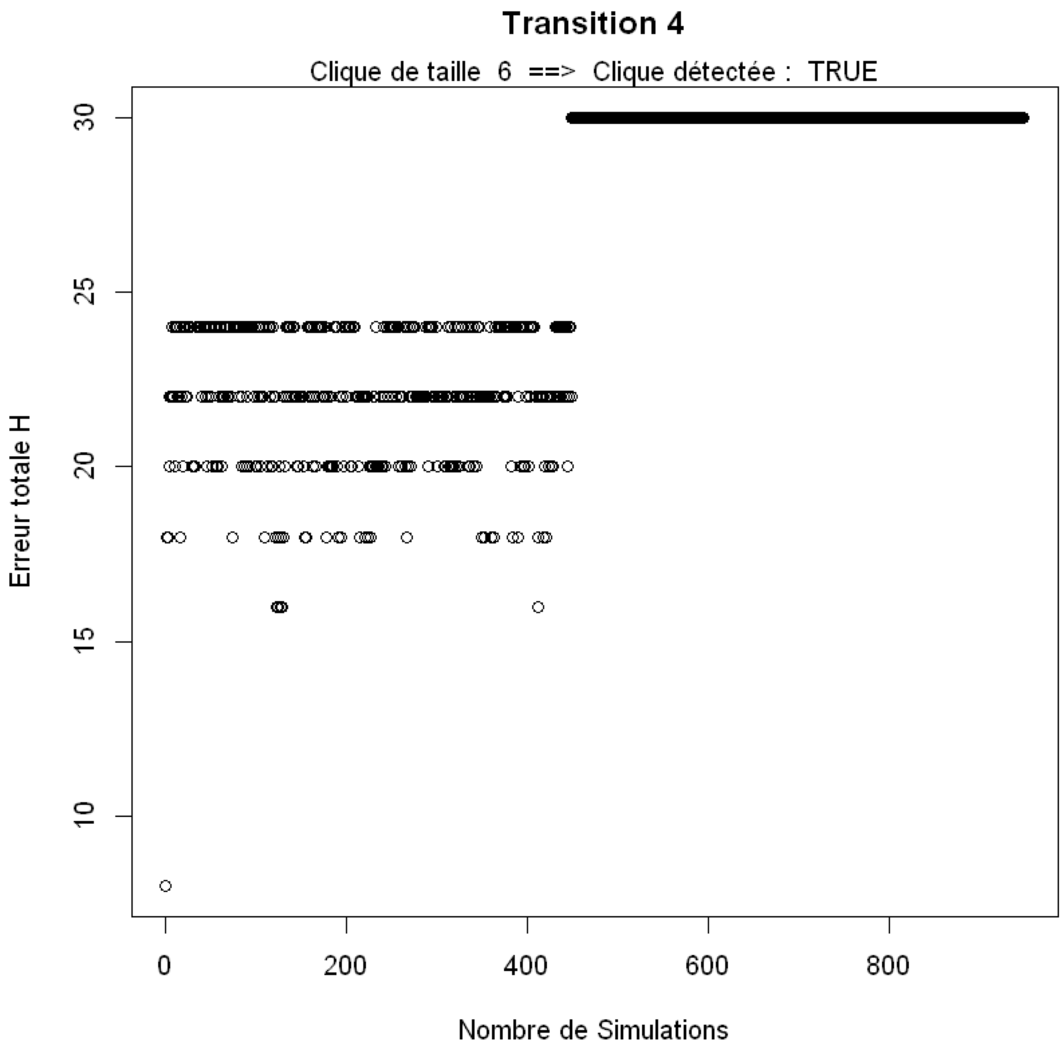


Transition 1
Clique de taille 6 ==> Clique détectée : FALSE
Erreur totale H
Nombre de Simulations
5
10
15
0
100
200
300
400
500

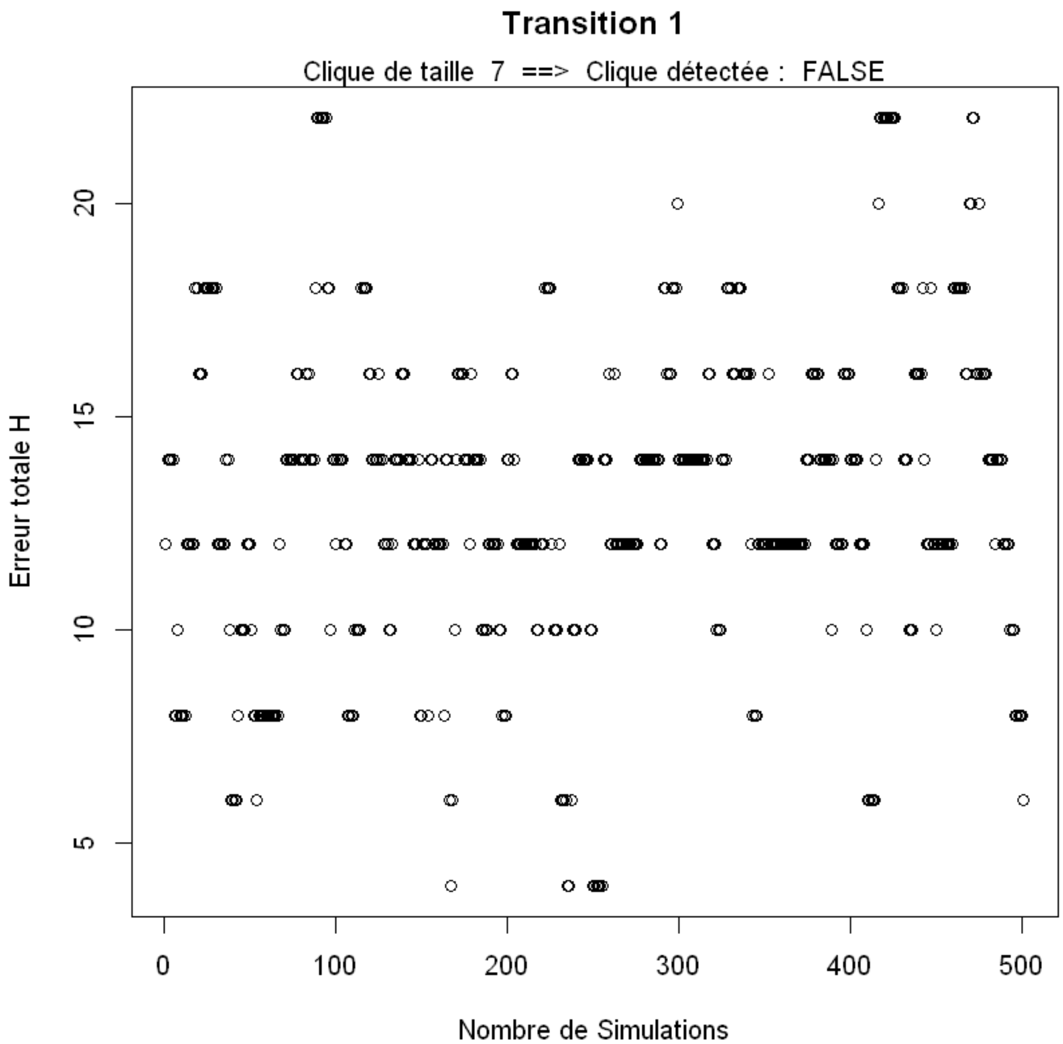

Transition 2
Clique de taille 6 ==> Clique détectée : TRUE
Erreur totale H
Nombre de Simulations
10
15
20
25
30
0
100
200
300
400
500
600

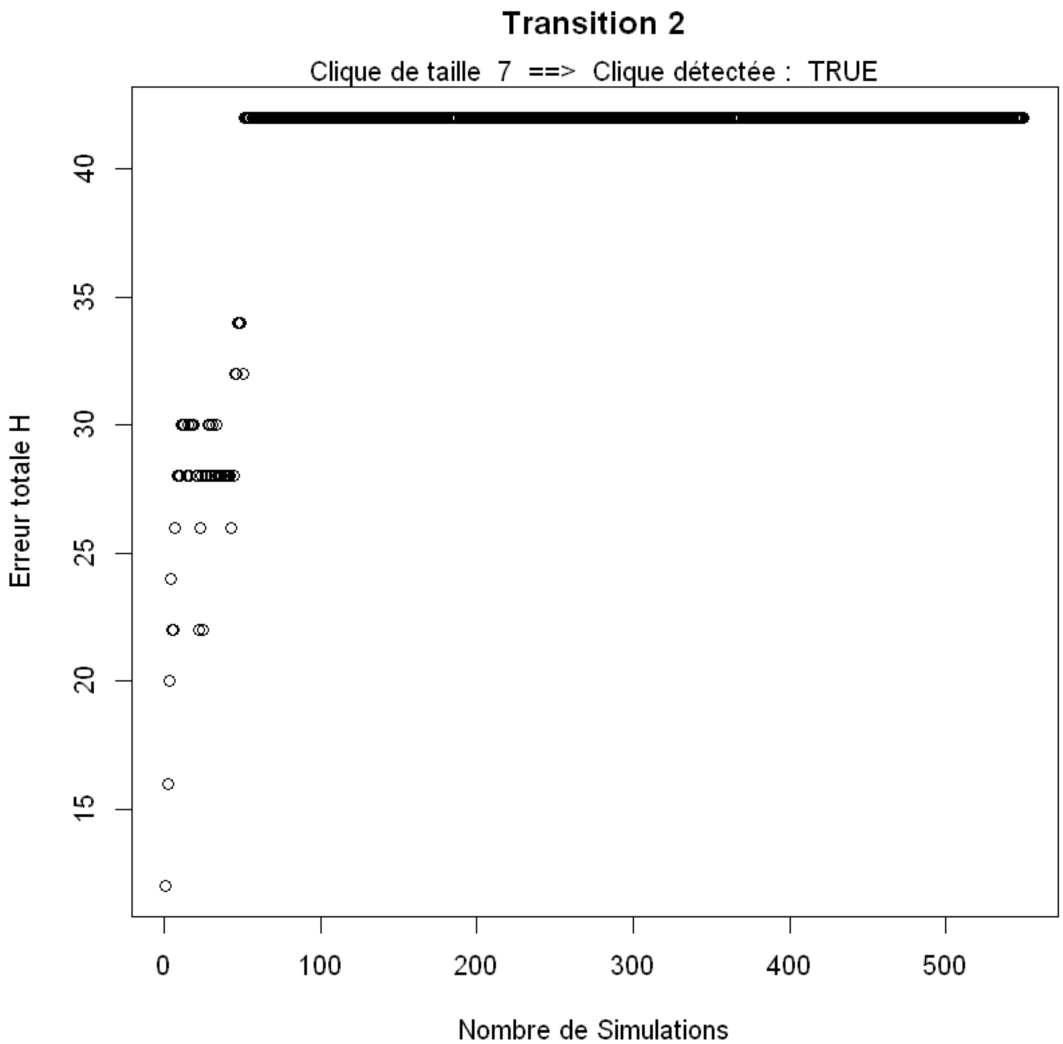
```
[1] 30
```


Transition 3
Clique de taille 6 ==> Clique détectée : FALSE
Erreur totale H
0
5
10
15
20
0
100
200
300
400
500
Nombre de Simulations

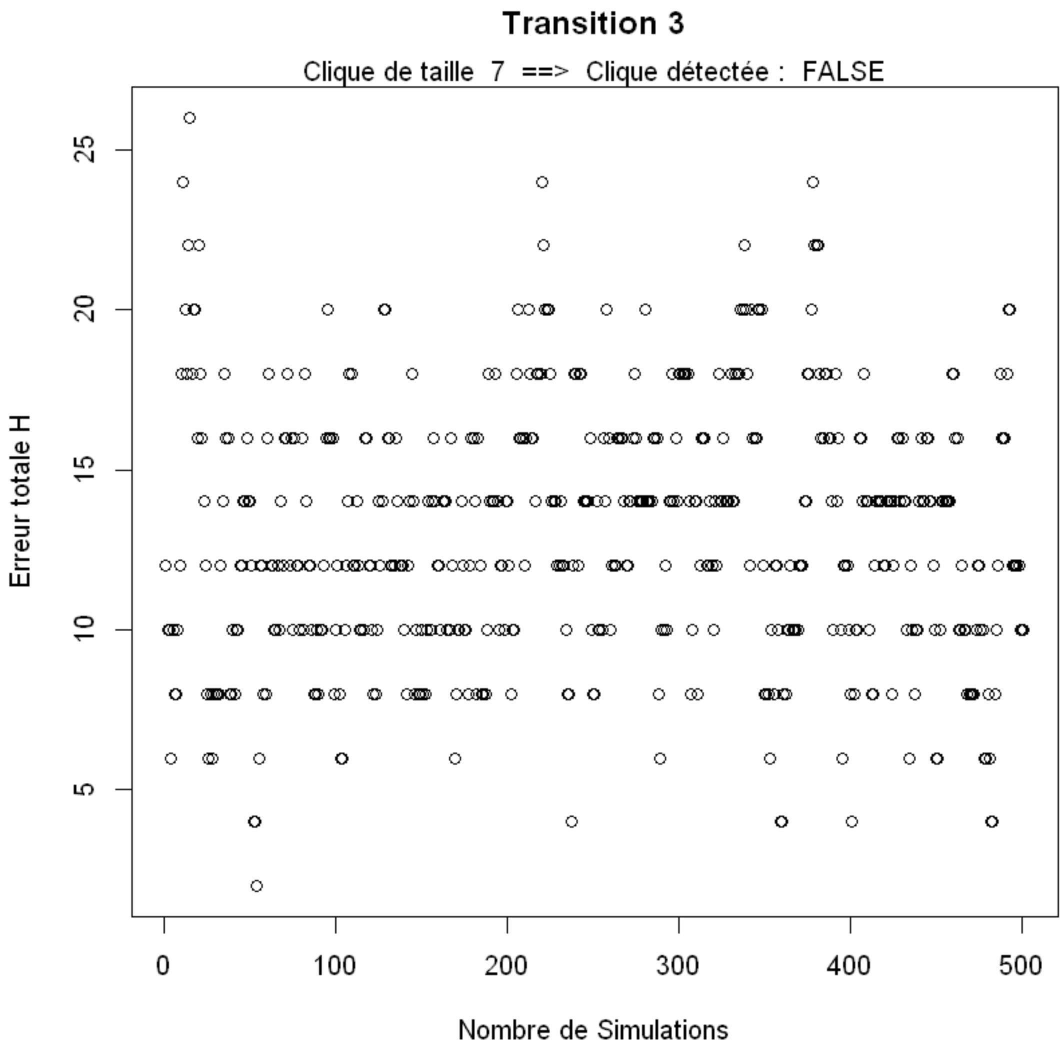

Transition 4
Clique de taille 6 ==> Clique détectée : TRUE
Erreur totale H
30
25
20
15
10
0
200
400
600
800
Nombre de Simulations

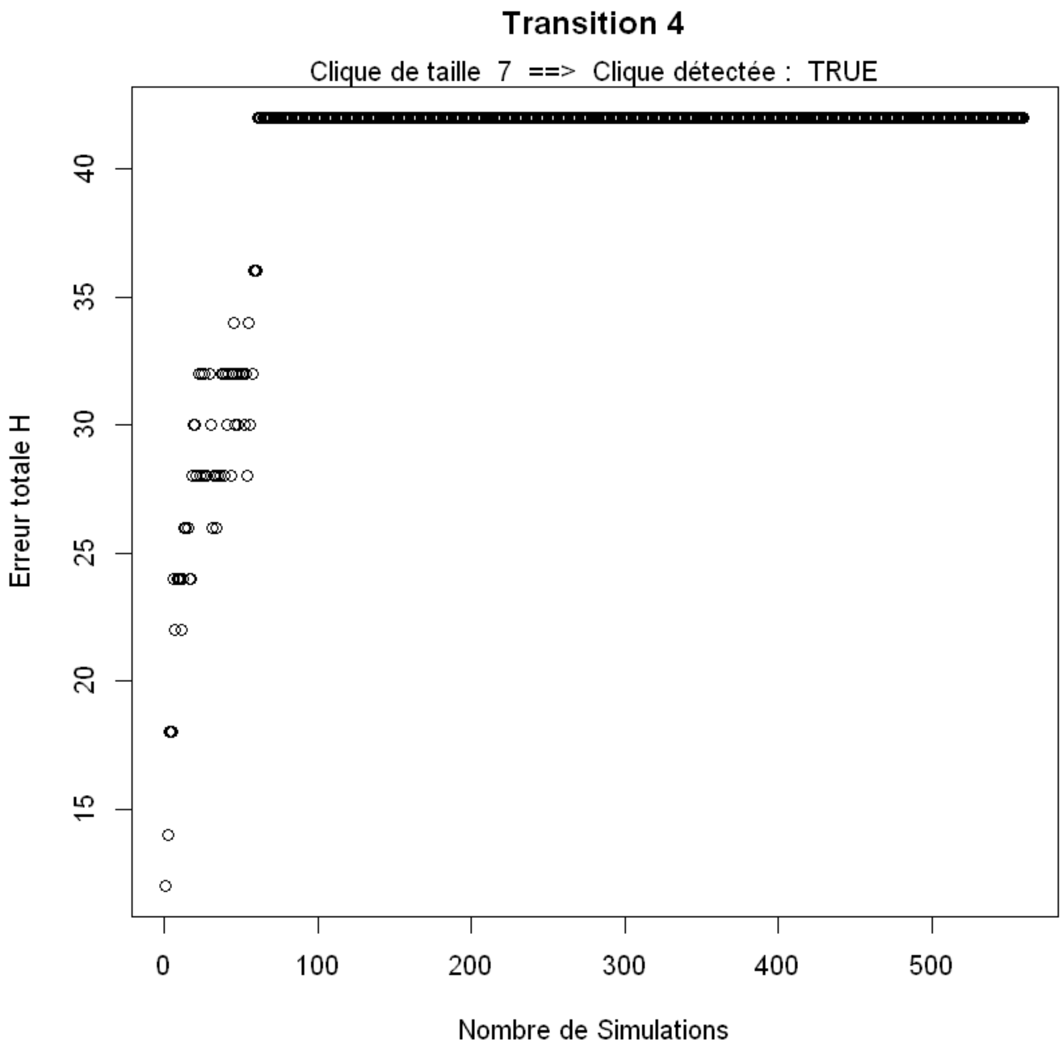
```
[1] 42
```

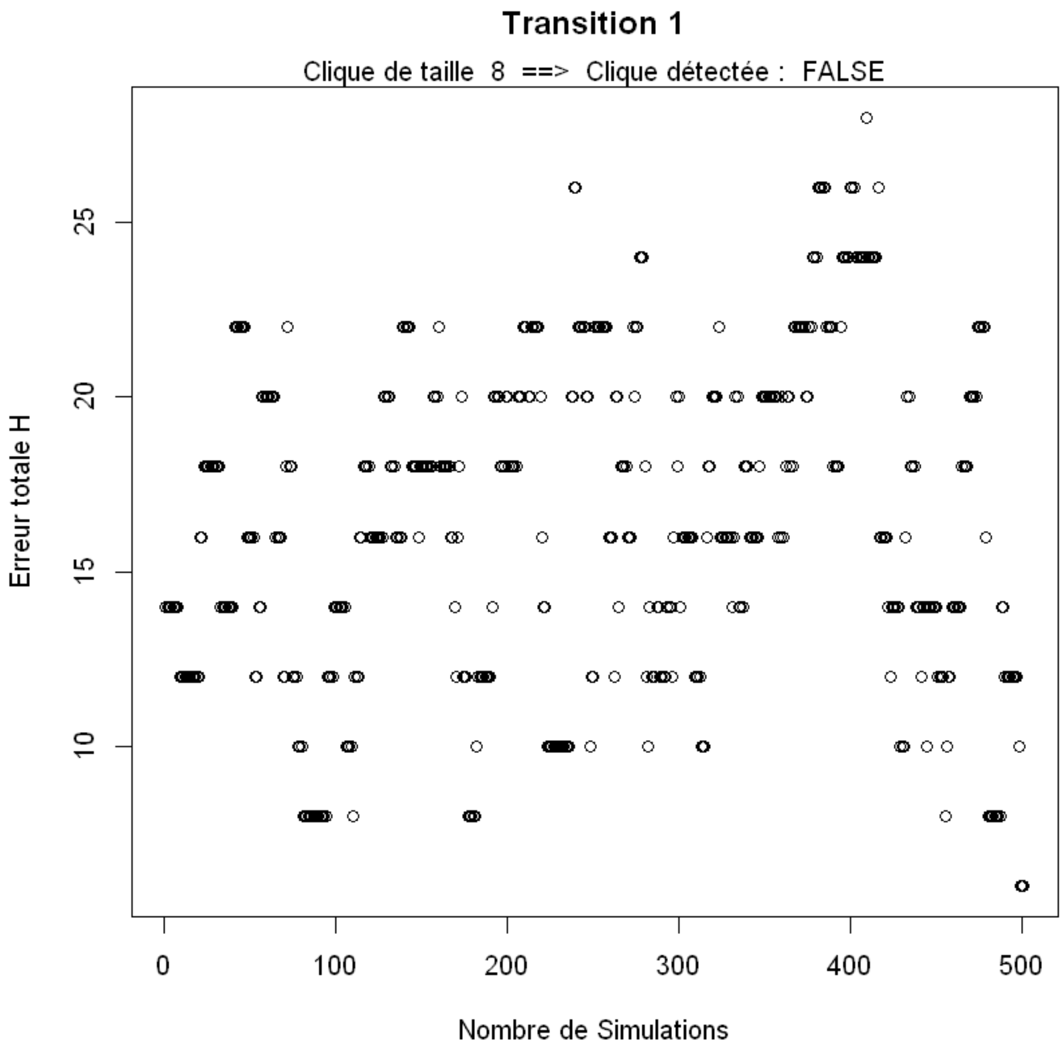

Transition 1
Clique de taille 7 ==> Clique détectée : FALSE
Erreur totale H
5
10
15
20
0
100
200
300
400
500
Nombre de Simulations

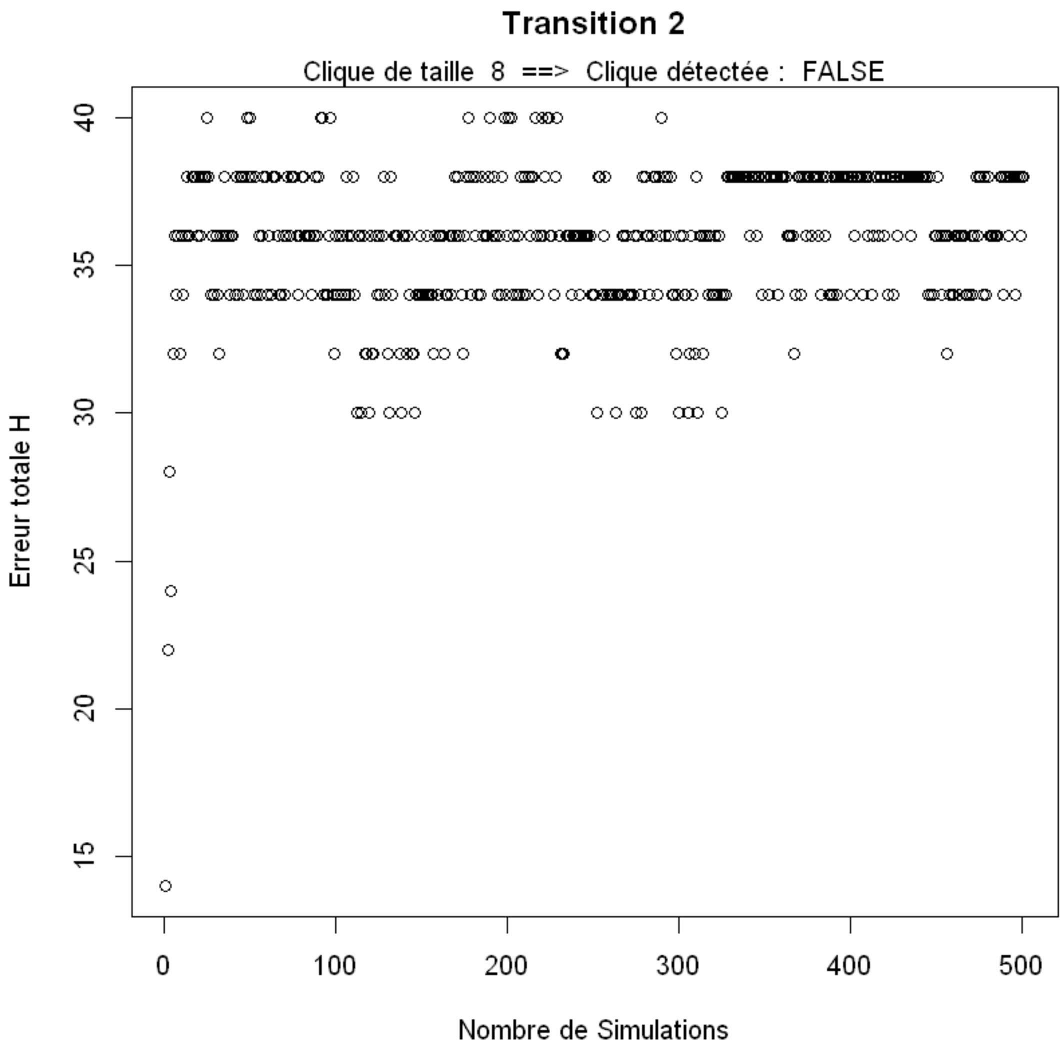

Transition 2
Clique de taille 7 ==> Clique détectée : TRUE
Erreur totale H
Nombre de Simulations
15
20
25
30
35
40
0
100
200
300
400
500

```
[1] 42
```

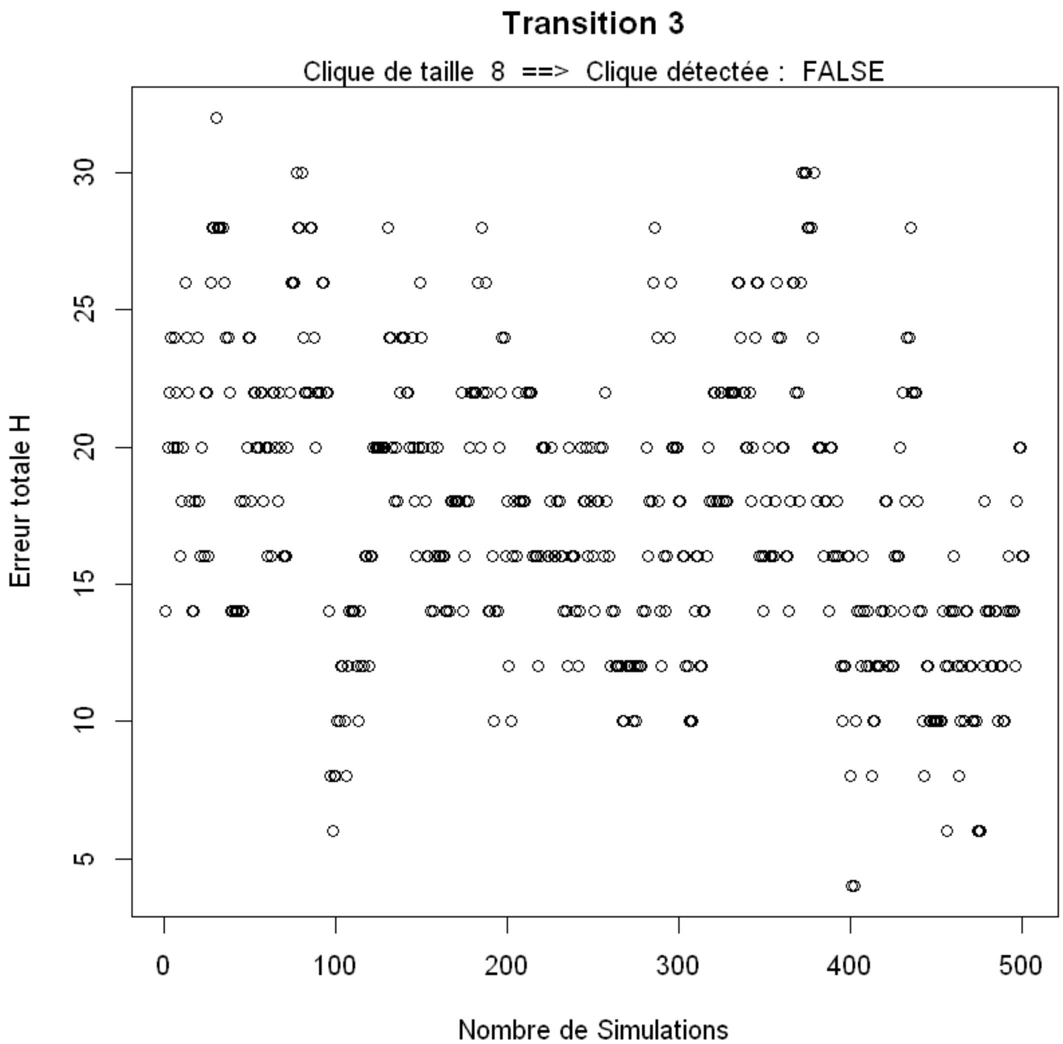

Transition 3
Clique de taille 7 ==> Clique détectée : FALSE
Erreur totale H
Nombre de Simulations
25
20
15
10
5
0
100
200
300
400
500


Transition 4
Clique de taille 7 ==> Clique détectée : TRUE
Erreur totale H
15
20
25
30
35
40
0
100
200
300
400
500
Nombre de Simulations

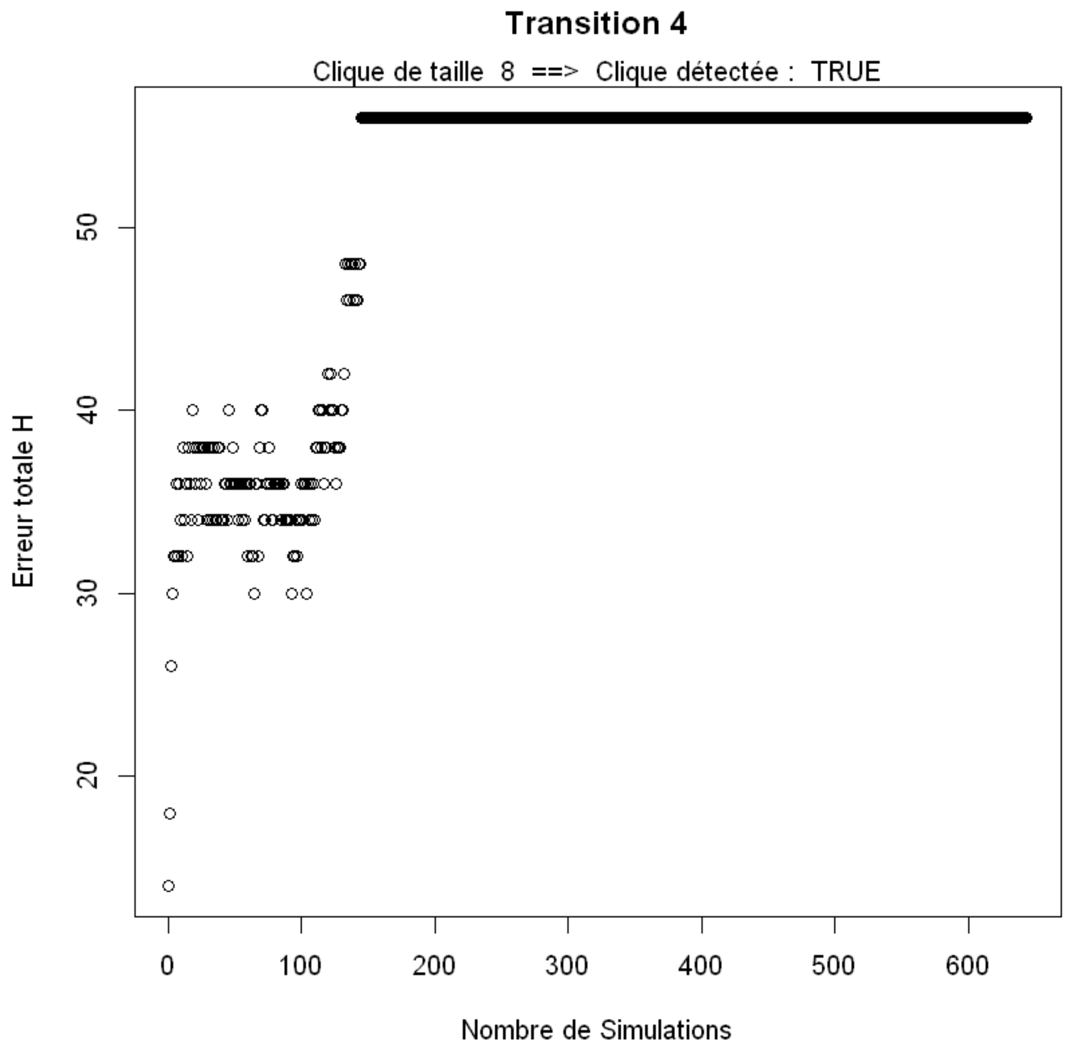

Transition 1
Clique de taille 8 ==> Clique détectée : FALSE
Erreur totale H
10
15
20
25
0
100
200
300
400
500
Nombre de Simulations

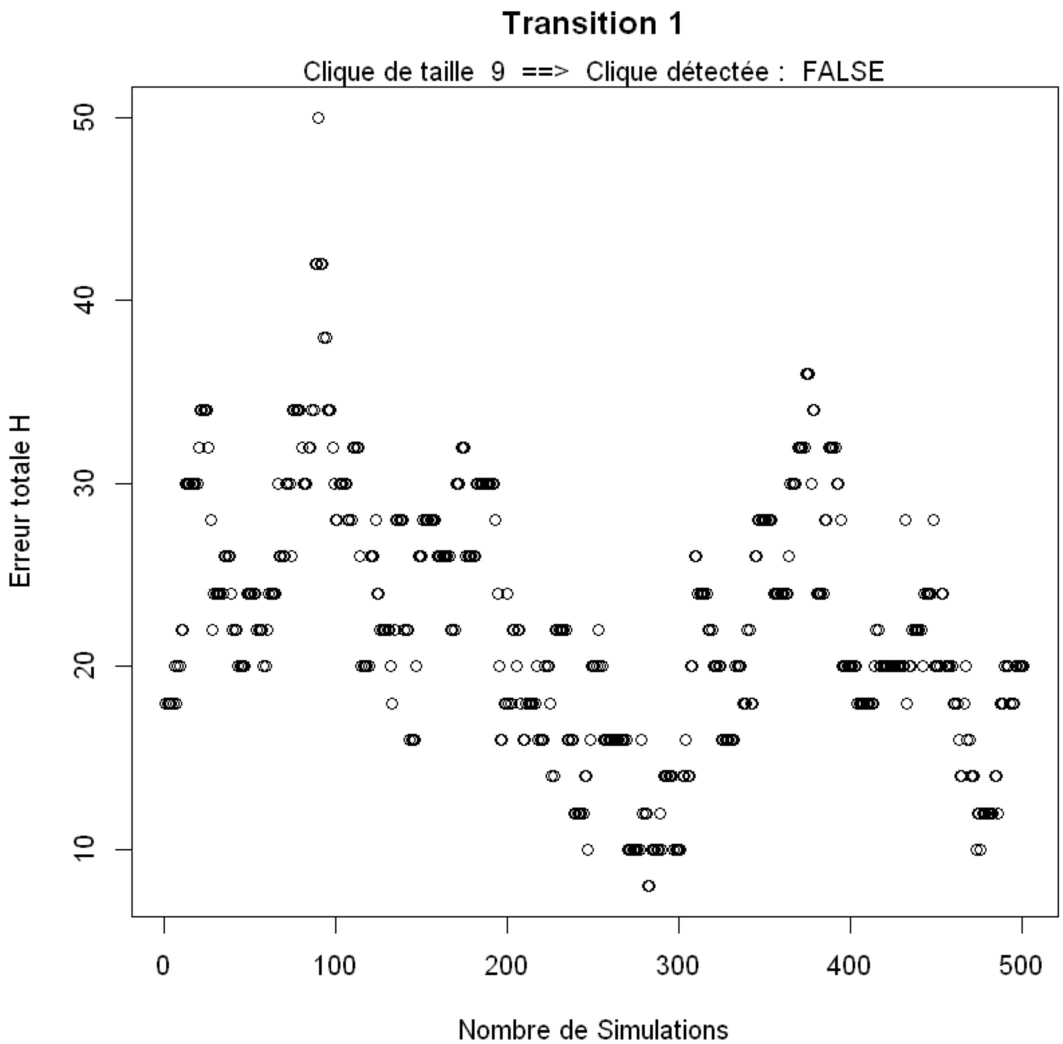

Transition 2
Clique de taille 8 ==> Clique détectée : FALSE
Erreur totale H
Nombre de Simulations
40
35
30
25
20
15
0
100
200
300
400
500

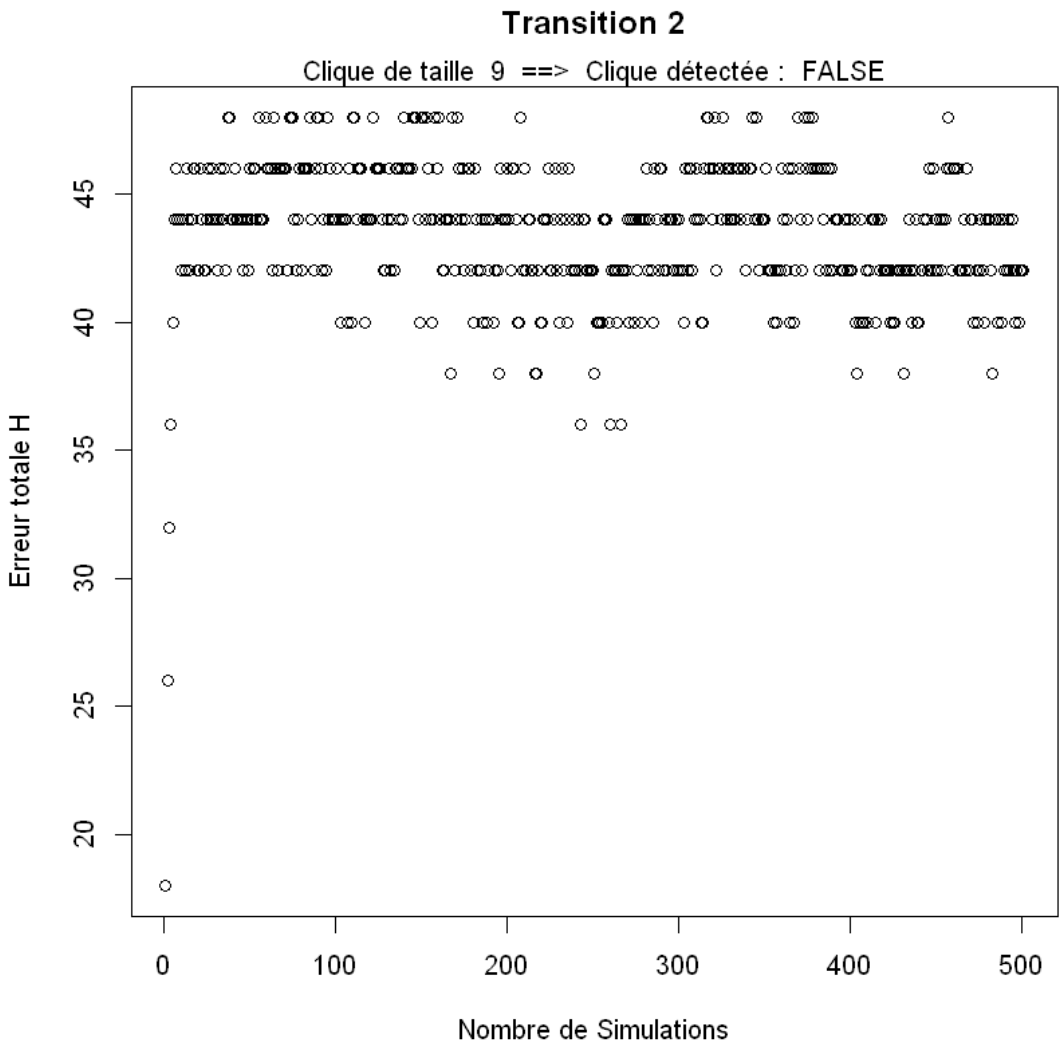
```
[1] 56
```



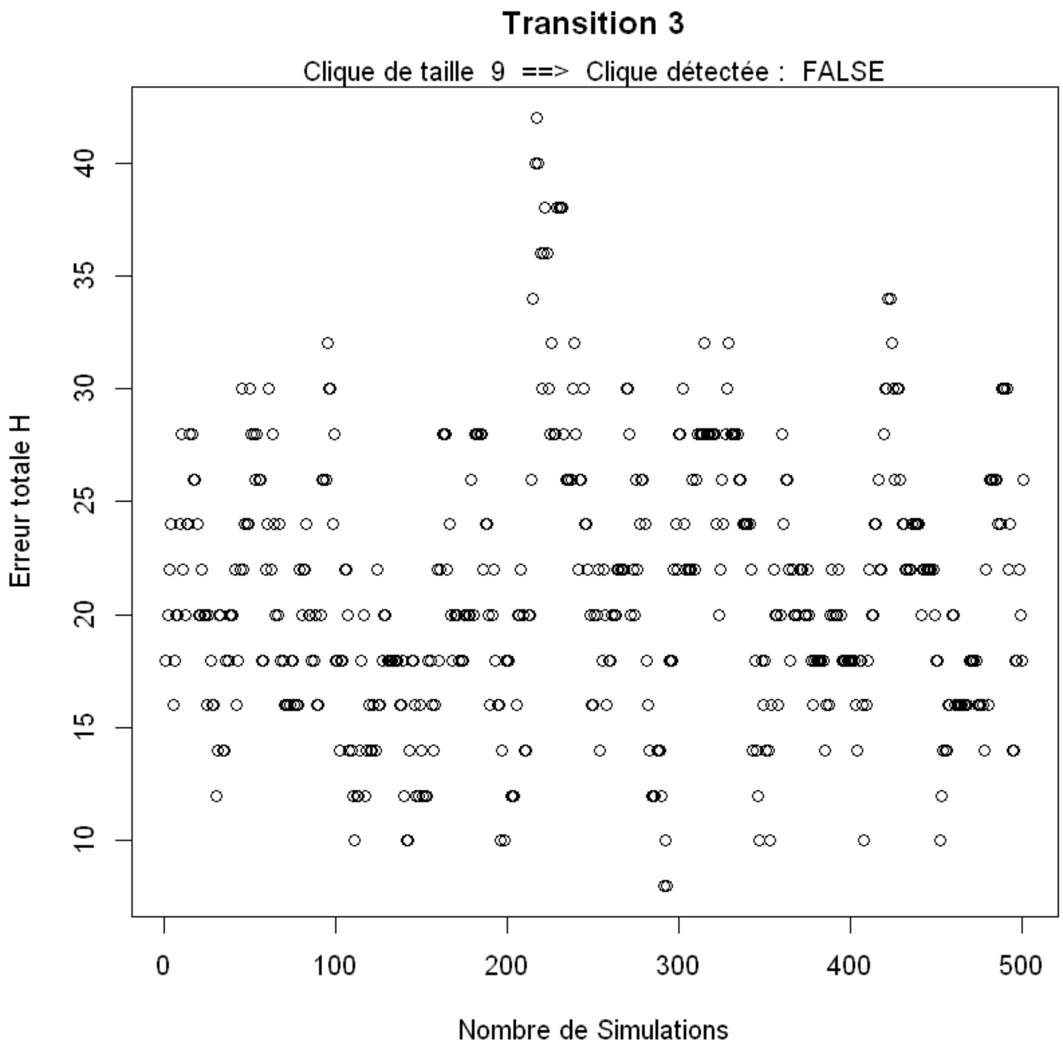
Transition 3
Clique de taille 8 ==> Clique détectée : FALSE
Erreur totale H
Nombre de Simulations
5
10
15
20
25
30
0
100
200
300
400
500


Transition 4
Clique de taille 8 ==> Clique détectée : TRUE
Erreur totale H
Nombre de Simulations
20
30
40
50
0
100
200
300
400
500
600

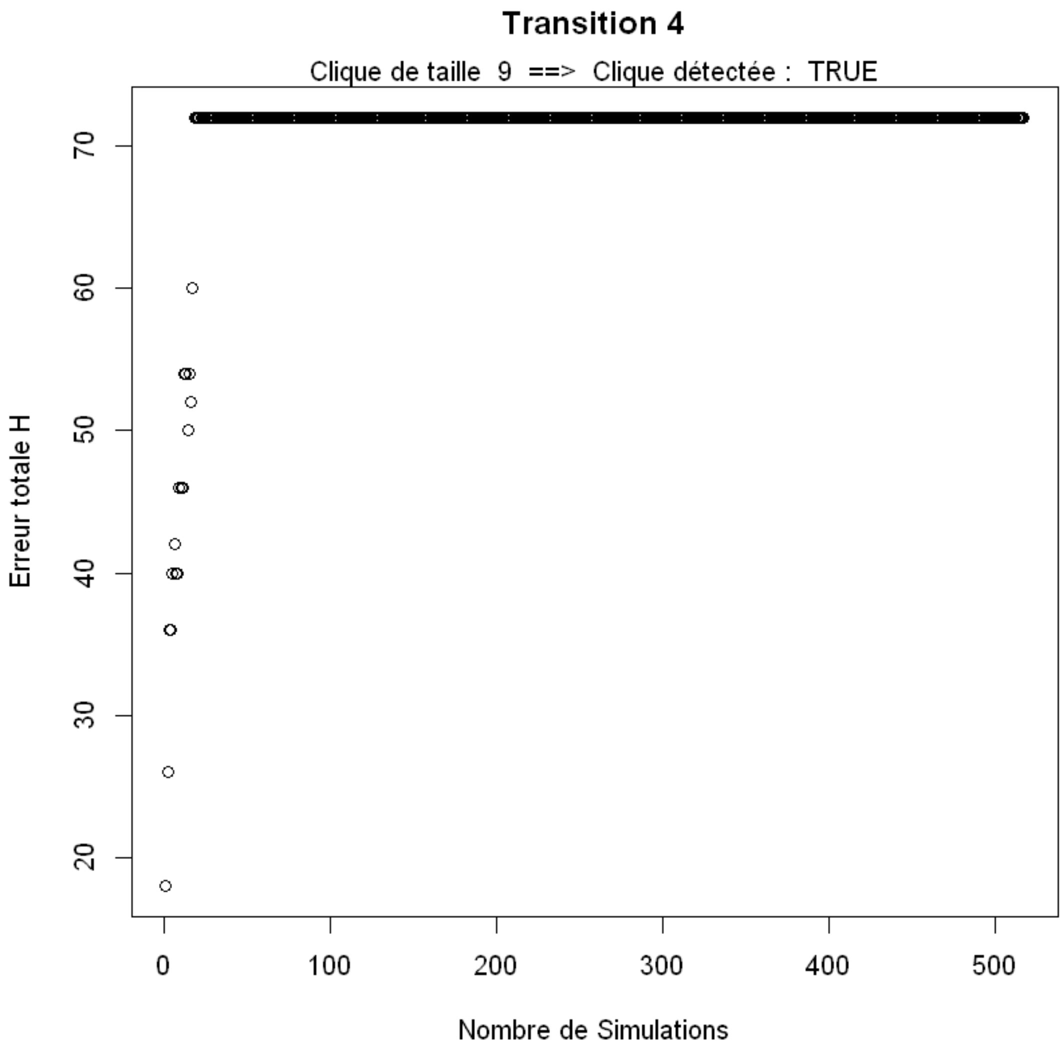

Transition 1
Clique de taille 9 ==> Clique détectée : FALSE
Erreur totale H
Nombre de Simulations
10
20
30
40
50
0
100
200
300
400
500


Transition 2
Clique de taille 9 ==> Clique détectée : FALSE
Erreur totale H
20
25
30
35
40
45
0
100
200
300
400
500
Nombre de Simulations

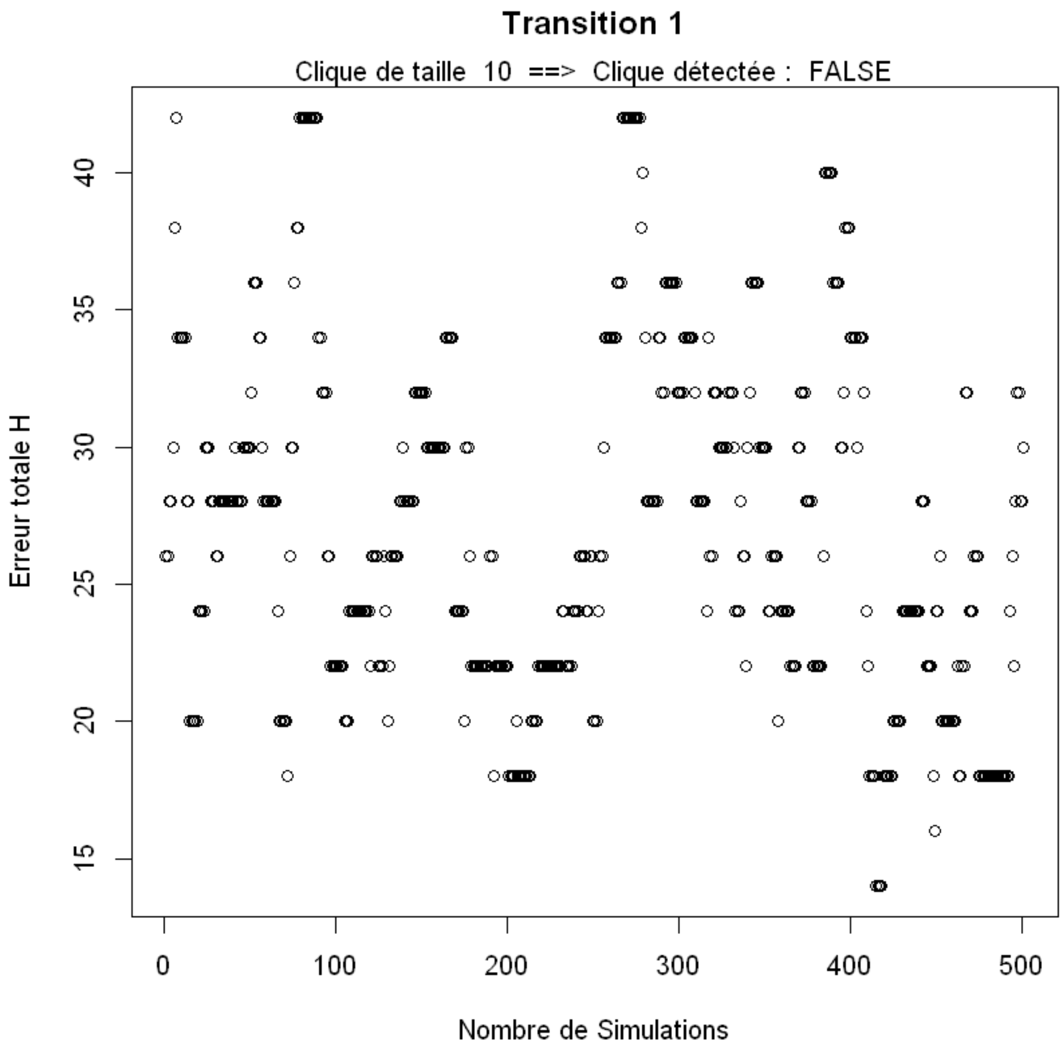
```
[1] 72
```

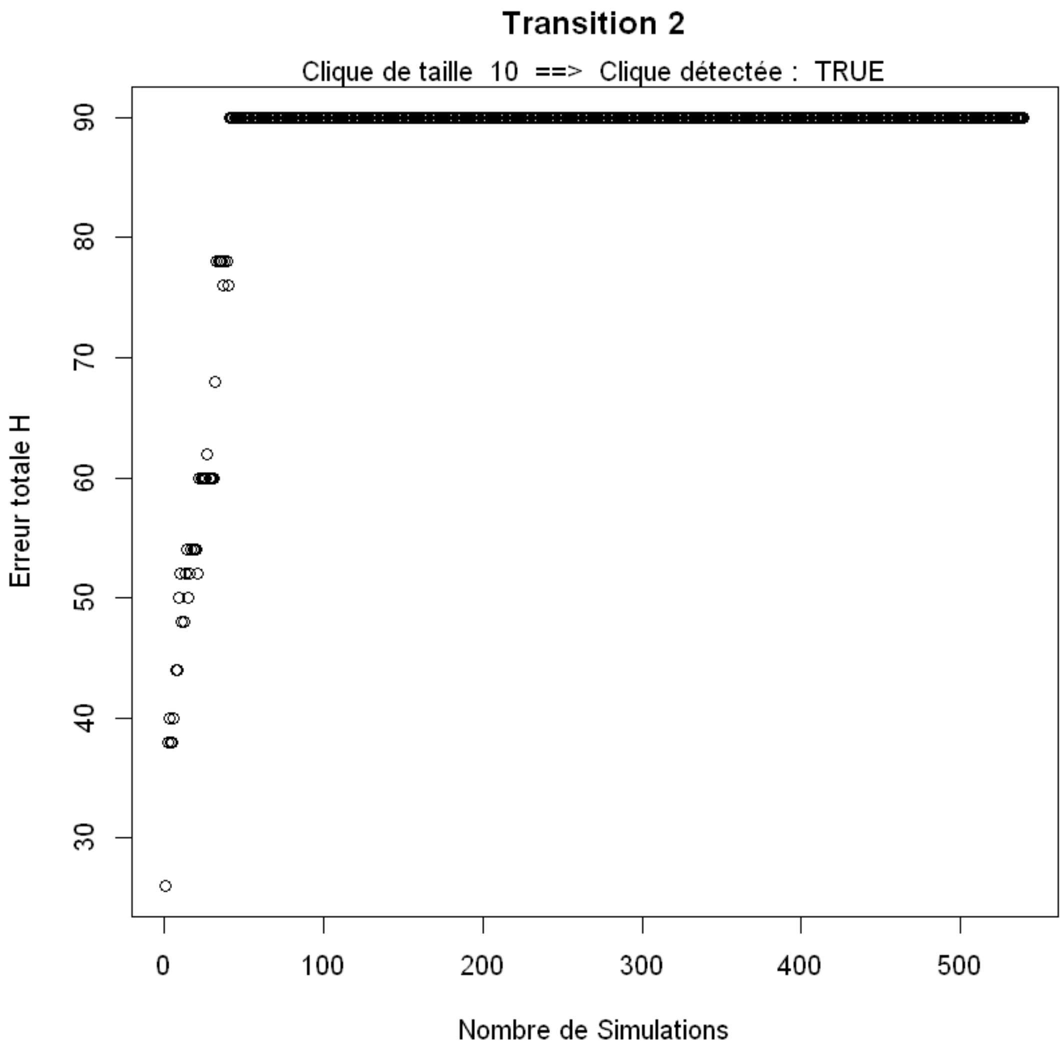


Transition 3
Clique de taille 9 ==> Clique détectée : FALSE
Erreur totale H
10
15
20
25
30
35
40
0
100
200
300
400
500
Nombre de Simulations

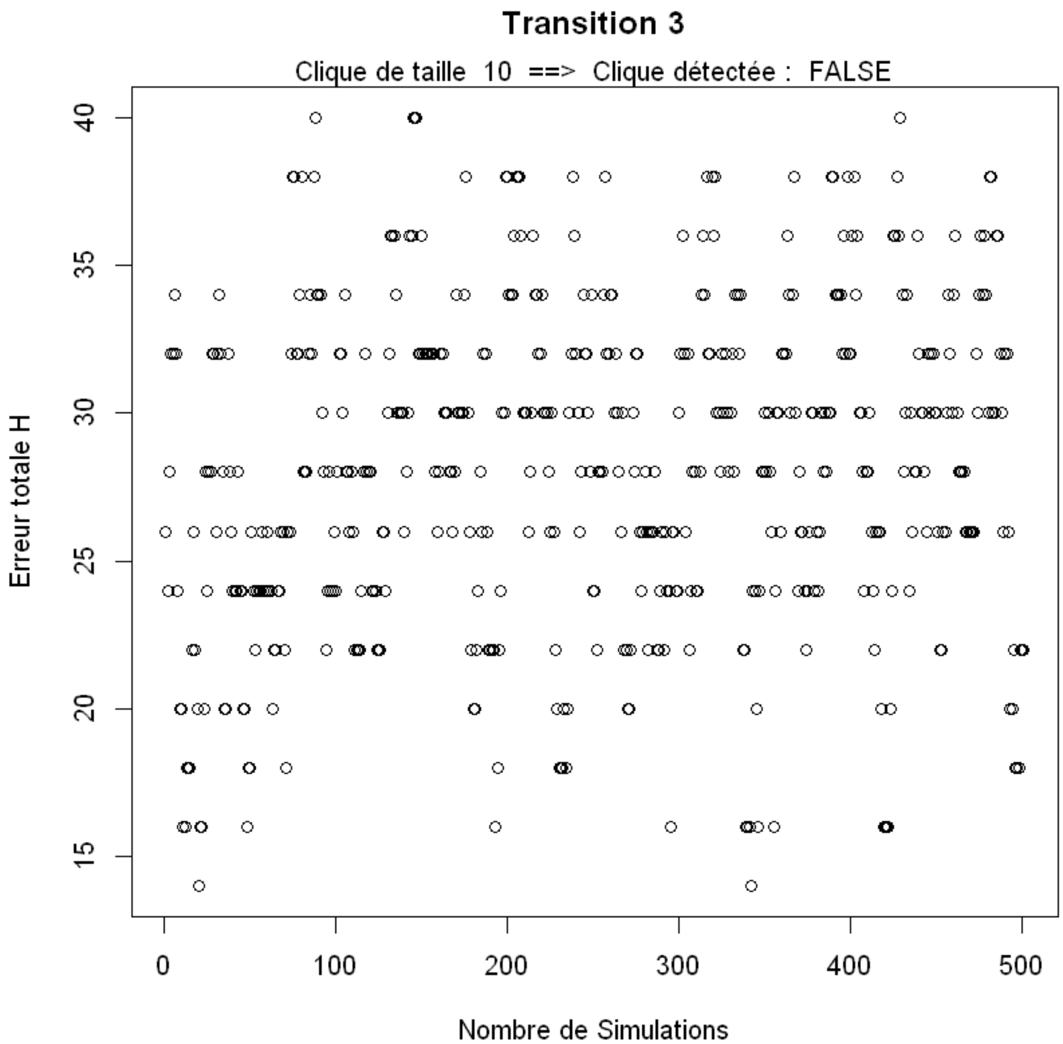

Transition 4
Clique de taille 9 ==> Clique détectée : TRUE
Erreur totale H
Nombre de Simulations
70
60
50
40
30
20
0
100
200
300
400
500

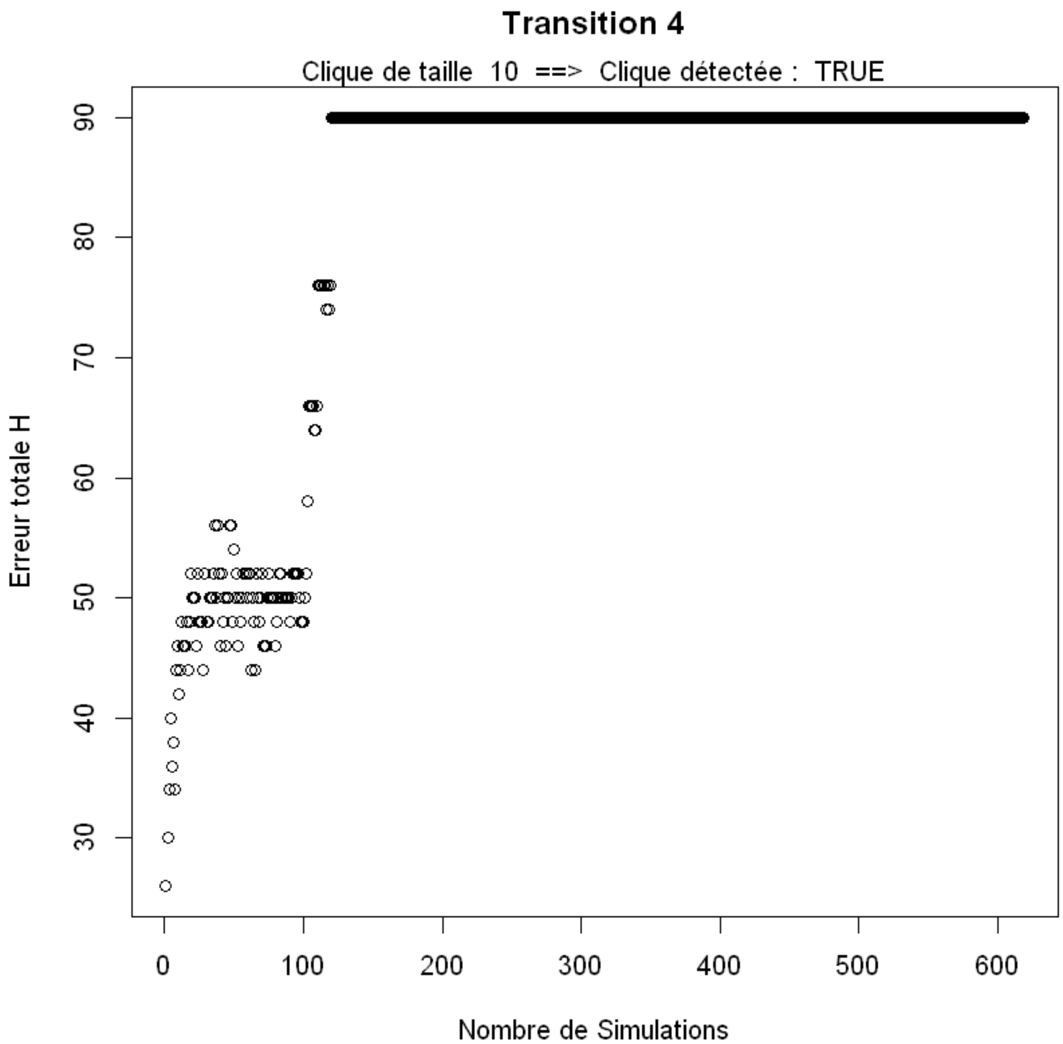
```
[1] 90
```

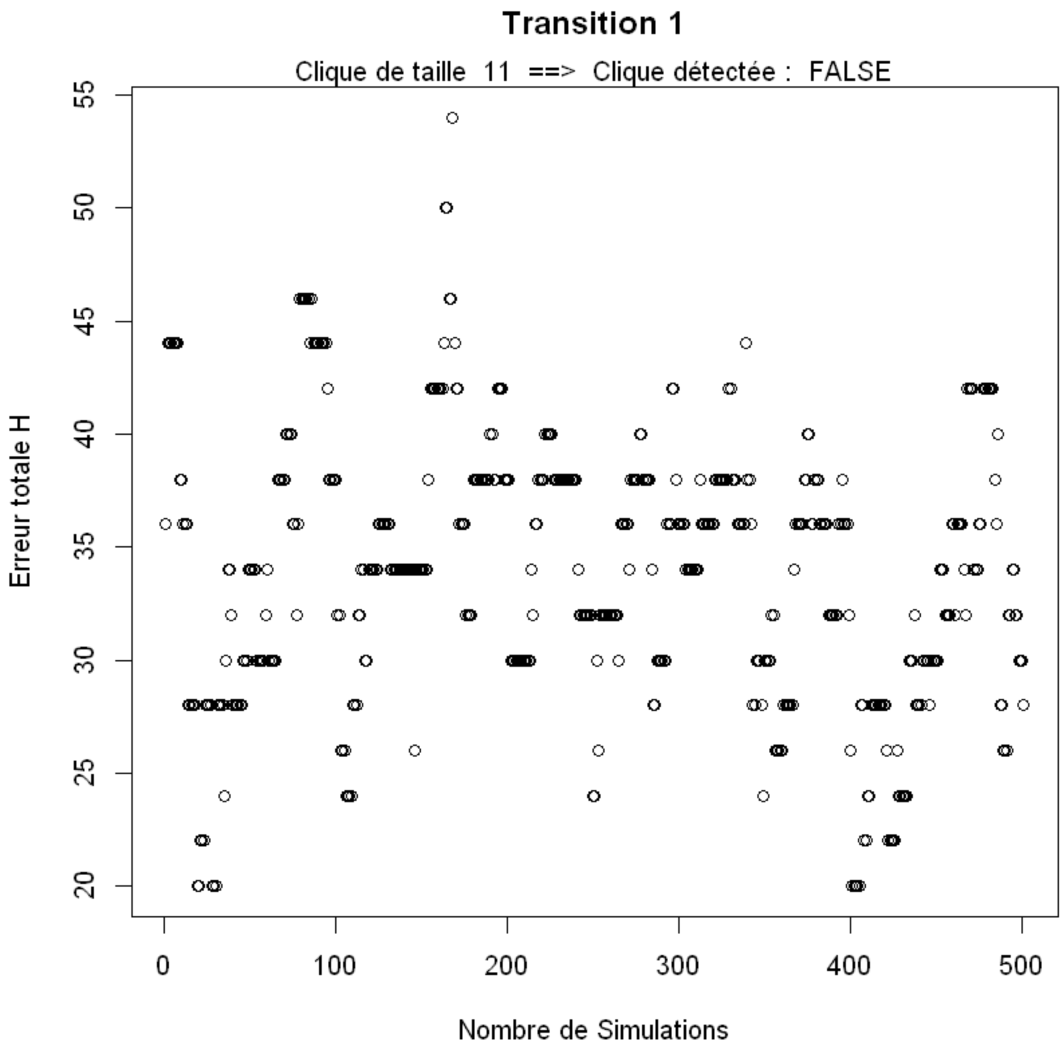

Transition 1
Clique de taille 10 ==> Clique détectée : FALSE
Erreur totale H
Nombre de Simulations
15
20
25
30
35
40
0
100
200
300
400
500

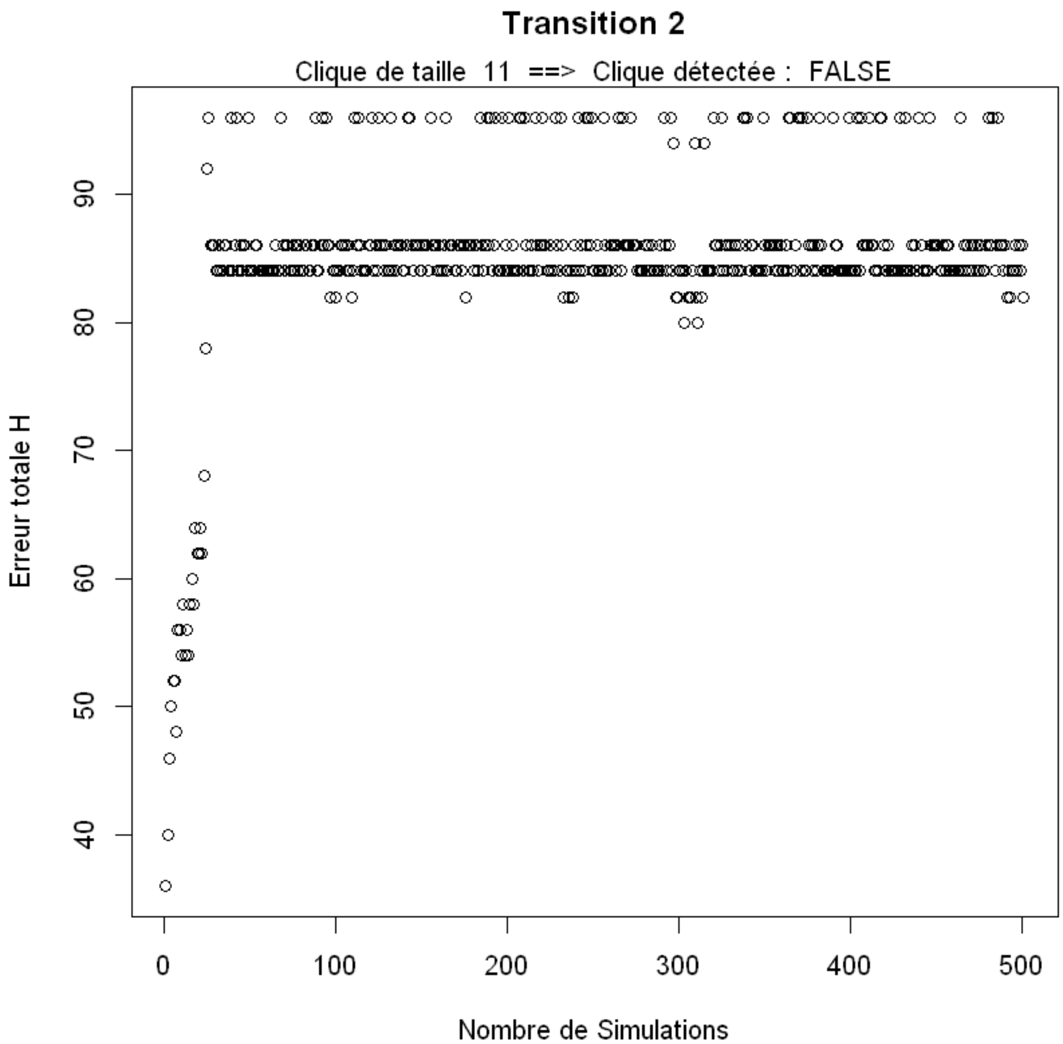

Transition 2
Clique de taille  10  ==>  Clique détectée :  TRUE
Erreur totale H
Nombre de Simulations
90
80
70
60
50
40
30
0
100
200
300
400
500

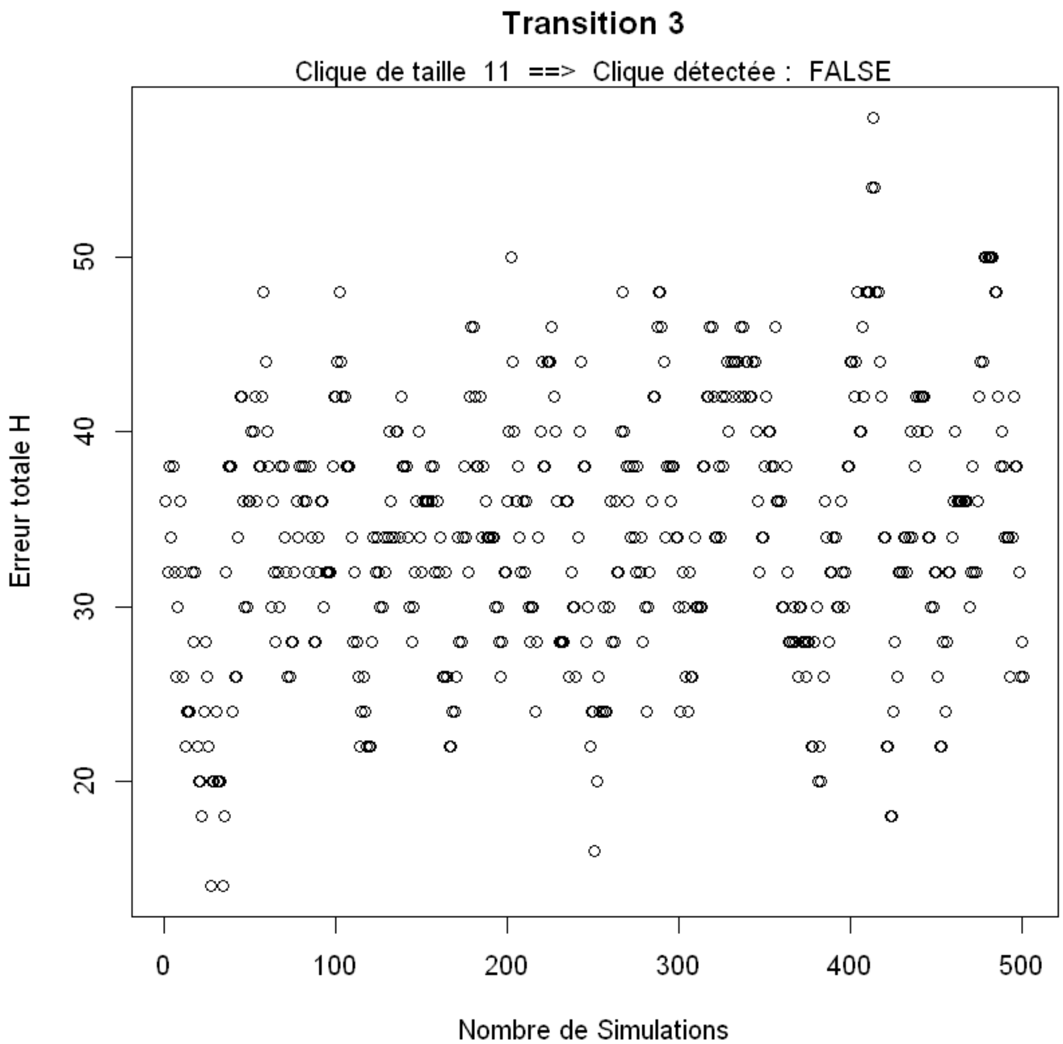
```
[1] 90
```


Transition 3
Clique de taille 10 ==> Clique détectée : FALSE
Erreur totale H
Nombre de Simulations
15
20
25
30
35
40
0
100
200
300
400
500

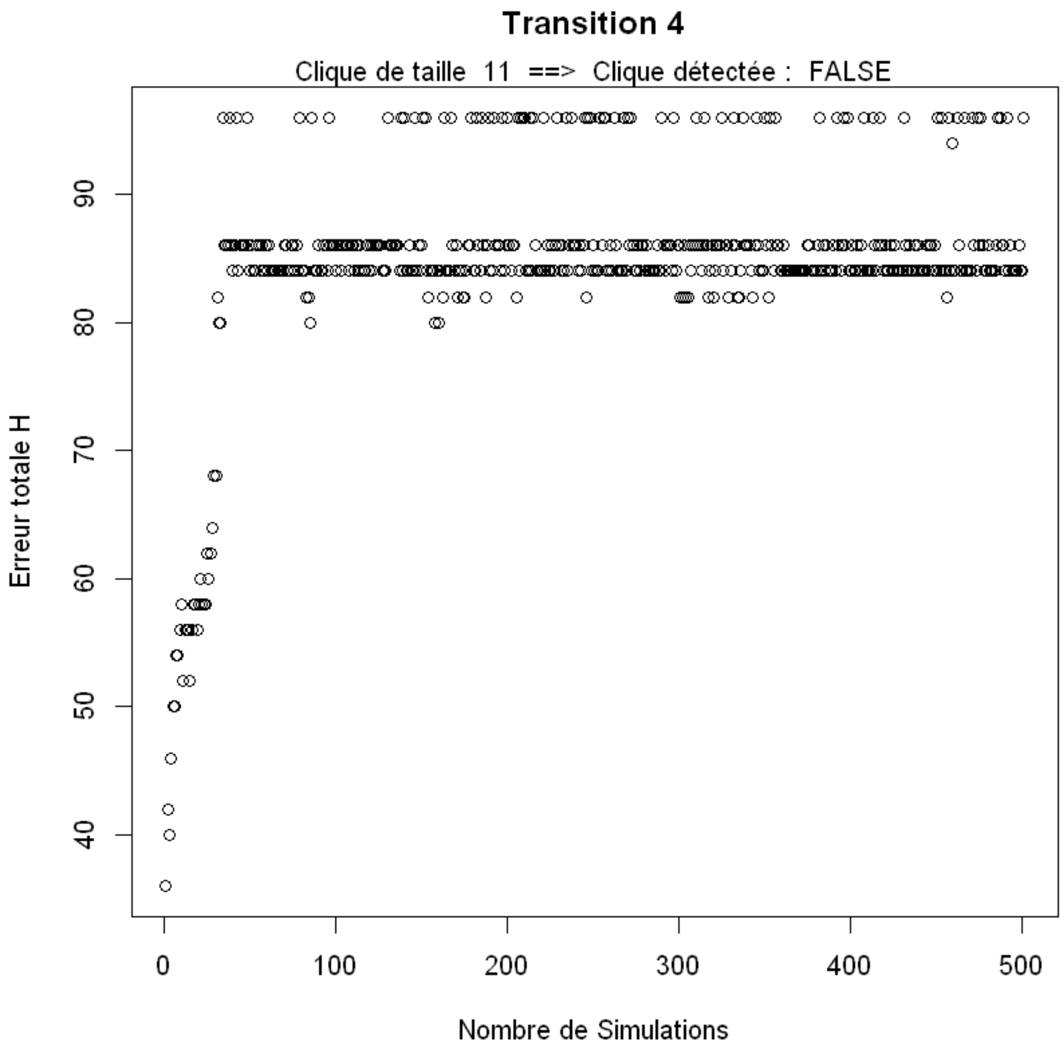

Transition 4
Clique de taille 10 ==> Clique détectée : TRUE
Erreur totale H
Nombre de Simulations
90
80
70
60
50
40
30
0
100
200
300
400
500
600


Transition 1
Clique de taille 11 ==> Clique détectée : FALSE
Erreur totale H
Nombre de Simulations
20
25
30
35
40
45
50
55
0
100
200
300
400
500


Transition 2
Clique de taille 11 ==> Clique détectée : FALSE
Erreur totale H
Nombre de Simulations
40
50
60
70
80
90
0
100
200
300
400
500




```
package 'xtable' successfully unpacked and MD5 sums checked

The downloaded binary packages are in
        C:\Users\Alexandre\AppData\Local\Temp\Rtmp8Ak28I\downloaded_packages
```




```
In [11]: xtable(output,caption = "Nombre d'itération avant convergence du modèle")
```

| | Transition 1 | Transition 2 | Transition 3 | Transition 4 |
|---|---|---|---|---|
| 3 | 6 | 2 | 55 | 2 |
| 4 | 500 | 3 | 69 | 3 |
| 5 | 500 | 283 | 500 | 144 |
| 6 | 500 | 98 | 500 | 448 |
| 7 | 500 | 50 | 500 | 60 |
| 8 | 500 | 500 | 500 | 144 |
| 9 | 500 | 500 | 500 | 17 |
| 10 | 500 | 40 | 500 | 119 |
| 11 | 500 | 500 | 500 | 500 |

- Création du vecteur max_clique

In [12]:
```
# On récupère le résultat de l'optimisation pour la transition 1
#clique_star1 = G_opt1
clique_star1 = unlist(G_opt1_list[nb_max_clique1 - (start_clique_sz-1)])
clique_star1 = which(clique_star1 == 1)

# On construit le vecteur max_clique
max_clique1 = rep(0, Nb_pts)
id = 1:Nb_pts
is_max_clique1 = rep(0, Nb_pts)
for(i in 1:Nb_pts)
{
    if (is.element(id[i], clique_star1))
    {
            is_max_clique1[i] = 1
    }
}
max_clique1 = cbind(id, is_max_clique1)

# On récupère le résultat de l'optimisation pour la transition 2
#clique_star2 = G_opt2
clique_star2 = unlist(G_opt2_list[nb_max_clique2 - (start_clique_sz-1)])
clique_star2 = which(clique_star2 == 1)

# On construit le vecteur max_clique
max_clique2 = rep(0, Nb_pts)
id = 1:Nb_pts
is_max_clique2 = rep(0, Nb_pts)
for(i in 1:Nb_pts)
{
    if (is.element(id[i], clique_star2))
    {
            is_max_clique2[i] = 1
    }
}
max_clique2 = cbind(id, is_max_clique2)

# On récupère le résultat de l'optimisation pour la transition 3
#clique_star3 = G_opt3
clique_star3 = unlist(G_opt3_list[nb_max_clique3 - (start_clique_sz-1)])
clique_star3 = which(clique_star3 == 1)

# On construit le vecteur max_clique
max_clique3 = rep(0, Nb_pts)
id = 1:Nb_pts
is_max_clique3 = rep(0, Nb_pts)
for(i in 1:Nb_pts)
{
    if (is.element(id[i], clique_star3))
```

```
    {
            is_max_clique3[i] = 1
    }
}
max_clique3 = cbind(id, is_max_clique3)

# On récupère le résultat de l'optimisation pour la transition 4
#clique_star4 = G_opt4
clique_star4 = unlist(G_opt4_list[nb_max_clique4 - (start_clique_sz-1)])
clique_star4 = which(clique_star4 == 1)

# On construit le vecteur max_clique
max_clique4 = rep(0, Nb_pts)
id = 1:Nb_pts
is_max_clique4 = rep(0, Nb_pts)
for(i in 1:Nb_pts)
{
    if (is.element(id[i], clique_star4))
    {
            is_max_clique4[i] = 1
    }
}
max_clique4 = cbind(id, is_max_clique4)
max_clique4
```

| id | is_max_clique4 |
|---|---|
| 1 | 0 |
| 2 | 0 |
| 3 | 1 |
| 4 | 0 |
| 5 | 0 |
| 6 | 0 |
| 7 | 0 |
| 8 | 0 |
| 9 | 1 |
| 10 | 0 |
| 11 | 1 |
| 12 | 0 |
| 13 | 0 |
| 14 | 0 |
| 15 | 0 |
| 16 | 0 |
| 17 | 0 |
| 18 | 0 |
| 19 | 0 |
| 20 | 0 |
| 21 | 0 |
| 22 | 0 |
| 23 | 0 |
| 24 | 0 |
| 25 | 0 |
| 26 | 0 |
| 27 | 1 |
| 28 | 0 |
| 29 | 0 |
| 30 | 1 |
| 31 | 1 |
| 32 | 1 |
| 33 | 1 |
| 34 | 0 |
| 35 | 1 |
| 36 | 0 |
| 37 | 0 |
| 38 | 1 |
| 39 | 0 |
| 40 | 0 |

### 4.0.4 On affiche le Graph de Points avec igraph

- Création du vecteur de couleur des points du graphe

A chaque ID correspondant à un point du graph, on associe un couleur en fonction de si le point fait partie de la clique maximum ou non. Couleur verte si le point fait partie de la clique maximum Couleur rouge sinon

```
In [13]: # On crée le vecteurs des coleurs correspondant à chaque point du graphe pour la tran
         colors1 = rep(1, Nb_pts)
         for (i in 1:Nb_pts)
         {

             if (is.element(i, clique_star1))
             {
                 # Si l'id de l'élement fait partie de la clique maximum, on l'affiche en vert
                 colors1[i] = "green"
             }
             else
             {
                 # Sinon, on l'affiche en rouge
                 colors1[i] = "red"
             }
         }

         # On crée le vecteurs des coleurs correspondant à chaque point du graphe pour la tran
         colors2 = rep(1, Nb_pts)
         for (i in 1:Nb_pts)
         {

             if (is.element(i, clique_star2))
             {
                 # Si l'id de l'élement fait partie de la clique maximum, on l'affiche en vert
                 colors2[i] = "green"
             }
             else
             {
                 # Sinon, on l'affiche en rouge
                 colors2[i] = "red"
             }
         }

         # On crée le vecteurs des coleurs correspondant à chaque point du graphe pour la tran
         colors3 = rep(1, Nb_pts)
         for (i in 1:Nb_pts)
         {

             if (is.element(i, clique_star3))
             {
                 # Si l'id de l'élement fait partie de la clique maximum, on l'affiche en vert
                 colors3[i] = "green"
             }
             else
             {
```

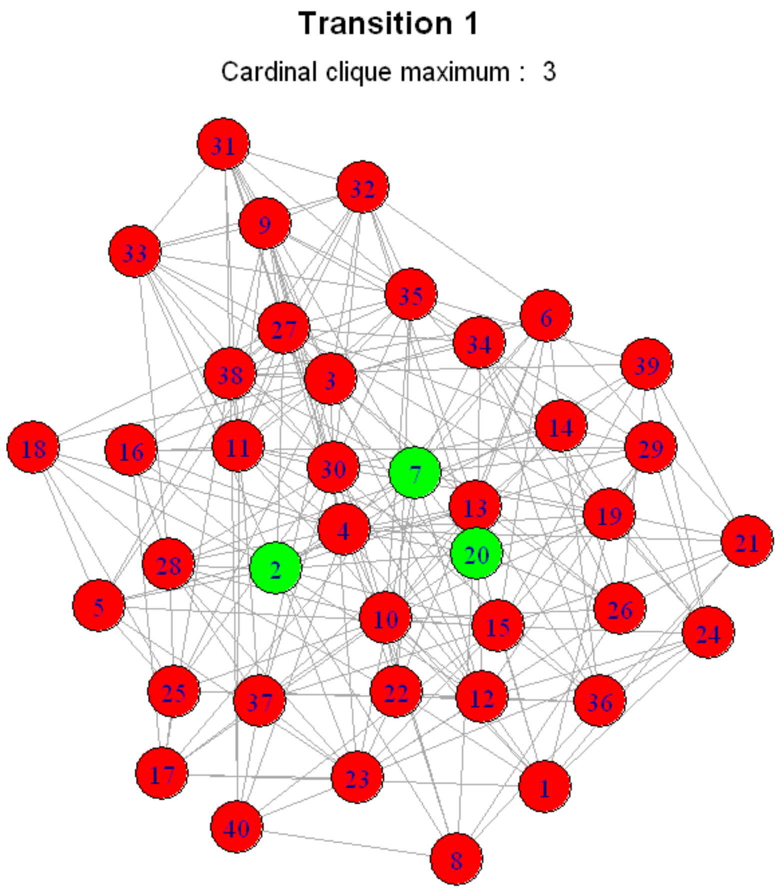
```
            # Sinon, on l'affiche en rouge
            colors3[i] = "red"
        }
    }

    # On crée le vecteurs des coleurs correspondant à chaque point du graphe pour la tran
    colors4 = rep(1, Nb_pts)
    for (i in 1:Nb_pts)
    {

        if (is.element(i, clique_star4))
        {
            # Si l'id de l'élement fait partie de la clique maximum, on l'affiche en vert
            colors4[i] = "green"
        }
        else
        {
            # Sinon, on l'affiche en rouge
            colors4[i] = "red"
        }
    }
```

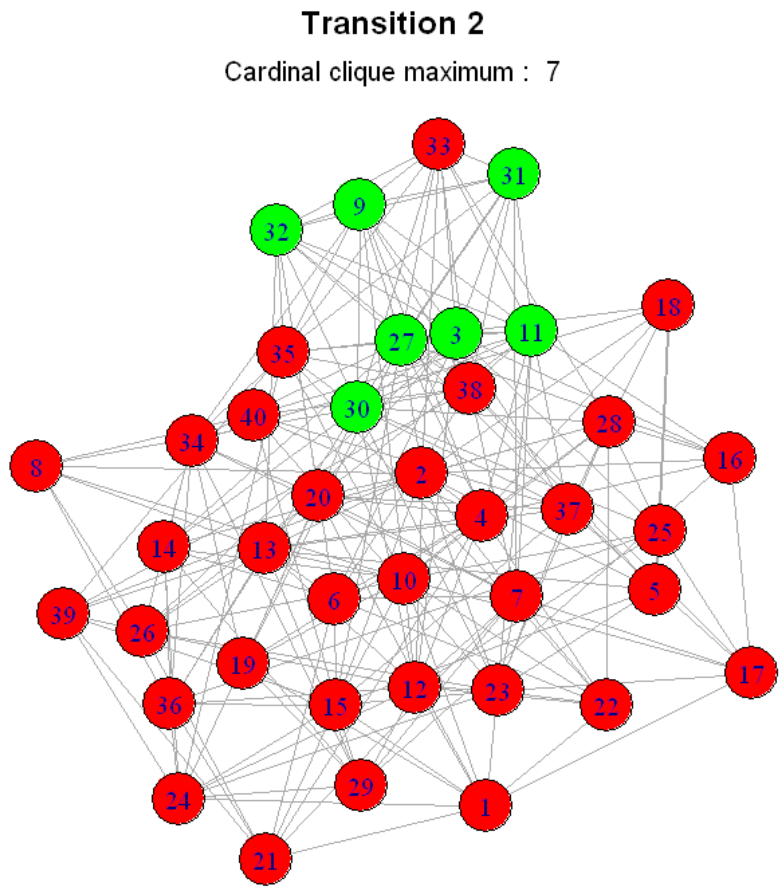
On génère le graph à l'aide du Vecteur X simulé, et du vecteur max_clique (résultat de Metropolis)

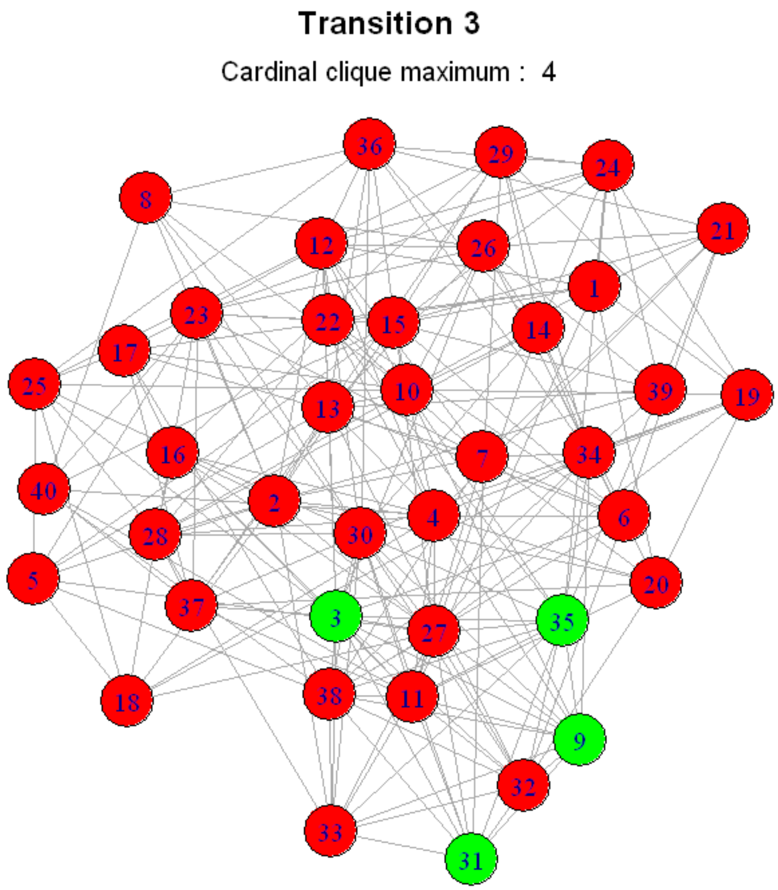
```
In [14]: library(igraph)
         answers1 = max_clique1
         answers2 = max_clique2
         answers3 = max_clique3
         answers4 = max_clique4
         topology = X

         g1 = graph.data.frame(topology, vertices=answers1, directed=FALSE)
         graph1 <- simplify(g1)

         plot.igraph(graph1, vertex.color=colors1, main="Transition 1")
         mtext(paste("Cardinal clique maximum : ", nb_max_clique1))

         g2 = graph.data.frame(topology, vertices=answers2, directed=FALSE)
         graph2 <- simplify(g2)

         plot.igraph(graph2, vertex.color=colors2, main="Transition 2")
         mtext(paste("Cardinal clique maximum : ", nb_max_clique2))

         g3 = graph.data.frame(topology, vertices=answers3, directed=FALSE)
         graph3 <- simplify(g3)

         plot.igraph(graph3, vertex.color=colors3, main="Transition 3")
```

```r
        mtext(paste("Cardinal clique maximum : ", nb_max_clique3))

        g4 = graph.data.frame(topology, vertices=answers4, directed=FALSE)
        graph4 <- simplify(g4)

        plot.igraph(graph4, vertex.color=colors4, main="Transition 4")
        mtext(paste("Cardinal clique maximum : ", nb_max_clique4))
```

```
Attaching package: 'igraph'

The following objects are masked from 'package:stats':

    decompose, spectrum

The following object is masked from 'package:base':

    union
```


Transition 1
Cardinal clique maximum : 3
31
32
9
33
35
6
27
34
39
38
3
14
18
16
11
29
30
7
13
19
4
21
28
2
20
5
10
15
26
24
25
37
22
12
36
17
23
1
40
8


Transition 2
Cardinal clique maximum : 7
33
31
9
32
18
27
3
11
35
38
30
40
28
34
16
8
2
20
37
4
25
14
13
10
5
7
6
39
26
19
17
36
12
23
22
15
29
24
1
21


Transition 3
Cardinal clique maximum : 4
36
29
24
8
12
26
21
1
23
22
15
14
17
25
10
13
39
19
16
7
34
40
2
4
6
28
30
5
20
37
3
27
35
18
38
11
9
32
33
31

**Transition 4**

Cardinal clique maximum : 10