nonames2

```
library(Matrix)
library(ggplot2)

#install.packages("hier.part")
library(hier.part)
```

```
## Loading required package: gtools
```

```
E <- function(W, s){
  summe <- 0
  for(i in 1:6){
    for(j in 1:6){
      summe <- summe + W[i,j]*s[i]*s[j]
    }
  }
  return(-0.5*summe)
}

E_of_si <- function(W, s, i, nachbarn){
  summe <- 0
    for(j in nachbarn){
      summe <- summe + W[i,j]*s[i]*s[j]
    }
  return(-0.5*summe)
}

P_func <- function(b, dE){
  return(1/(1+exp(b*dE)))
}



mean_field <- function(w, s, i){
  mean_field_value <- 0
  for(j in N){
    #diagonal is zero so k equals i is never calculated
    mean_field_value <- (w[i,j]*s[j]) + mean_field_value
  }
 return((-1)*mean_field_value)
}

state <- function(beta,mean_field_value){
  s <- tanh((-1)*beta*mean_field_value)
  return(s)
}
```

weight initalization and states

```
N <- 6

#assumption
set.seed(509)
W <- matrix(rnorm(N*N,0,1),N,N)
W[lower.tri(W)] = t(W)[lower.tri(W)]
diag(W) = 0

# set s randomly
set.seed(123)
s0 <- sample(c(-1,1),6, replace = TRUE)
```

8.1 Simulated Annealing

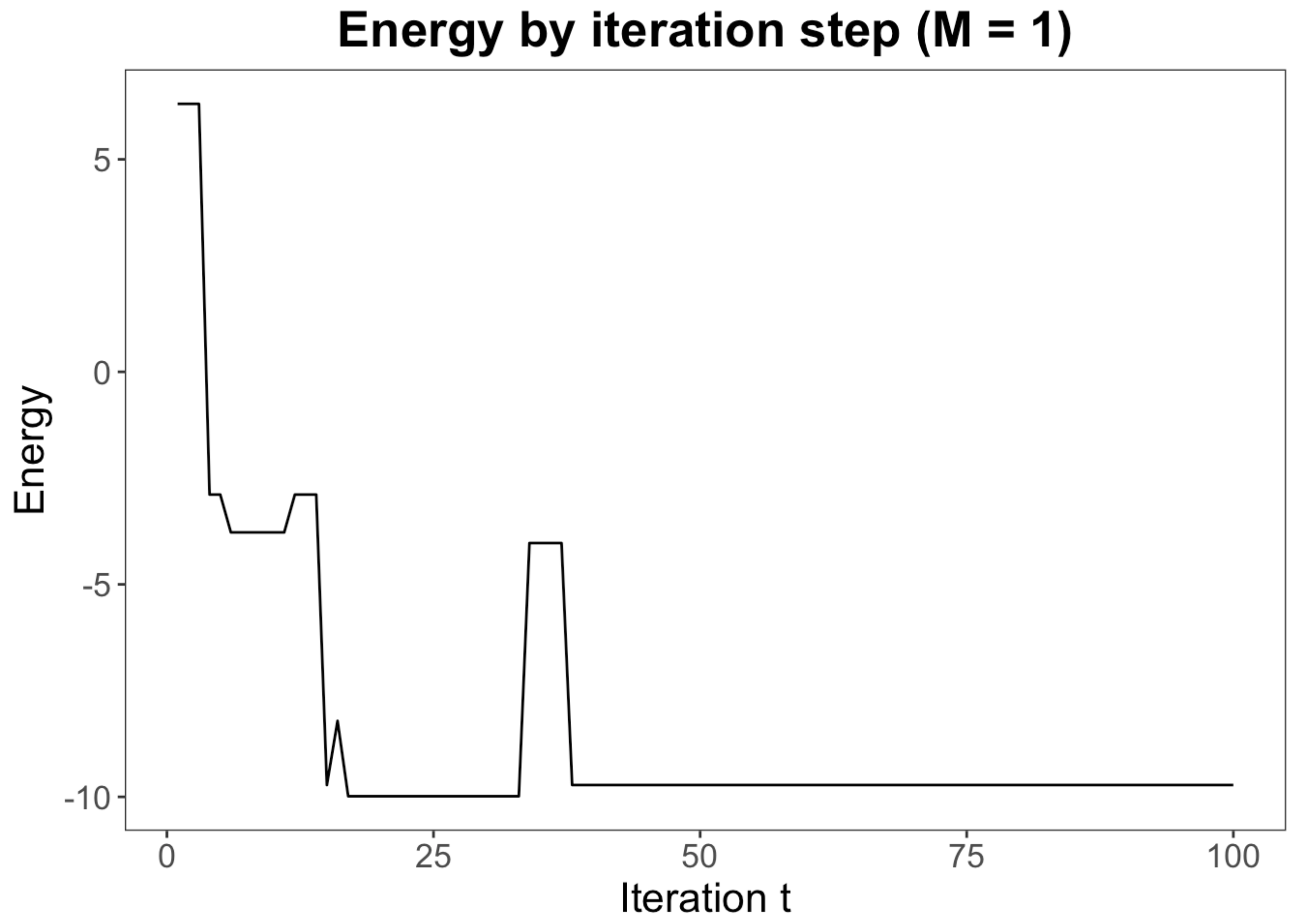
```
## Initialization
beta0 <- 1/10
tau <- 1.1
t_max <- 100

# state update loop iterations iterations
M1 <- 1
M2 <- 500

## Optimization
neighborhelper <- 1:6
EnergyM1 <- matrix(nrow = 1, ncol = t_max)
EnergyM2 <- matrix(nrow = 1, ncol = t_max)
```

State Update Loop with M = 1 Iterations

```r
start.time <- Sys.time()
for(i in 1:t_max){
  if(i == 1){
    s <- s0
    beta <- beta0}
  for(j in 1:M1){
    node <- sample(c(1:6),1)
    neighbors <- neighborhelper[-node]

    E_si <- E_of_si(W, s, node, neighbors)

    E_minus_si <- -E_si

    delta_E <- E_minus_si - E_si

    if(P_func(beta, delta_E)>=runif(1,0,1)){
      s[node] <- (-1)*s[node]}
  } # end state update loop
  beta <- beta*tau
  EnergyM1[i] <- E(W, s)
} #end annealing loop
sM1 <- s


end.time <- Sys.time()
time.taken_M1 <- end.time - start.time
```

Plot m1 iterations

```r
energy_plot_data <- data.frame(t = 1:t_max, e1 = as.vector(EnergyM1))

ggplot(energy_plot_data) + geom_line(aes(x = t, y = e1)) +
    labs(x = "Iteration t", y = "Energy") +
    ggtitle("Energy by iteration step (M = 1)") +
    theme_bw()  +
    theme(panel.grid.major = element_blank(),
          panel.grid.minor = element_blank(),
          legend.key = element_rect(colour = "black"),
          plot.title = element_text(face="bold", hjust = .5)) +
    theme(text = element_text(size = 16))
```

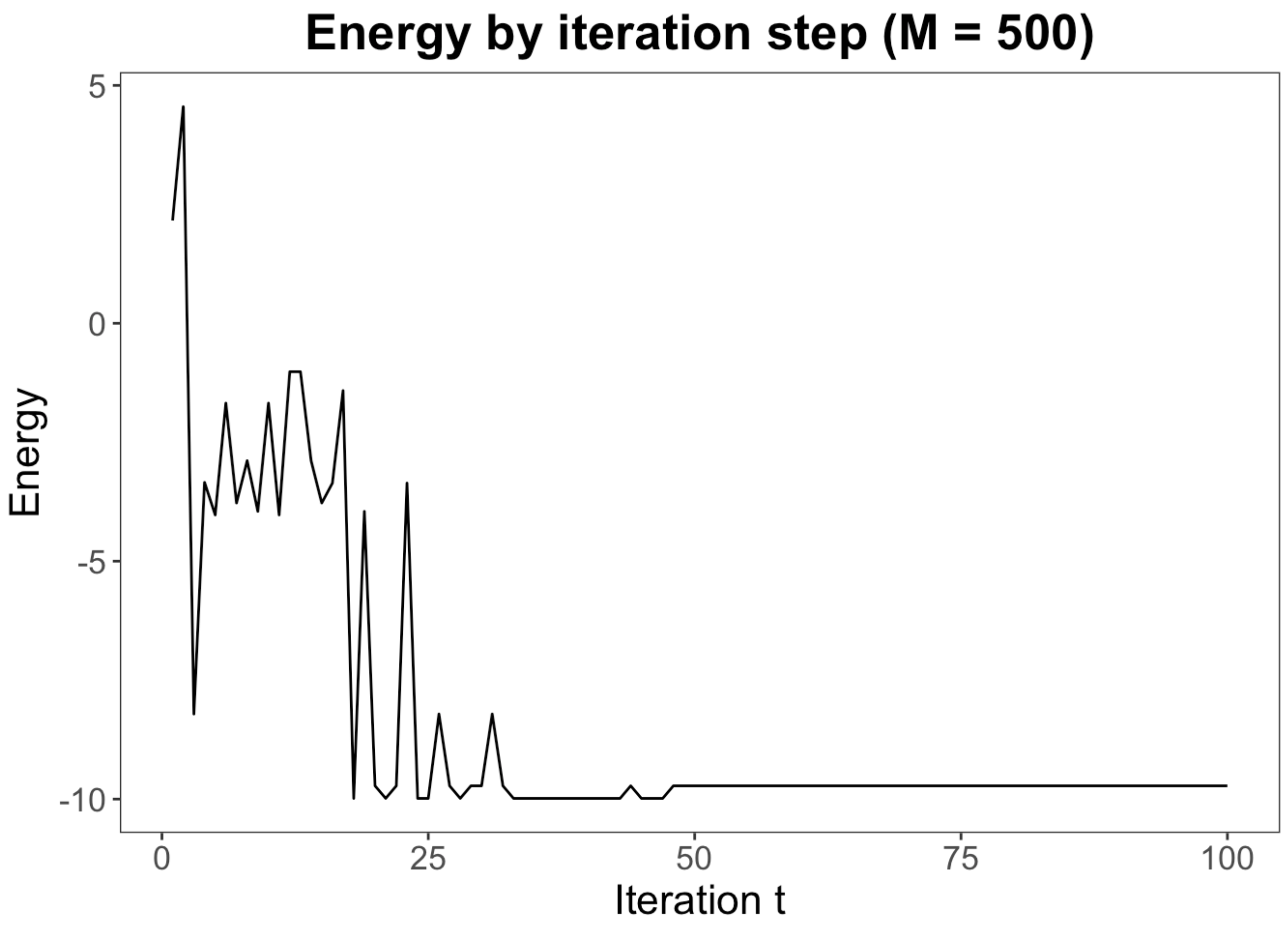


State Update Loop with M = 500 Iterations

```
start.time <- Sys.time()

for(i in 1:t_max){
  if(i == 1){
    s <- s0
    beta <- beta0}
  for(j in 1:M2){
    node <- sample(c(1:6),1)
    neighbors <- neighborhelper[-node]

    E_si <- E_of_si(W, s, node, neighbors)

    E_minus_si <- -E_si

    delta_E <- E_minus_si - E_si

    if(P_func(beta, delta_E)>=runif(1,0,1)){
      s[node] <- (-1)*s[node]}
  } # end state update loop
  beta <- beta*tau
  EnergyM2[i] <- E(W, s)
} #end annealing loop
sM2 <- s

end.time <- Sys.time()
time.taken_M2 <- end.time - start.time
```

compare results

```
energy_plot_data$e2 = as.vector(EnergyM2)

ggplot(energy_plot_data) + geom_line(aes(x = t, y = e2)) +
    labs(x = "Iteration t", y = "Energy") +
    ggtitle("Energy by iteration step (M = 500)") +
    theme_bw()  +
    theme(panel.grid.major = element_blank(),
          panel.grid.minor = element_blank(),
          legend.key = element_rect(colour = "black"),
          plot.title = element_text(face="bold", hjust = .5)) +
    theme(text = element_text(size = 16))
```



compare results

```
E(W, s0)
```

```
## [1] 6.305124
```

```
E(W, sM1)
```

```
## [1] -9.721218
```

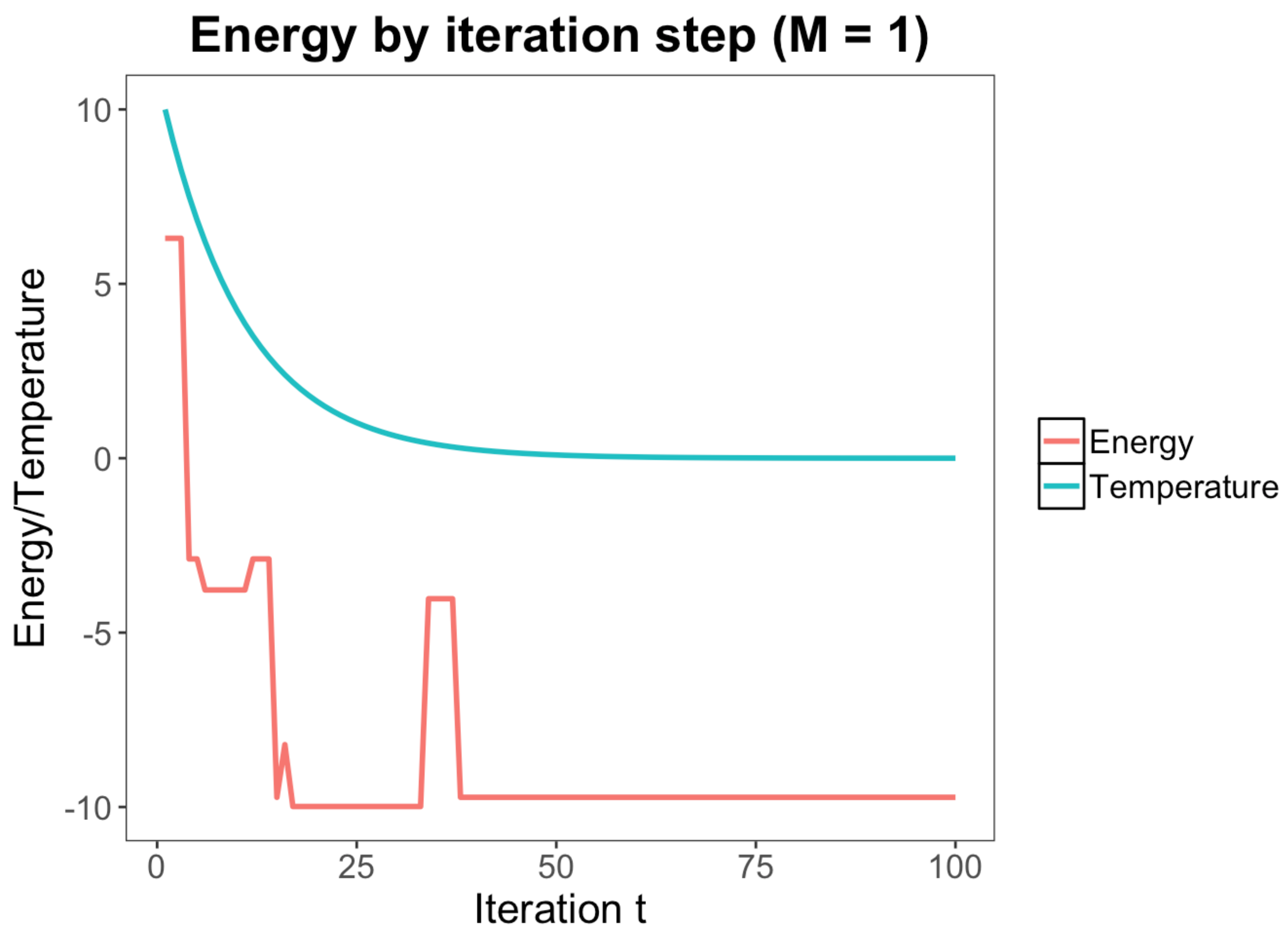
```
E(W, sM2)
```

```
## [1] -9.721218
```

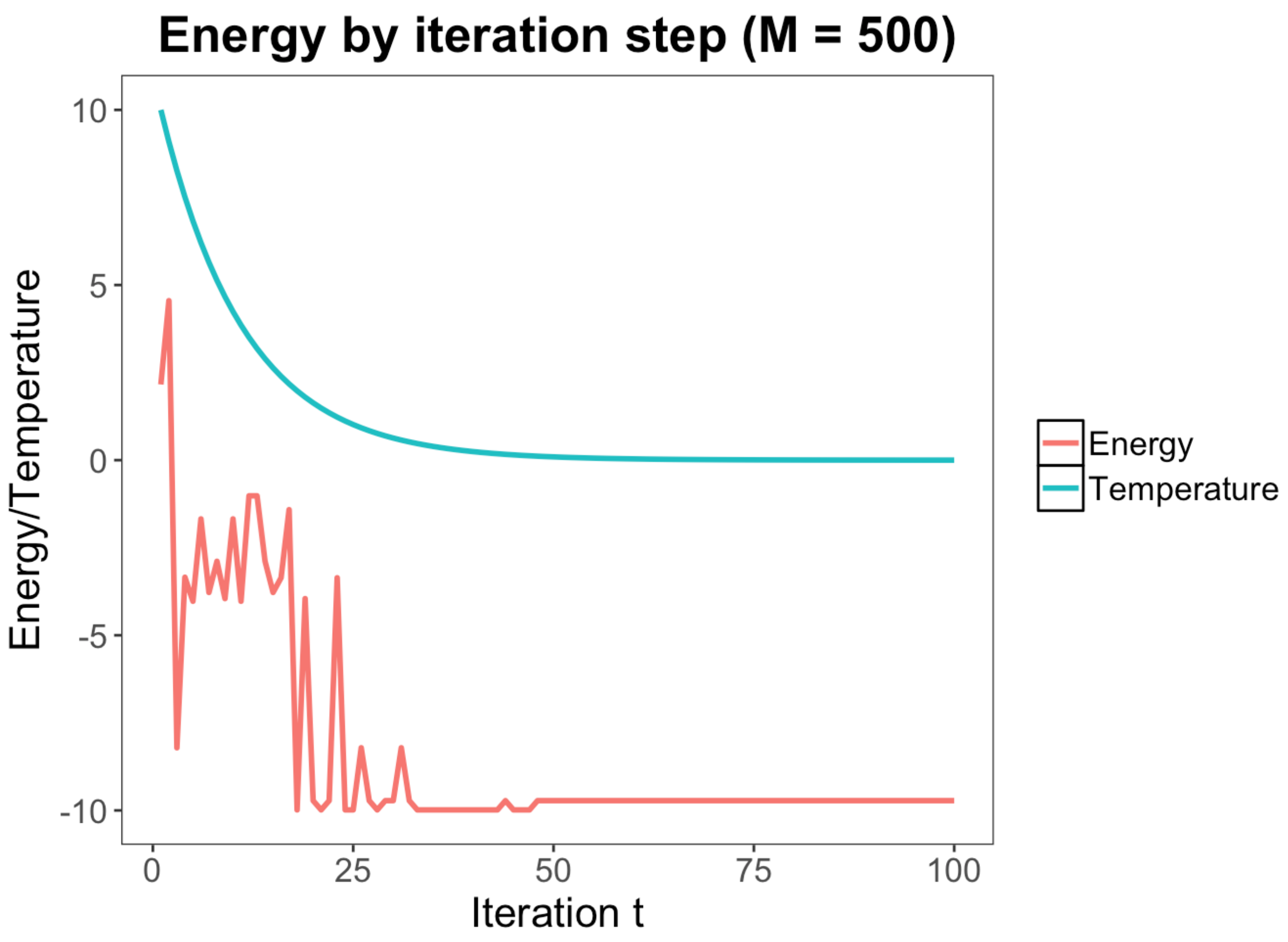
plots 8.1

```
# Plots
# values for beta for every iteration
betas <- matrix(nrow = t_max, ncol = 1)
betas[1] <- beta0
for(i in 1:(t_max-1)){
  betas[i+1] <- betas[i]*tau
}
```

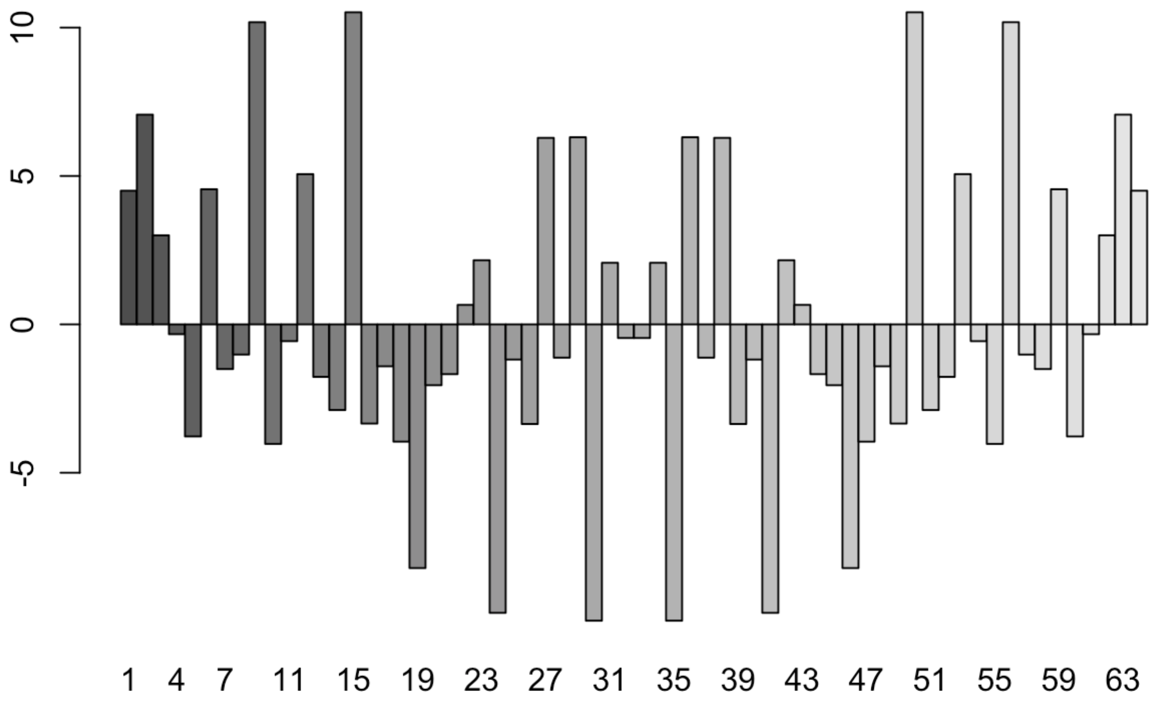
```
energy_tmp_plot_data = data.frame(t = rep(1:t_max, 2),
                                  l1 = c(as.vector(EnergyM1), as.vector(1/betas)),
                                  l2 = c(as.vector(EnergyM2), as.vector(1/betas)),
                                  group = factor(rep(c("Energy", "Temperature"), each
 = t_max)))

ggplot(energy_tmp_plot_data) +
    geom_line(aes(x = t, y = l1, color = group), size = 1) +
    labs(x = "Iteration t", y = "Energy/Temperature") +
    ggtitle("Energy by iteration step (M = 1)") +
    theme_bw()  +
    theme(panel.grid.major = element_blank(),
          panel.grid.minor = element_blank(),
          legend.key = element_rect(colour = "black"),
          plot.title = element_text(face="bold", hjust = .5)) +
    theme(text = element_text(size = 16)) +
    theme(legend.title=element_blank())
```



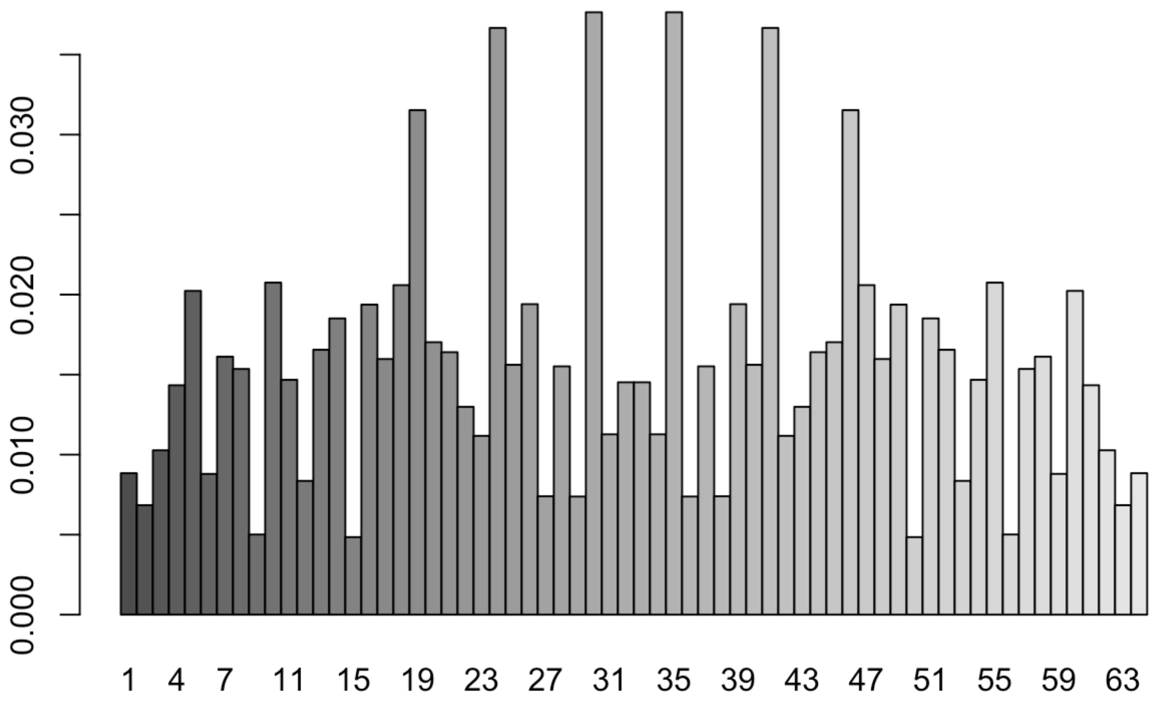
```
ggplot(energy_tmp_plot_data) +
    geom_line(aes(x = t, y = l2, color = group), size = 1) +
    labs(x = "Iteration t", y = "Energy/Temperature") +
    ggtitle("Energy by iteration step (M = 500)") +
    theme_bw()  +
    theme(panel.grid.major = element_blank(),
          panel.grid.minor = element_blank(),
          legend.key = element_rect(colour = "black"),
          plot.title = element_text(face="bold", hjust = .5)) +
    theme(text = element_text(size = 16)) +
    theme(legend.title=element_blank())
```



```
# Matrix with all states of s1,...s6
States <- rbind(rep(0,6), combos(6)$binary) # combos-function from package hier.part
States[States == 0] <- -1
# fill in Energy of states
E_all <- matrix(nrow = 64, ncol = 1)

for(i in 1:length(E_all)){
    E_all[i] <- E(W, States[i,])
}

# fill in Probability of chosing a state with fixed beta
P_all <- matrix(nrow = 64, ncol = 1)
Z <- 0
for(i in 1:64){
    Z <- Z + exp(-beta0*E(W, States[i,]))
}
for(i in 1:length(P_all)){
    P_all[i] <- (exp(-beta0*E(W, States[i,])))/Z
}

# bar plot of Energy of all states and its Probability
barplot(E_all, beside = TRUE, names.arg = 1:64)
```



```
barplot(P_all, beside = TRUE, names.arg = 1:64)
```



8.2 Simulated Annealing

beta, tau, tmax, state, e

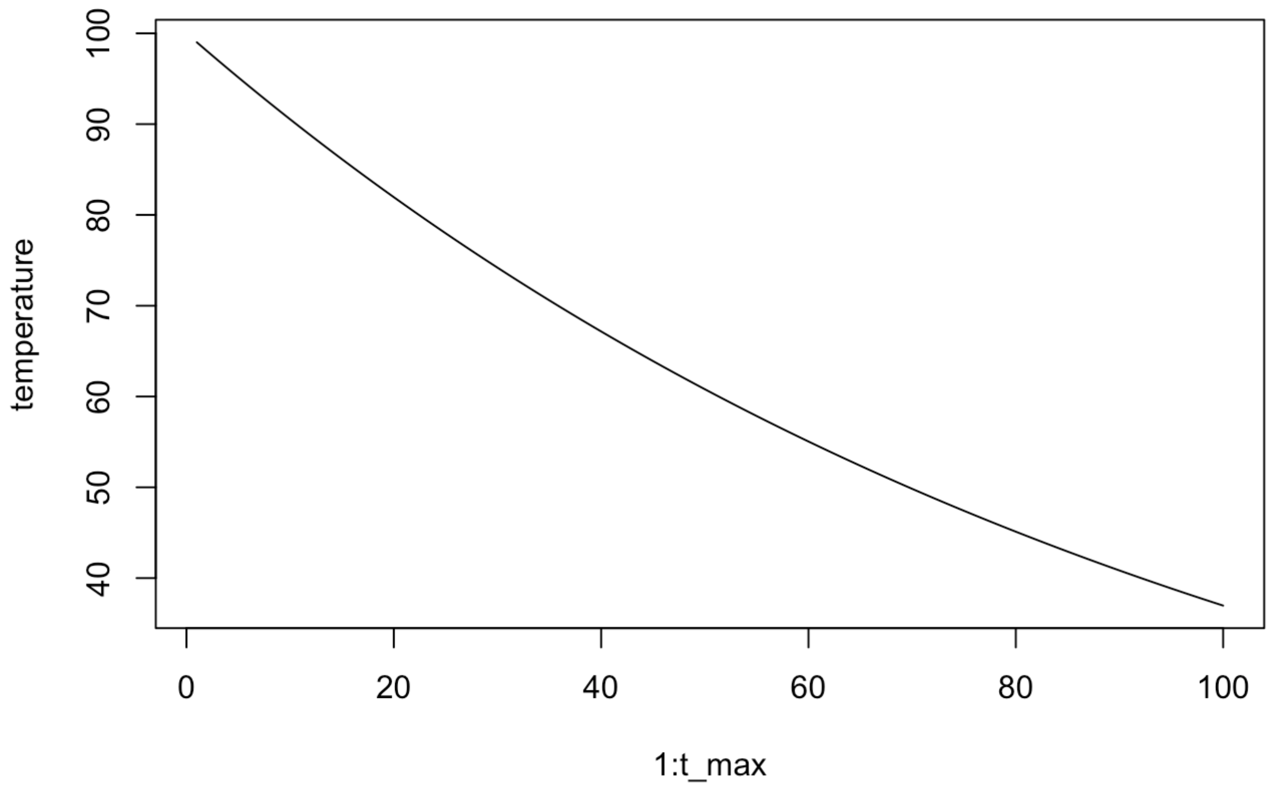
```
beta <- 0.01
tau <- 1.01
t_max <- 100
epsilon <- 0.001
s = sample(c(-1,1), 6, replace = TRUE)
N = length(s)
e_diff = 1
i = 1
e_old = 0

temperature = rep(0, t_max)
energy = rep(0, t_max)
```

mean-field algorithm

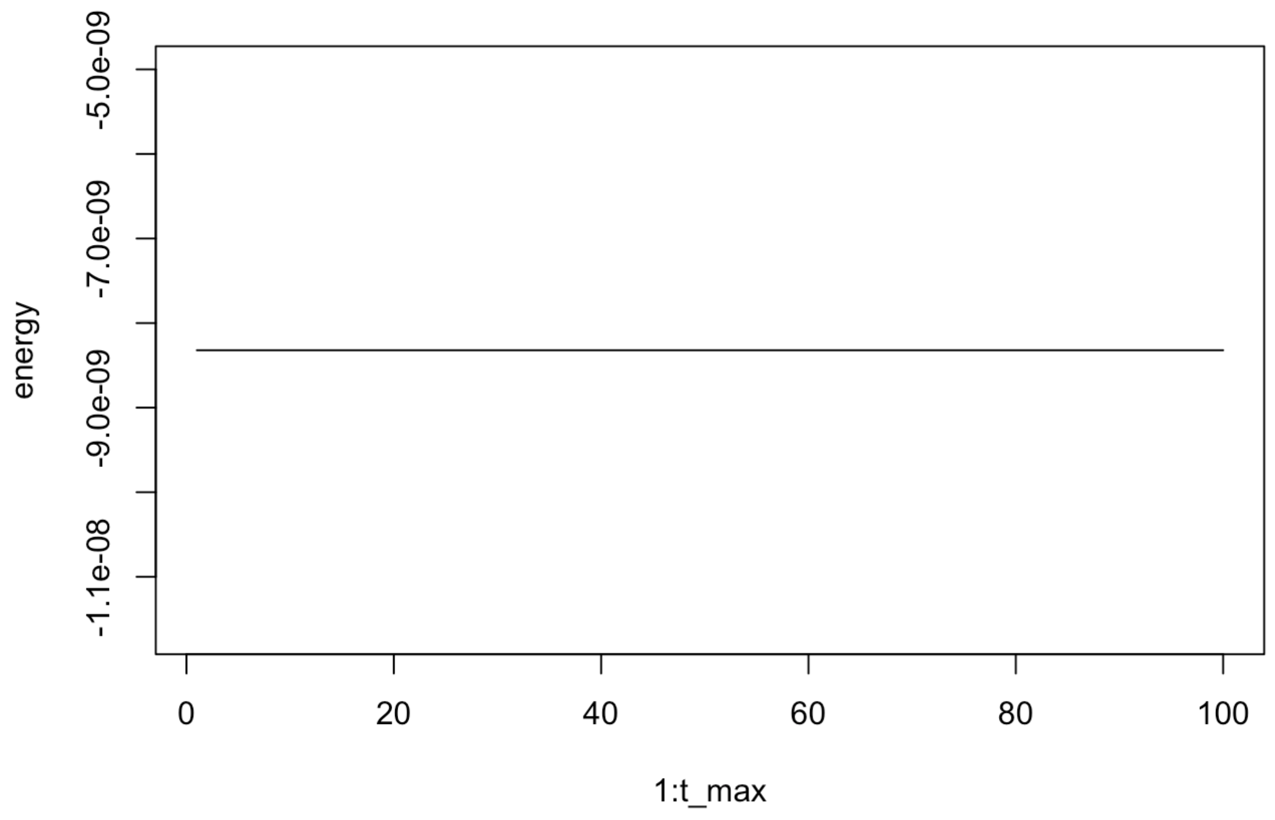
```
start.time <- Sys.time()

for(t in 1:t_max){

    while(epsilon < e_diff){
        #j = j+1
        N_j = setdiff(1:N, i)
        e_i = -sum(W[i, N_j] * s[N_j])
        s[i] = tanh(-beta*e_i)

        e_diff = abs(e_i - e_old)
        e_old = e_i

        if(i == 6){
            i = 1
        } else{
            i = i+1
        }

    }
    beta = beta*tau
    temperature[t] = 1/beta
    energy[t] = E(W, s)
}

end.time <- Sys.time()
time.taken_meanfield <- end.time - start.time
```

Plot the temperature Tt = 1 and the energy Et over the iterations t = 0, . . . , tmax.

```
plot(1:t_max, temperature, type = "l")
```



```
plot(1:t_max, energy, type = "l")
```



Run time comparison for

mean field column 1: tau <- 1.01 epsilon <- 0.001

simulated M 500 column 2: tau <- 1.1 simulated M 1 column 3: tau <- 1.1

```
run_time_comparison <- matrix(data = c(time.taken_meanfield, time.taken_M2, time.take
n_M1), nrow = 1, ncol = 3)

run_time_comparison
```

```
##             [,1]     [,2]        [,3]
## [1,] 0.007441044 0.880043 0.009968042
```

Comparison of s mean field column 1: tau <- 1.01 epsilon <- 0.001

simulated M 500 column 2: tau <- 1.1

simulated M 1 column 3: tau <- 1.1

```
s_comparison <- matrix(data = c(s, sM2, sM1), nrow = 6, ncol = 3)
s_comparison
```

```
##               [,1] [,2] [,3]
## [1,]  7.345854e-05    1    1
## [2,] -1.993920e-04    1    1
## [3,] -2.489114e-04   -1   -1
## [4,] -8.075207e-06    1    1
## [5,]  5.243286e-06   -1   -1
## [6,] -1.930561e-06   -1   -1
```