Computational Biology [HW4]

R06849014　張宏卿

Q1:基因演算法使用兩點交配

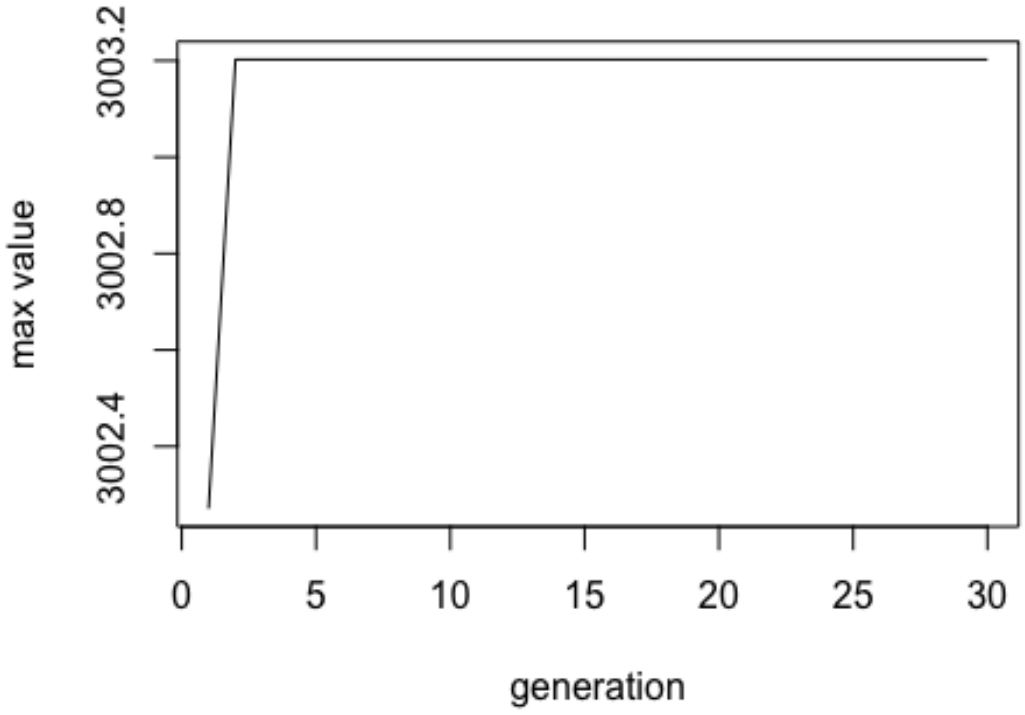
基本設定: 30 個個體, 重複 30 代, 單點突變, 突變機率 0.05, 一個個體請設計為 8 個位元

在這個設定下, 大概在第 5 世代左右就會收斂！

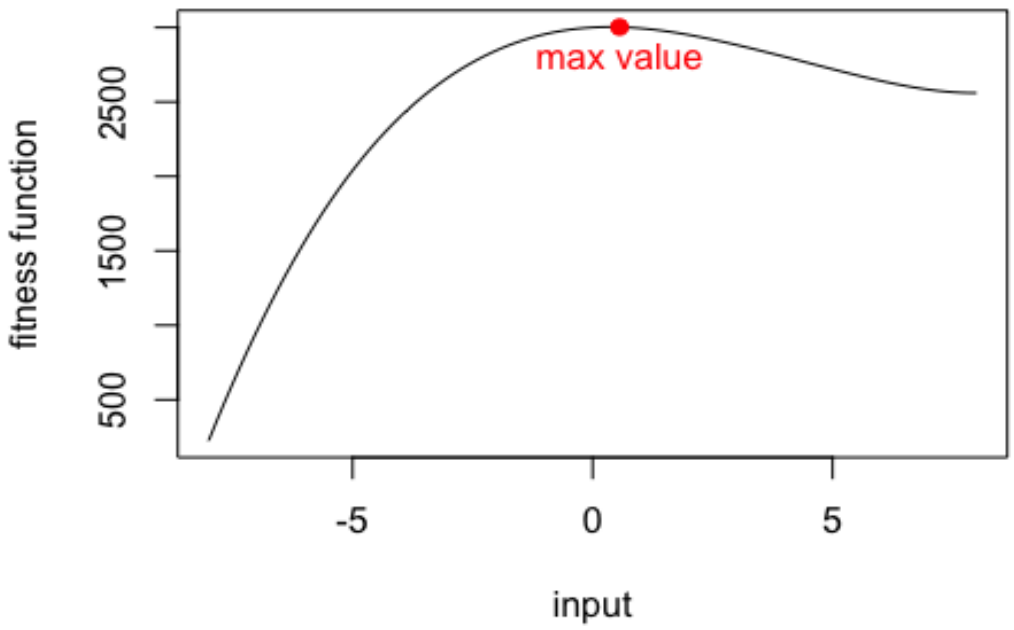
另外, 雖然只有用 8 位元來設計,但是從第 2 張圖可以看出, 演算法得到的最大值與實際最大值差不了多少.

```
N_size <- 30   #個體數
N_bit <- 8
Gen <- 30
Population <- matrix(sample(c(0,1), size = N_size*N_bit, replace = T),
                     N_size, N_bit)
mut_freq <- 0.05
#2 進位到 10 進位
Po_value <- 8*Population[,1] + 4*Population[,2] + 2*Population[,3] + 1*
Population[,4] + 1/2*Population[,5] + 1/4*Population[,6] +1/8*Populatio
n[,7] + 1/16*Population[,8]
Po_value <- Po_value - 8
#fitness function
fit <- function(x){
  2*x^3 - 25*x^2 + 18*x + 3000 - x*sin(x)
}
fit_value <- fit(Po_value)

#第一世代
idx <- 1:N_size
max_Gen_value <- rep(0, Gen)
max_Gen_value[1] <- max(fit_value)
max_Gen_ind <- matrix(0, Gen, N_bit)
max_Gen_ind[1,] <- Population[which(fit_value == max(fit_value))[1],]
max_Gen_Po_value <- rep(0, Gen)
max_Gen_Po_value[1] <- Po_value[which(fit_value == max(fit_value))[1]]

#GA
for(now_Gen in 2:Gen){
  child <- matrix(0, N_size, N_bit)
  child[1,] <- Population[which(fit_value == max(fit_value))[1],]
  switch_bit <- sample(N_bit, 1)
  child[2,] <- child[1,]
```

```
  child[2, switch_bit:N_bit] <- !child[2, switch_bit:N_bit]
  child_size <- 2
  total_wheel <- sum(fit_value)
  select_frequency <- fit_value/total_wheel

  #生子代
  while(child_size < N_size){
    P_idx <- sample(idx, size = 2, replace = F, prob = select_frequency)
    P1 <- Population[P_idx[1],]
    P2 <- Population[P_idx[2],]
    #兩點交配
    switch_bit <- sample(N_bit, 2)
    switch_bit <- sort(switch_bit)
    P1_new <- P1
    P1_new[switch_bit[1]:switch_bit[2]] <- P2[switch_bit[1]:switch_bit
[2]]
    P2_new <- P2
    P2_new[switch_bit[1]:switch_bit[2]] <- P1[switch_bit[1]:switch_bit
[2]]
    if(runif(1, 0, 1) < mut_freq){
      tar_bit <- sample(N_bit,1)
      P1_new[tar_bit] <- !P1_new[tar_bit]
    }
    if(runif(1, 0, 1) < mut_freq){
      tar_bit <- sample(N_bit,1)
      P2_new[tar_bit] <- !P2_new[tar_bit]
    }
    child[child_size + 1,] <- P1_new
    child[child_size + 2,] <- P2_new
    child_size <- child_size + 2
  }
  #記錄該代最佳解
  Population <- child
  Po_value <- 8*Population[,1] + 4*Population[,2] + 2*Population[,3] +
1*Population[,4] + 1/2*Population[,5] + 1/4*Population[,6] +1/8*Populat
ion[,7] + 1/16*Population[,8]
  Po_value <- Po_value - 8
  fit_value <- fit(Po_value)
  max_Gen_value[now_Gen] <- max(fit_value)
  max_Gen_Po_value[now_Gen] <- Po_value[which(fit_value == max(fit_valu
e))[1]]
  max_Gen_ind[now_Gen,] <- Population[which(fit_value == max(fit_value))
[1],]
}

#line plot
plot(max_Gen_value, type = "l",
     xlab = "generation", ylab = "max value")
```




```
#plot
curve(2*x^3 - 25*x^2 + 18*x + 3000 - x*sin(x), from = -8, to = 8,
      xlab = "input", ylab = "fitness function")
points(max(max_Gen_Po_value), max_Gen_value[max_Gen_Po_value == max(max
_Gen_Po_value)][1],
       col = "red", pch = 19)
text(max(max_Gen_Po_value), max_Gen_value[max_Gen_Po_value == max(max_G
en_Po_value)][1],
     labels = "max value", pos = 1, col = "red")
```

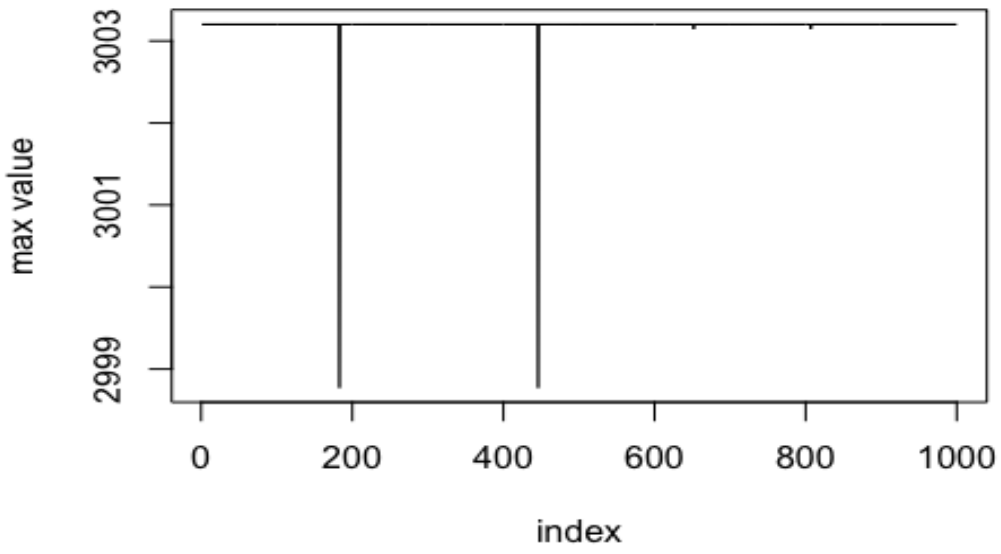


Q2: 重複跑程式碼 1000 次, 比較每次結果是否相同？

記錄每次跑程式碼得到的最大值, 觀察在這樣的參數設定下, 是否每次都會收斂到可得到的最大值, 確實從結果上來看, 大部分都有收斂到最佳解, 不過也是因為這個問題很簡單.

```
repeat_max <- rep(0, 1000)
for(k in 1:1000){
  Population <- matrix(sample(c(0,1), size = N_size*N_bit, replace = T)
,
                       N_size, N_bit)
  mut_freq <- 0.05
  #2 進位到10 進位
  Po_value <- 8*Population[,1] + 4*Population[,2] + 2*Population[,3] +
1*Population[,4] + 1/2*Population[,5] + 1/4*Population[,6] +1/8*Populat
ion[,7] + 1/16*Population[,8]
  Po_value <- Po_value - 8
  #fitness function
  fit <- function(x){
    2*x^3 - 25*x^2 + 18*x + 3000 - x*sin(x)
  }
  fit_value <- fit(Po_value)

  #第一世代
  idx <- 1:N_size
  max_Gen_value <- rep(0, Gen)
  max_Gen_value[1] <- max(fit_value)
  max_Gen_ind <- matrix(0, Gen, N_bit)
  max_Gen_ind[1,] <- Population[which(fit_value == max(fit_value))[1],]
  max_Gen_Po_value <- rep(0, Gen)
  max_Gen_Po_value[1] <- Po_value[which(fit_value == max(fit_value))[1]
]

  #GA
  for(now_Gen in 2:Gen){
    child <- matrix(0, N_size, N_bit)
    child[1,] <- Population[which(fit_value == max(fit_value))[1],]
    switch_bit <- sample(N_bit, 1)
    child[2,] <- child[1,]
    child[2, switch_bit:N_bit] <- !child[2, switch_bit:N_bit]
    child_size <- 2
    total_wheel <- sum(fit_value)
    select_frequency <- fit_value/total_wheel

    #生子代
    while(child_size < N_size){
      P_idx <- sample(idx, size = 2, replace = F, prob = select_frequen
```

```
cy)
      P1 <- Population[P_idx[1],]
      P2 <- Population[P_idx[2],]
      #兩點交配
      switch_bit <- sample(N_bit, 2)
      switch_bit <- sort(switch_bit)
      P1_new <- P1
      P1_new[switch_bit[1]:switch_bit[2]] <- P2[switch_bit[1]:switch_bi
t[2]]
      P2_new <- P2
      P2_new[switch_bit[1]:switch_bit[2]] <- P1[switch_bit[1]:switch_bi
t[2]]
      if(runif(1, 0, 1) < mut_freq){
        tar_bit <- sample(N_bit,1)
        P1_new[tar_bit] <- !P1_new[tar_bit]
      }
      if(runif(1, 0, 1) < mut_freq){
        tar_bit <- sample(N_bit,1)
        P2_new[tar_bit] <- !P2_new[tar_bit]
      }
      child[child_size + 1,] <- P1_new
      child[child_size + 2,] <- P2_new
      child_size <- child_size + 2
    }
    #記錄該代最佳解
    Population <- child
    Po_value <- 8*Population[,1] + 4*Population[,2] + 2*Population[,3]
+ 1*Population[,4] + 1/2*Population[,5] + 1/4*Population[,6] +1/8*Popul
ation[,7] + 1/16*Population[,8]
    Po_value <- Po_value - 8
    fit_value <- fit(Po_value)
    max_Gen_value[now_Gen] <- max(fit_value)
    max_Gen_Po_value[now_Gen] <- Po_value[which(fit_value == max(fit_va
lue))[1]]
    max_Gen_ind[now_Gen,] <- Population[which(fit_value == max(fit_valu
e))[1],]
  }
  repeat_max[k] <- max(max_Gen_value)
}
plot(repeat_max, type = "l", xlab = "index", ylab = "max value")
```



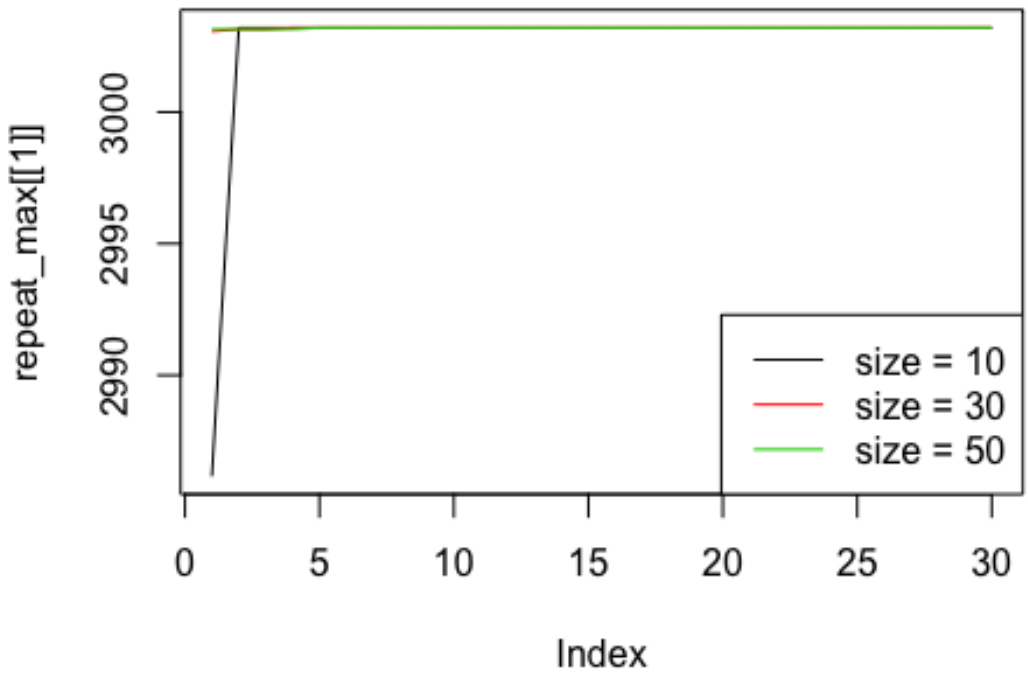
Q3: 請生成 10, 30, 50 個個體比較結果是否有影響？

個體數越多的 GA 能越快達到收斂值, 這次的結果中, 甚至在第一世代就已經很接近收斂值.

```
size_set <- c(10, 30, 50)
repeat_max <- list()
for(k in 1:3){
  N_size <-  size_set[k]  #個體數
  Population <- matrix(sample(c(0,1), size = N_size*N_bit, replace = T)
,
                       N_size, N_bit)
  mut_freq <- 0.05
  #2 進位到10 進位
  Po_value <- 8*Population[,1] + 4*Population[,2] + 2*Population[,3] +
1*Population[,4] + 1/2*Population[,5] + 1/4*Population[,6] +1/8*Populat
ion[,7] + 1/16*Population[,8]
  Po_value <- Po_value - 8
  #fitness function
  fit <- function(x){
    2*x^3 - 25*x^2 + 18*x + 3000 - x*sin(x)
  }
  fit_value <- fit(Po_value)

  #第一世代
  idx <- 1:N_size
  max_Gen_value <- rep(0, Gen)
  max_Gen_value[1] <- max(fit_value)
  max_Gen_ind <- matrix(0, Gen, N_bit)
  max_Gen_ind[1,] <- Population[which(fit_value == max(fit_value))[1],]
  max_Gen_Po_value <- rep(0, Gen)
```

```r
  max_Gen_Po_value[1] <- Po_value[which(fit_value == max(fit_value))[1]
]

  #GA
  for(now_Gen in 2:Gen){
    child <- matrix(0, N_size, N_bit)
    child[1,] <- Population[which(fit_value == max(fit_value))[1],]
    switch_bit <- sample(N_bit, 1)
    child[2,] <- child[1,]
    child[2, switch_bit:N_bit] <- !child[2, switch_bit:N_bit]
    child_size <- 2
    total_wheel <- sum(fit_value)
    select_frequency <- fit_value/total_wheel

    #生子代
    while(child_size < N_size){
      P_idx <- sample(idx, size = 2, replace = F, prob = select_frequen
cy)
      P1 <- Population[P_idx[1],]
      P2 <- Population[P_idx[2],]
      #兩點交配
      switch_bit <- sample(N_bit, 2)
      switch_bit <- sort(switch_bit)
      P1_new <- P1
      P1_new[switch_bit[1]:switch_bit[2]] <- P2[switch_bit[1]:switch_bi
t[2]]
      P2_new <- P2
      P2_new[switch_bit[1]:switch_bit[2]] <- P1[switch_bit[1]:switch_bi
t[2]]
      if(runif(1, 0, 1) < mut_freq){
        tar_bit <- sample(N_bit,1)
        P1_new[tar_bit] <- !P1_new[tar_bit]
      }
      if(runif(1, 0, 1) < mut_freq){
        tar_bit <- sample(N_bit,1)
        P2_new[tar_bit] <- !P2_new[tar_bit]
      }
      child[child_size + 1,] <- P1_new
      child[child_size + 2,] <- P2_new
      child_size <- child_size + 2
    }
    #記錄該代最佳解
    Population <- child
    Po_value <- 8*Population[,1] + 4*Population[,2] + 2*Population[,3]
+ 1*Population[,4] + 1/2*Population[,5] + 1/4*Population[,6] +1/8*Popul
ation[,7] + 1/16*Population[,8]
    Po_value <- Po_value - 8
    fit_value <- fit(Po_value)
    max_Gen_value[now_Gen] <- max(fit_value)
```

```
    max_Gen_Po_value[now_Gen] <- Po_value[which(fit_value == max(fit_va
lue))[1]]
    max_Gen_ind[now_Gen,] <- Population[which(fit_value == max(fit_valu
e))[1],]
  }
  repeat_max[[k]] <- max_Gen_value
}
plot(repeat_max[[1]], type = "l", col = 1, ylab = "max value")
lines(repeat_max[[2]], type = "l", col = 2)
lines(repeat_max[[3]], type = "l", col = 3)
legend("bottomright",lty = 1, col = c(1,2,3), legend = c("size = 10", "
size = 30", "size = 50"))
```



Q4: 請生成代數 10, 30, 50 比較結果是否有影響？

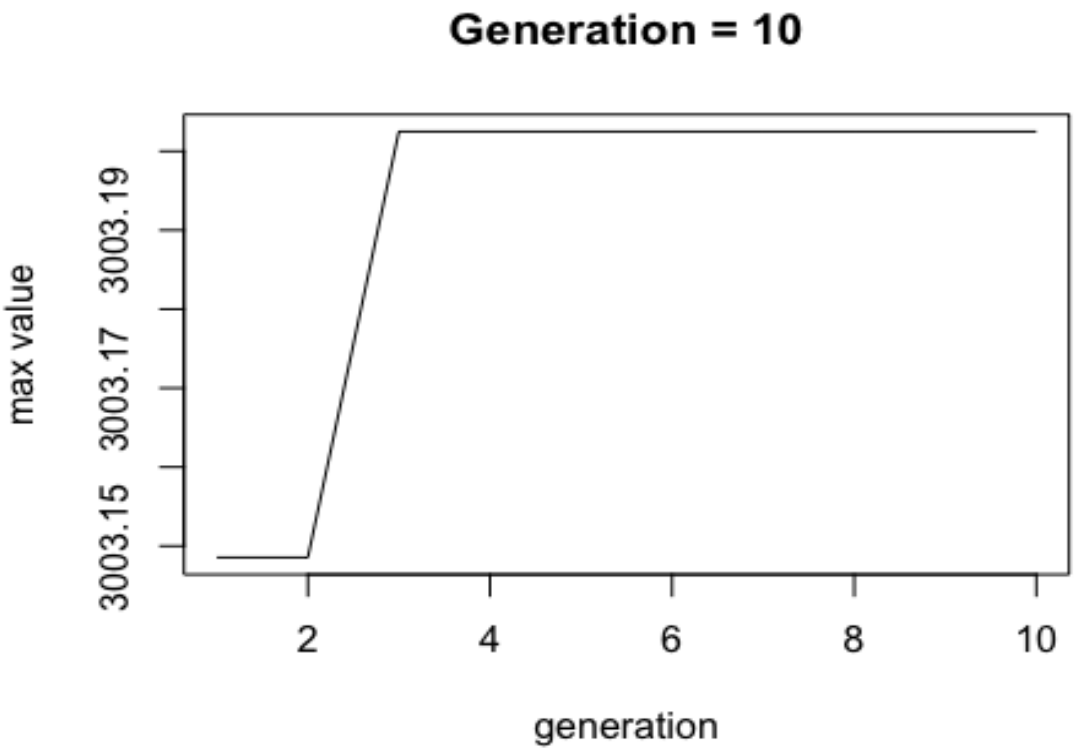
影響不大, 因為這個問題通常在前 5 世代就已經收斂了, 不過當然世代數越大越有可能達到收斂值.

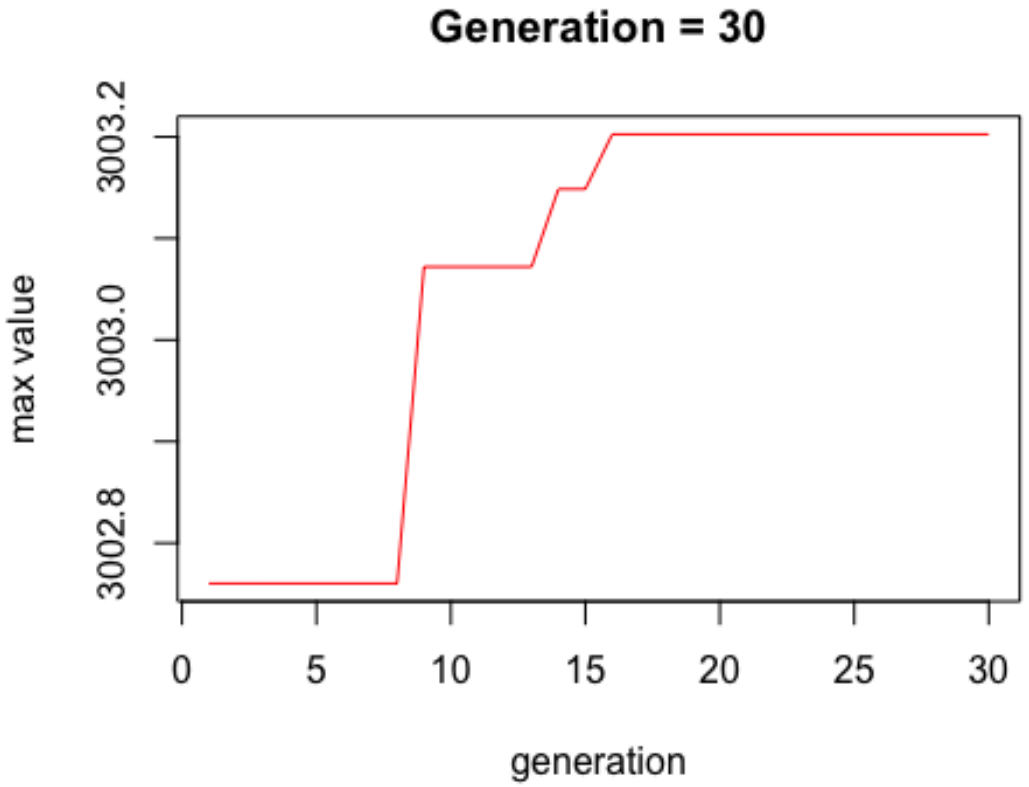
```
Gen_set <- c(10, 30, 50)
N_size <- 30
repeat_max <- list()
for(k in 1:3){
  Gen <-  Gen_set[k]  #個體數
```

```
  Population <- matrix(sample(c(0,1), size = N_size*N_bit, replace = T)
,
                       N_size, N_bit)
  mut_freq <- 0.05
  #2 進位到10 進位
  Po_value <- 8*Population[,1] + 4*Population[,2] + 2*Population[,3] +
1*Population[,4] + 1/2*Population[,5] + 1/4*Population[,6] +1/8*Populat
ion[,7] + 1/16*Population[,8]
  Po_value <- Po_value - 8
  #fitness function
  fit <- function(x){
    2*x^3 - 25*x^2 + 18*x + 3000 - x*sin(x)
  }
  fit_value <- fit(Po_value)

  #第一世代
  idx <- 1:N_size
  max_Gen_value <- rep(0, Gen)
  max_Gen_value[1] <- max(fit_value)
  max_Gen_ind <- matrix(0, Gen, N_bit)
  max_Gen_ind[1,] <- Population[which(fit_value == max(fit_value))[1],]
  max_Gen_Po_value <- rep(0, Gen)
  max_Gen_Po_value[1] <- Po_value[which(fit_value == max(fit_value))[1]
]

  #GA
  for(now_Gen in 2:Gen){
    child <- matrix(0, N_size, N_bit)
    child[1,] <- Population[which(fit_value == max(fit_value))[1],]
    switch_bit <- sample(N_bit, 1)
    child[2,] <- child[1,]
    child[2, switch_bit:N_bit] <- !child[2, switch_bit:N_bit]
    child_size <- 2
    total_wheel <- sum(fit_value)
    select_frequency <- fit_value/total_wheel

    #生子代
    while(child_size < N_size){
      P_idx <- sample(idx, size = 2, replace = F, prob = select_frequen
cy)
      P1 <- Population[P_idx[1],]
      P2 <- Population[P_idx[2],]
      #兩點交配
      switch_bit <- sample(N_bit, 2)
      switch_bit <- sort(switch_bit)
      P1_new <- P1
      P1_new[switch_bit[1]:switch_bit[2]] <- P2[switch_bit[1]:switch_bi
t[2]]
      P2_new <- P2
```

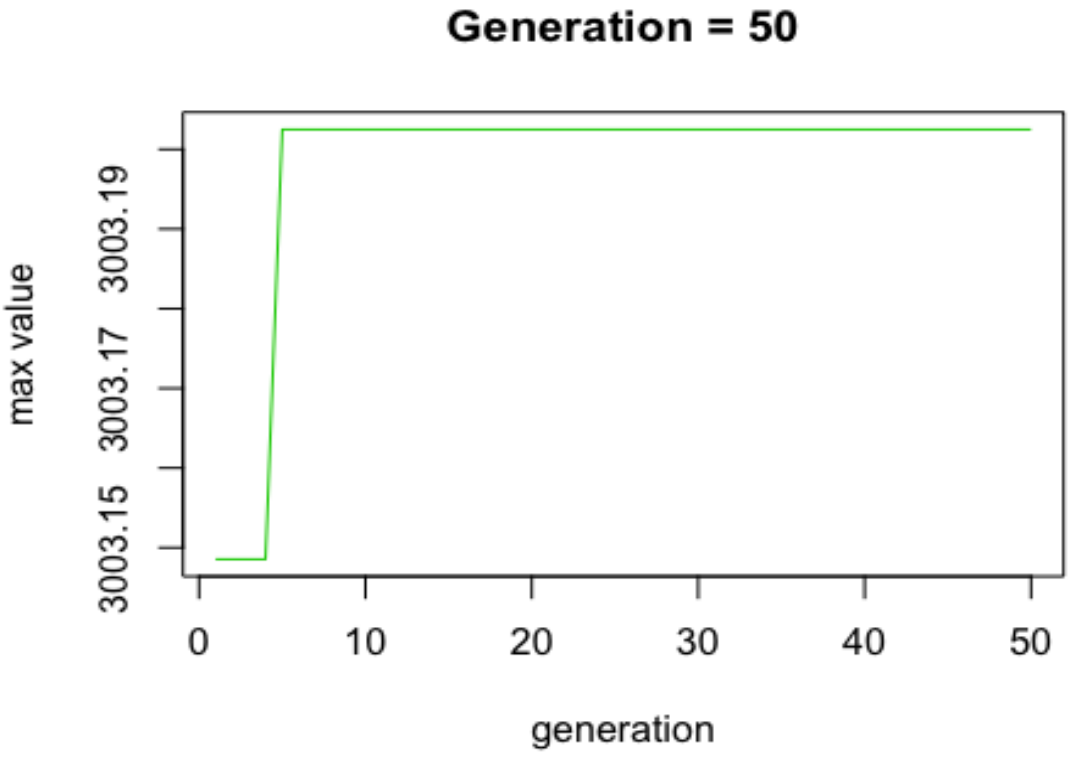
```
      P2_new[switch_bit[1]:switch_bit[2]] <- P1[switch_bit[1]:switch_bi
t[2]]
      if(runif(1, 0, 1) < mut_freq){
        tar_bit <- sample(N_bit,1)
        P1_new[tar_bit] <- !P1_new[tar_bit]
      }
      if(runif(1, 0, 1) < mut_freq){
        tar_bit <- sample(N_bit,1)
        P2_new[tar_bit] <- !P2_new[tar_bit]
      }
      child[child_size + 1,] <- P1_new
      child[child_size + 2,] <- P2_new
      child_size <- child_size + 2
    }
    #記錄該代最佳解
    Population <- child
    Po_value <- 8*Population[,1] + 4*Population[,2] + 2*Population[,3]
+ 1*Population[,4] + 1/2*Population[,5] + 1/4*Population[,6] +1/8*Popul
ation[,7] + 1/16*Population[,8]
    Po_value <- Po_value - 8
    fit_value <- fit(Po_value)
    max_Gen_value[now_Gen] <- max(fit_value)
    max_Gen_Po_value[now_Gen] <- Po_value[which(fit_value == max(fit_va
lue))[1]]
    max_Gen_ind[now_Gen,] <- Population[which(fit_value == max(fit_valu
e))[1],]
  }
  repeat_max[[k]] <- max_Gen_value
```

```
}
plot(repeat_max[[1]], type = "l", col = 1, main = "Generation = 10",
     xlab = "generation", ylab = "max value")

plot(repeat_max[[2]], type = "l", col = 2, main = "Generation = 30",
     xlab = "generation", ylab = "max value")
```






```
plot(repeat_max[[3]], type = "l", col = 3, main = "Generation = 50",
     xlab = "generation", ylab = "max value")
```

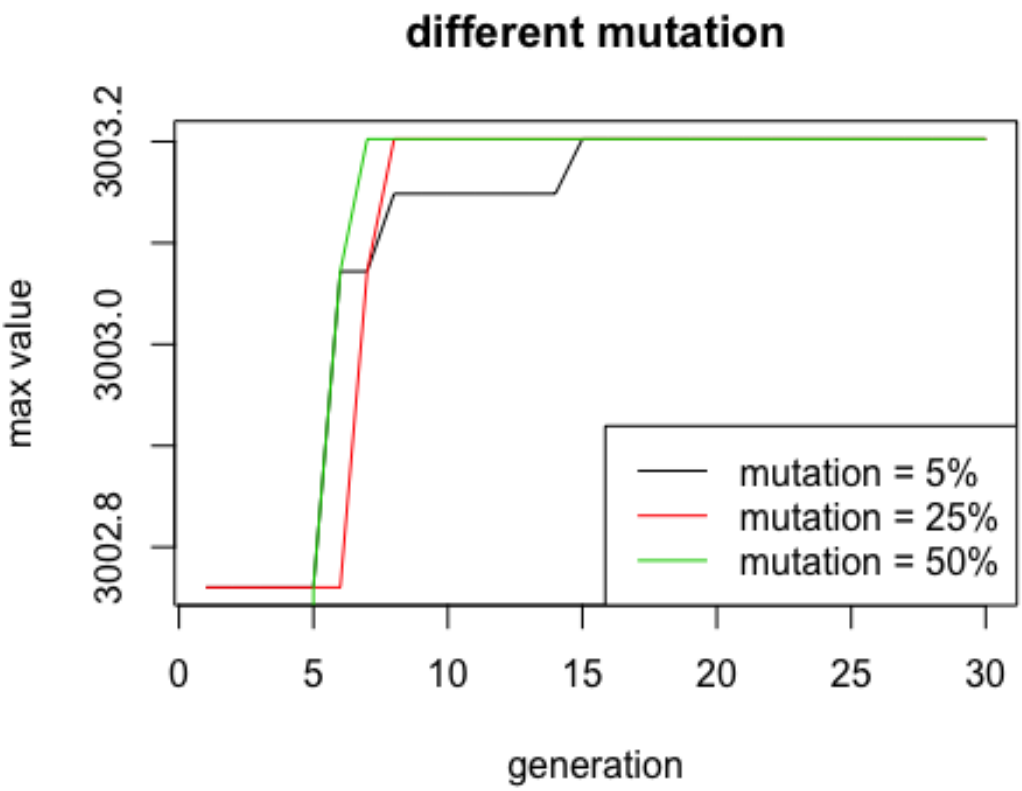


Q5: 請改變突變機率 0.05, 0.25, 0.5 比較結果是否有影響？

突變機率越高的演算法, 在還沒收斂之前, 越有可能有大幅度的成長(接近收斂值), 但也相對不穩定, 我曾經生成過一個突變機率 0.5 的 GA, 做到最後一個世代還是離收斂值很遠.

```
Gen <- 30
mut_freq_set <- c(0.05, 0.25, 0.5)
repeat_max <- list()
for(k in 1:3){
  mut_freq <- 0.05
  Population <- matrix(sample(c(0,1), size = N_size*N_bit, replace = T)
,
                       N_size, N_bit)
  mut_freq <- 0.05
  #2 進位到10 進位
  Po_value <- 8*Population[,1] + 4*Population[,2] + 2*Population[,3] +
1*Population[,4] + 1/2*Population[,5] + 1/4*Population[,6] +1/8*Populat
ion[,7] + 1/16*Population[,8]
  Po_value <- Po_value - 8
  #fitness function
  fit <- function(x){
    2*x^3 - 25*x^2 + 18*x + 3000 - x*sin(x)
  }
  fit_value <- fit(Po_value)

  #第一世代
  idx <- 1:N_size
  max_Gen_value <- rep(0, Gen)
  max_Gen_value[1] <- max(fit_value)
  max_Gen_ind <- matrix(0, Gen, N_bit)
  max_Gen_ind[1,] <- Population[which(fit_value == max(fit_value))[1],]
  max_Gen_Po_value <- rep(0, Gen)
  max_Gen_Po_value[1] <- Po_value[which(fit_value == max(fit_value))[1]
]

  #GA
  for(now_Gen in 2:Gen){
    child <- matrix(0, N_size, N_bit)
    child[1,] <- Population[which(fit_value == max(fit_value))[1],]
    switch_bit <- sample(N_bit, 1)
    child[2,] <- child[1,]
    child[2, switch_bit:N_bit] <- !child[2, switch_bit:N_bit]
    child_size <- 2
    total_wheel <- sum(fit_value)
    select_frequency <- fit_value/total_wheel

    #生子代
    while(child_size < N_size){
```

```
      P_idx <- sample(idx, size = 2, replace = F, prob = select_frequen
cy)
      P1 <- Population[P_idx[1],]
      P2 <- Population[P_idx[2],]
      #兩點交配
      switch_bit <- sample(N_bit, 2)
      switch_bit <- sort(switch_bit)
      P1_new <- P1
      P1_new[switch_bit[1]:switch_bit[2]] <- P2[switch_bit[1]:switch_bi
t[2]]
      P2_new <- P2
      P2_new[switch_bit[1]:switch_bit[2]] <- P1[switch_bit[1]:switch_bi
t[2]]
      if(runif(1, 0, 1) < mut_freq){
        tar_bit <- sample(N_bit,1)
        P1_new[tar_bit] <- !P1_new[tar_bit]
      }
      if(runif(1, 0, 1) < mut_freq){
        tar_bit <- sample(N_bit,1)
        P2_new[tar_bit] <- !P2_new[tar_bit]
      }
      child[child_size + 1,] <- P1_new
      child[child_size + 2,] <- P2_new
      child_size <- child_size + 2
    }
    #記錄該代最佳解
    Population <- child
    Po_value <- 8*Population[,1] + 4*Population[,2] + 2*Population[,3]
+ 1*Population[,4] + 1/2*Population[,5] + 1/4*Population[,6] +1/8*Popul
ation[,7] + 1/16*Population[,8]
    Po_value <- Po_value - 8
    fit_value <- fit(Po_value)
    max_Gen_value[now_Gen] <- max(fit_value)
    max_Gen_Po_value[now_Gen] <- Po_value[which(fit_value == max(fit_va
lue))[1]]
    max_Gen_ind[now_Gen,] <- Population[which(fit_value == max(fit_valu
e))[1],]
  }
  repeat_max[[k]] <- max_Gen_value
}
plot(repeat_max[[1]], type = "l", col = 1,
     xlab = "generation", ylab = "max value", main = "different mutatio
n")
lines(repeat_max[[2]], type = "l", col = 2)
lines(repeat_max[[3]], type = "l", col = 3)
legend("bottomright",lty = 1, col = c(1,2,3), legend = c("mutation = 5%
", "mutation = 25%", "mutation = 50%"))
```


different mutation
max value
3002.8
3003.0
3003.2
0
5
10
15
20
25
30
generation
mutation = 5%
mutation = 25%
mutation = 50%