1. 請撰寫GA計算下列函數的最大值

> max(max\_Gen\_value)

[1] 3003.202

Code:

N\_size <- 30

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

Po\_value <- 8\*Population[,1] + 4\*Population[,2] + 2\*Population[,3] + 1\*

Population[,4] + 1/2\*Population[,5] + 1/4\*Population[,6] +1/8\*Population[,7] + 1/16\*Population[,8]

Po\_value <- Po\_value - 8

fit <- function(x){

2\*x^3 - 25\*x^2 + 18\*x + 3000 - x\*sin(x)

}

fit\_value <- fit(Po\_value)

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]]

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

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],]

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

}

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\*Population[,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\_value))[1]]

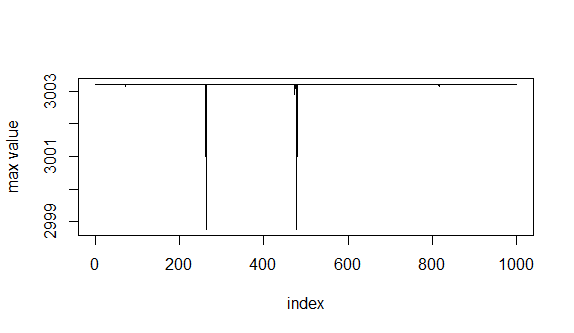
max\_Gen\_ind[now\_Gen,] <- Population[which(fit\_value == max(fit\_value))[1],]

}

max(max\_Gen\_value)

1. 請重複跑你的程式碼1000次，比較每次結果是否相同

幾乎所有結果都有收斂到最佳解，即3003.202。



Code: ans1000 <- c()

for(i in 1:1000){

N\_size <- 30

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

Po\_value <- 8\*Population[,1] + 4\*Population[,2] + 2\*Population[,3] + 1\*

Population[,4] + 1/2\*Population[,5] + 1/4\*Population[,6] +1/8\*Population[,7] + 1/16\*Population[,8]

Po\_value <- Po\_value - 8

fit <- function(x){

2\*x^3 - 25\*x^2 + 18\*x + 3000 - x\*sin(x)

}

fit\_value <- fit(Po\_value)

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]]

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

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],]

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

}

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\*Population[,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\_value))[1]]

max\_Gen\_ind[now\_Gen,] <- Population[which(fit\_value == max(fit\_value))[1],]

}

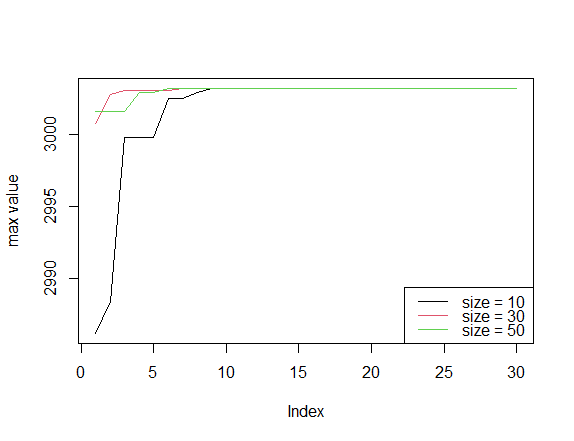
ans1000[i] <- max(max\_Gen\_value)

}

plot(ans1000, type="l", xlab = "index", ylab = "max value")

1. 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。



Code:

size <- c(10, 30, 50)

repeat\_max <- list()

for(i in 1:3){

N\_size <- size[i]

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

Po\_value <- 8\*Population[,1] + 4\*Population[,2] + 2\*Population[,3] + 1\*

Population[,4] + 1/2\*Population[,5] + 1/4\*Population[,6] +1/8\*Population[,7] + 1/16\*Population[,8]

Po\_value <- Po\_value - 8

fit <- function(x){

2\*x^3 - 25\*x^2 + 18\*x + 3000 - x\*sin(x)

}

fit\_value <- fit(Po\_value)

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]]

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

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],]

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

}

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\*Population[,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\_value))[1]]

max\_Gen\_ind[now\_Gen,] <- Population[which(fit\_value == max(fit\_value))[1],]

}

repeat\_max[[i]] <- max\_Gen\_value

}

plot(repeat\_max[[1]], type = "l", col = 1, ylab = "max value")

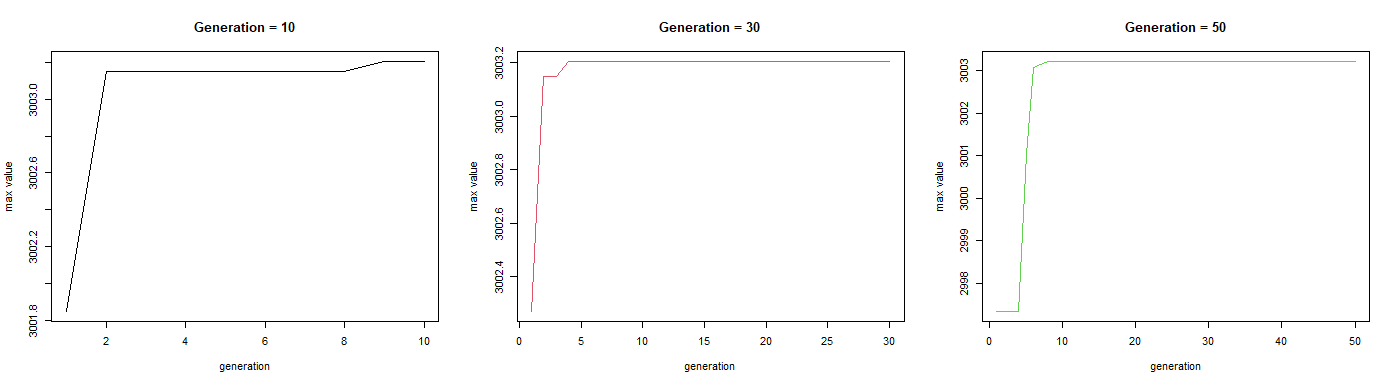
lines(repeat\_max[[2]], type = "l", col = 2)

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"))

1. 請生成代數10, 30, 50比較結果是否有影響

影響不大，因為通常在前 5 世代就已經收斂，不過世代數越大依然越有可能達到收斂值。



Code:

##4

Gen\_set <- c(10, 30, 50)

N\_size <- 30

N\_bit <- 8

repeat\_max <- list()

for(k in 1:3){

Gen <- Gen\_set[k]

Population <- matrix(sample(c(0,1), size = N\_size\*N\_bit, replace = T),N\_size, N\_bit)

mut\_freq <- 0.05

Po\_value <- 8\*Population[,1] + 4\*Population[,2] + 2\*Population[,3] +1\*Population[,4] + 1/2\*Population[,5] + 1/4\*Population[,6] +1/8\*Population[,7] + 1/16\*Population[,8]

Po\_value <- Po\_value - 8

fit <- function(x){

2\*x^3 - 25\*x^2 + 18\*x + 3000 - x\*sin(x)

}

fit\_value <- fit(Po\_value)

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]

]

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

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],]

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

}

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\*Population[,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\_value))[1]]

max\_Gen\_ind[now\_Gen,] <- Population[which(fit\_value == max(fit\_value))[1],]

}

repeat\_max[[k]] <- max\_Gen\_value

}

par(mfcol=c(1,3))

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",

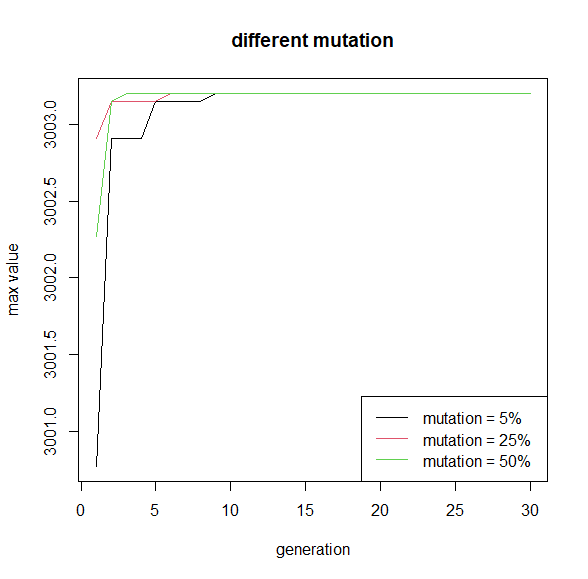
xlab = "generation", ylab = "max value")

plot(repeat\_max[[3]], type = "l", col = 3, main = "Generation = 50",

xlab = "generation", ylab = "max value")

1. 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，在收斂前，越有可能有大幅度的成長(接近收斂值)，但也相對不穩定。



Code:

N\_size <- 30

N\_bit <- 8

Gen <- 30

mut\_freq\_set <- c(0.05, 0.25, 0.5)

repeat\_max <- list()

for(k in 1:3){

mut\_freq <- mut\_freq\_set[i]

Population <- matrix(sample(c(0,1), size = N\_size\*N\_bit, replace = T), N\_size, N\_bit)

Po\_value <- 8\*Population[,1] + 4\*Population[,2] + 2\*Population[,3] + 1\*Population[,4] + 1/2\*Population[,5] + 1/4\*Population[,6] +1/8\*Population[,7] + 1/16\*Population[,8]

Po\_value <- Po\_value - 8

fit <- function(x){

2\*x^3 - 25\*x^2 + 18\*x + 3000 - x\*sin(x)

}

fit\_value <- fit(Po\_value)

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]

]

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

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],]

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

}

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\*Population[,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\_value))[1]]

max\_Gen\_ind[now\_Gen,] <- Population[which(fit\_value == max(fit\_value))[1],]

}

repeat\_max[[k]] <- max\_Gen\_value

}

plot(repeat\_max[[1]], type = "l", col = 1, xlab = "generation", ylab = "max value", main = "different mutation")

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%"))