Computational Statistics

Lab 6 report

Jonathan Dorairaj, YiHung Chen

2022-12-15

Question 1 : Genetic Algorithm

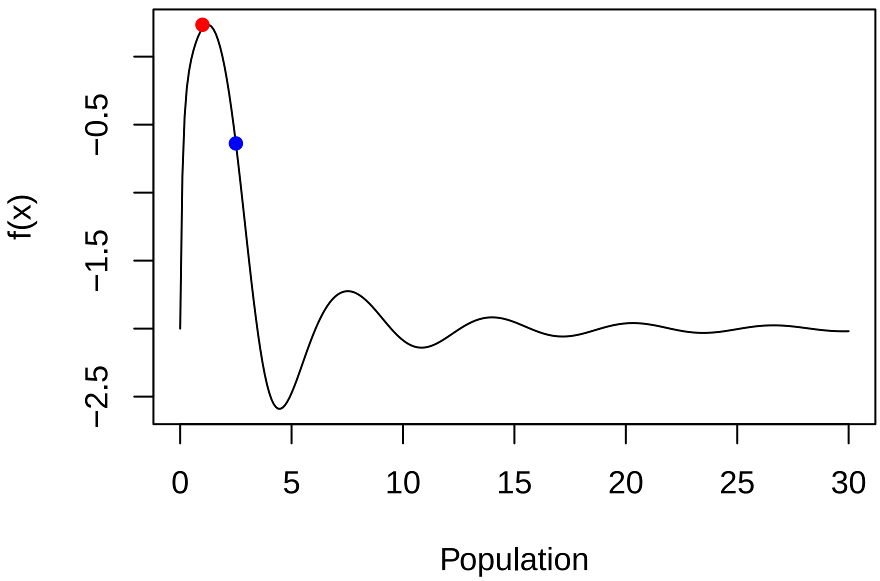
```
## 1.1

fx <- function(x){
  term1 <- ((x^2)/exp(x))
  term2 <- (2*exp((-9*sin(x))/(x^2 + x + 1)))
  res <- term1 - term2
  return(res)
}

crossover <- function(x,y){
  kid <- (x + y)/2
  return(kid)
}

mutate <- function(x){
  res <- (x^2)%%30
  return(res)
}

somefun <- function(maxiter,mutprob){
## 1.4a
x <- seq(0,30,0.1)
y <- fx(x)
plot.new()
plot(x = x,y = y,type = 'l',ylab = "f(x)", xlab = "Population",
     main = paste("maxiterations: ",maxiter,"","mutation prob: ",mutprob))
points(max(y),col = 'red',pch = 16)

## 1.4b
X  <- seq(0,30,5)
X
# y <- fx(X)

# plot(x = x,y = y)
# points(max(y),col = 'red')

values <- fx(X)
```

```r
max_val <- max(values)
for (it in 1:maxiter) {
  parents <- sample(X,2,replace = F)
  # cat("parents:",parents,"\n")


  small_index <- order(values)[1]
  # cat("small_index:", small_index,"\n")

  victim <- X[small_index]
  # print(victim)

  kid <- crossover(parents[1],parents[2])
  # print(kid)

  s <- sample(1:2,1,prob = c(mutprob,1-mutprob))
  # print(s)

  kid <- ifelse(s==1,mutate(kid),kid)
  # print(kid)

  X[small_index] <- kid
  values <- fx(X)
  max_val[it+1] <- max(values)
  # plot(x = X,y = values,col = 'blue',pch  = 16)
  # Sys.sleep(0.5)
}

# points(x = X,y = values,col = 'blue',pch  = 16)
points(x = X[which.max(values)],y= max(values),col = 'blue',pch  = 16)
 Sys.sleep(0.5)
 return(max(max_val))
}
```

Q: Run your code with different combinations of maxiter= 10, 100 and mutprob= 0.1, 0.5, 0.9. Observe the initial population and final population. Conclusions?

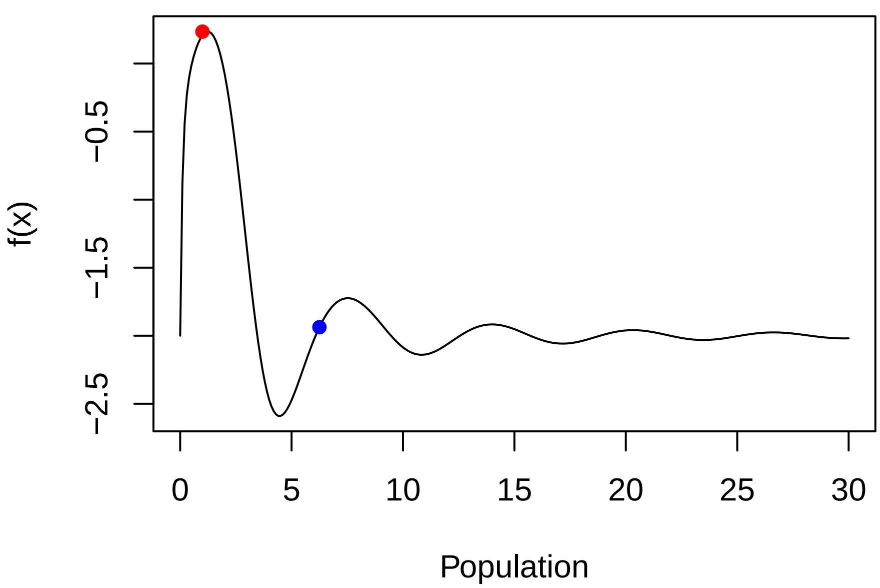
```r
maxv <- c()
# par(mfrow = c(3,2))
maxv[1] <- somefun(10,0.1)
```

maxiterations: 10 mutation prob: 0.1

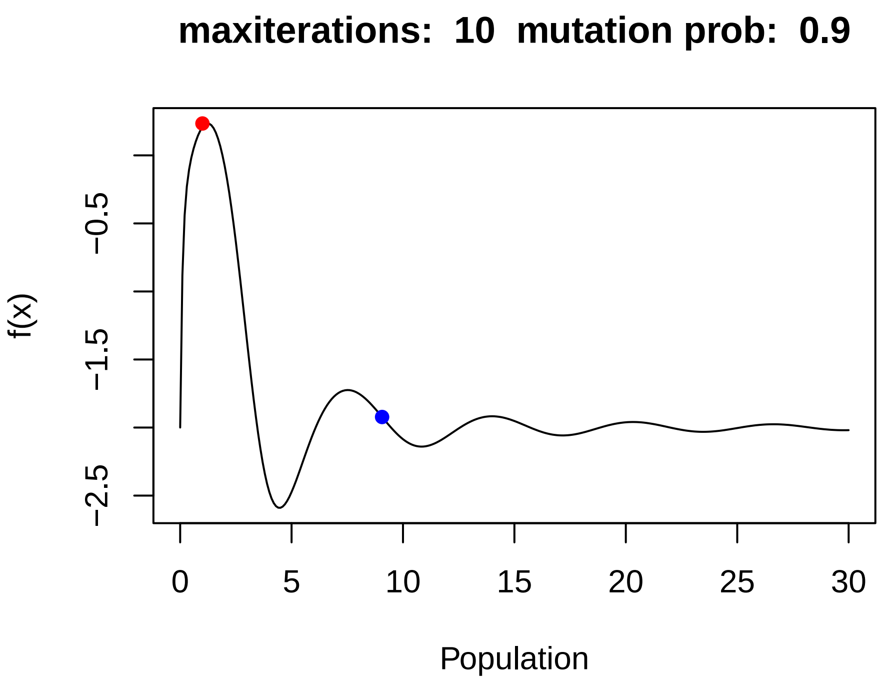



```r
maxv[2] <- somefun(10,0.5)
```

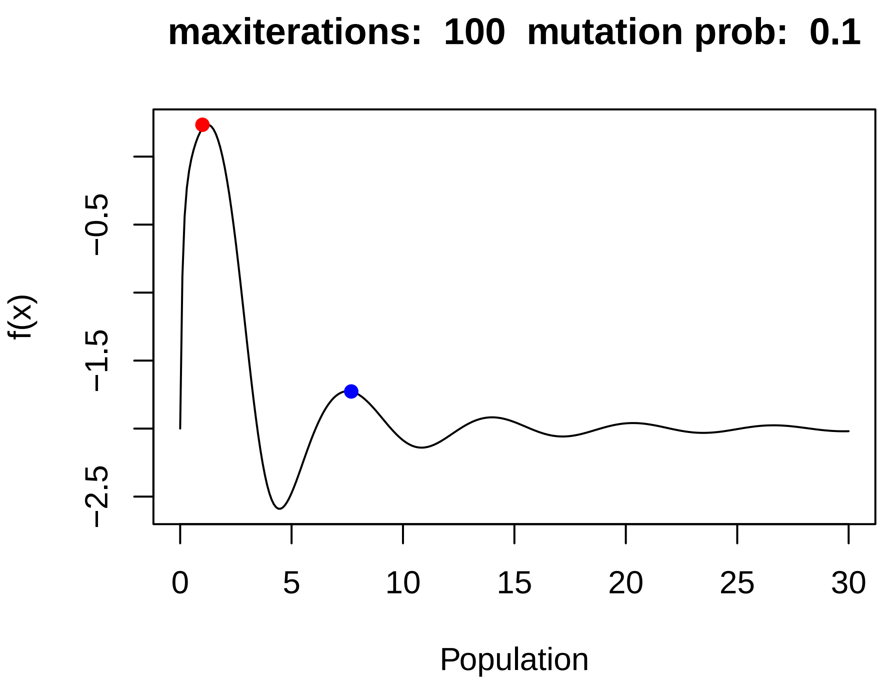
maxiterations: 10 mutation prob: 0.5



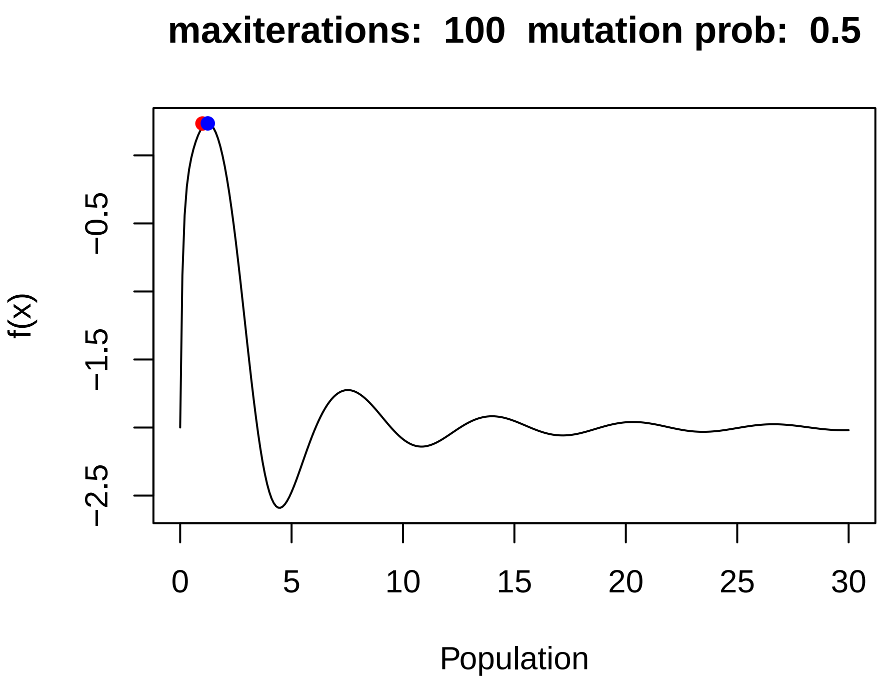
```r
maxv[3] <- somefun(10,0.9)
```



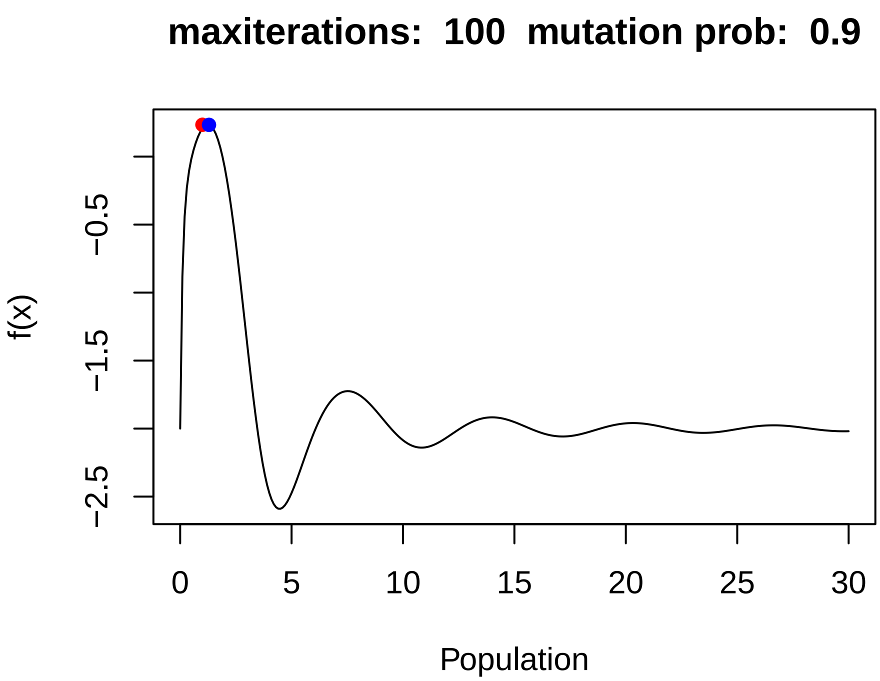

```r
maxv[4] <- somefun(100,0.1)
```



```
maxv[5] <- somefun(100,0.5)
```




```
maxv[6] <- somefun(100,0.9)
```



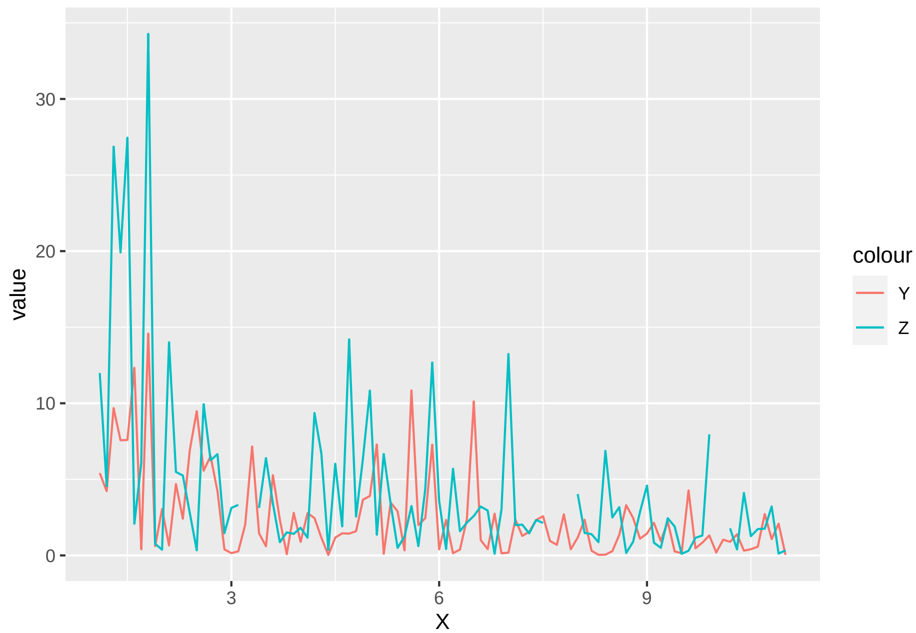
From the plots of the figures above, we observe that as the number of iterations `maxiter` increass, the genetic algorithm is more capble to estimate the maximum. Additionally, as the probability of mutation `mutprob` increases, the algorithm is able to estimate the maximum value of the function better. This is due to the fact higher mutations probability can ensure we avoid getting stuck in a local mazimum.

Question 2: EM algorithm

The data file physical.csv describes a behavior of two related physical processes Y = Y (X) and Z = Z(X).

Q2.1

Make a time series plot describing dependence of Z and Y versus X



By observing the plot above, both Y and Z value tend to decrease as X increase, with Z tend to have higher value then Y. However,there is no clear relation between Y and Z as they have different spikes for X values. Which we can assume Y and Z are independent in the later calculation.

Q2.2

Note that there are some missing values of Z in the data which implies problems in estimating models by maximum likelihood. Use the following model

$$Y_i \sim exp(X_i/\lambda),\ Z_i \sim exp(X_i/2\lambda)$$

$$where\ \lambda\ is\ some\ unknown\ parameter.$$

The goal is to derive an EM algorithm that estimates λ.

The probability density function of Y and Z are

$$f(Y_i) = (\frac{X_i}{\lambda})e^{-(\frac{X_i}{\lambda})Y_i}$$

$$f(Z_i) = (\frac{X_i}{\lambda})e^{-(\frac{X_i}{2\lambda})Z_i}$$

Likelihood

Since Y and Z are independent, we can joint them together to calculate the likelihood. (n is the total number of data)

$$L = \prod_{i=1}^{n}(\frac{X_i}{\lambda}e^{-(\frac{X_i}{\lambda})Y_i}) * \prod_{i=1}^{n}(\frac{X_i}{2\lambda}e^{-(\frac{X_i}{2\lambda})Z_i})$$
$$= \frac{1}{\lambda^n}(X_i...X_n)e^{\frac{-1}{\lambda}\sum_1^n X_iY_i} * \frac{1}{(2\lambda)^n}(X_i...X_n)e^{\frac{-1}{2\lambda}\sum_1^n X_iZ_i}$$

log-likelihood

$$lnL = -nln(\lambda) + \sum_{i=i}^{n} ln(X_i) - \frac{1}{\lambda}\sum_{i=1}^{n} X_iY_i - nln(2\lambda) + \sum_{i=1}^{n} ln(X_i) - \frac{1}{2\lambda}\sum_{i=1}^{n} X_iZ_i$$

E-step
When computing Q function, since X, Y, Z_{obs}(non-missing Z data),λ are given, so we can move them outside the exception as they can be seen as constant, leaving Z_miss inside exception. (m is the number of missing data)

$$Q(\lambda, \lambda^k) = E[lnL|\lambda^k, X, Y, Z_{obs}]$$
$$= -nln(\lambda) + \sum_{i=1}^{n} ln(X_i) - \frac{1}{\lambda}\sum_{i=1}^{n} X_iY_i - nln(2\lambda) + \sum_{i=1}^{n} ln(X_i) - E[\frac{1}{2\lambda}\sum_{i=1}^{n} X_iZ_i|Z_{obs}]$$
$$= -nln(\lambda) + \sum_{i=1}^{n} ln(X_i) - \frac{1}{\lambda}\sum_{i=1}^{n} X_iY_i - nln(2\lambda) + \sum_{i=1}^{n} ln(X_i) - \frac{1}{2\lambda}[\sum_{i=1}^{n-m} X_iZ_{obs} + \sum_{i=n-m+1}^{n} Xi\frac{2\lambda^k}{X_i}]$$
$$where\ E[Z_{miss}] = \frac{2\lambda^k}{X_i}$$
$$= -nln(\lambda) + \sum_{i=1}^{n} ln(X_i) - \frac{1}{\lambda}\sum_{i=1}^{n} X_iY_i - nln(2\lambda) + \sum_{i=1}^{n} ln(X_i) - \frac{1}{2\lambda}[\sum_{i=1}^{n-m} X_iZ_{obs} + m * 2\lambda_k]$$

M-step Do the derivative of Q function and set it to 0

$$\frac{dQ(\lambda, \lambda^k)}{d\lambda} = \frac{-n}{\lambda} + \frac{1}{\lambda^2}\sum_{i=1}^{n} X_iY_i - \frac{n}{\lambda} + \frac{1}{2\lambda^2}\sum_{i=1}^{n-m} X_iZ_{obs} + \frac{1}{\lambda^2}m\lambda^k = 0$$

$$\Rightarrow -n\lambda + \sum_{i=1}^{n} X_iY_i - n\lambda + \frac{1}{2}\sum_{i=1}^{n-m} X_iZ_{obs} + m\lambda^k = 0$$

$$\Rightarrow 2n\lambda = \sum_{i=1}^{n} X_iY_i + \frac{1}{2}\sum_{i=1}^{n-m} X_iZ_{obs} + m\lambda^k$$

$$\Rightarrow \ \lambda = \frac{1}{2n}\sum_{i=1}^{n-m} X_iY_i + \frac{1}{4n}\sum_{i=1}^{n-m} X_iZ_{obs} + \frac{m}{2n}\lambda^k$$

```
X <- physical1$X
Y <- physical1$Y
Z <- physical1$Z
X_obs <- X[!is.na(Z)]
Z_obs <- Z[!is.na(Z)]
Z_miss <- Z[is.na(Z)]
```

```
EM_function<-function(X,Y,Z,X_obs,Z_obs,kmax,eps){
  n <- length(X)
  m <- length(Z_miss)

  lambda_prev <- 0
  lambda_current <- 100
  k <- 0
  while ((abs(lambda_prev-lambda_current)>eps) && (k<(kmax+1))){
    lambda_prev<-lambda_current
    #Usimg Q function
    lambda_current <- (1/(2*n))*(sum(X*Y))+(1/(4*n))*sum(X_obs*Z_obs)+(m/(2*n))*lambda_prev

    k<-k+1

  }

  return(c(lambda_current,k))
}

EM_result <- EM_function(X,Y,Z,X_obs,Z_obs,50,0.001)
# kmax is assign arbitary since we are not given the limit of iteration


optimal_lambda <- EM_result[1]
steps <- EM_result[2]
```

Q2.3

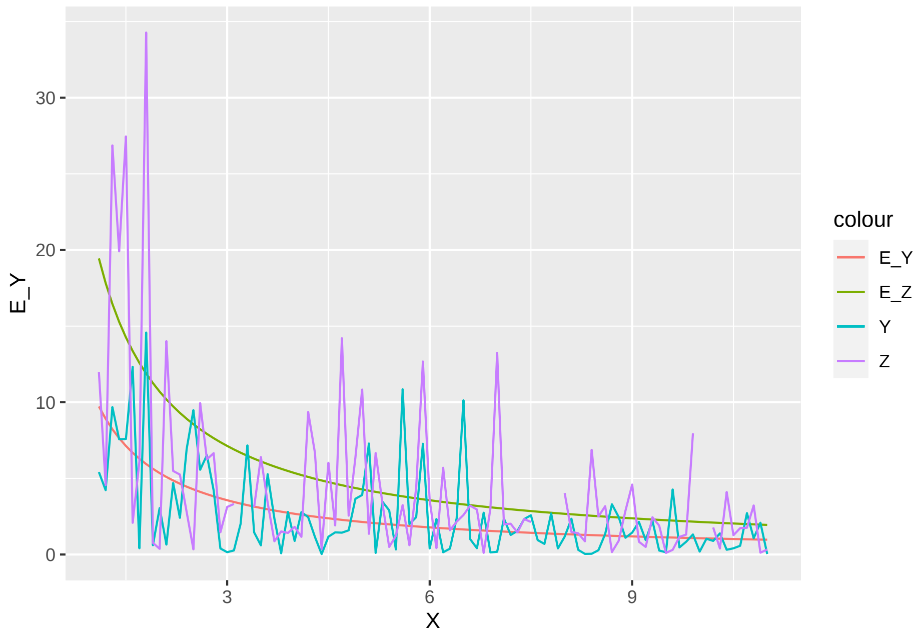
After implement the EM algorithm, and using $\lambda_0 = 100$ and convergence criterion "stop if the change in λ is less than 0.001". We obtain the optimal λ =10.6956555 after 5 iterations.

Q2.4

Plot E[Y] and E[Z] versus X in the same plot as Y and Z versus X. Comment whether the computed λ seems to be reasonable.

Since Y and Z are exponential distribution, so E[Y] and E[Z] have the form

$$E[Y] = \frac{\lambda_{optimal}}{X_i} \ , \ E[Z] = \frac{2\lambda_{optimal}}{X_i}$$



As the graph shown, both E[Y] and E[Z] has the decreasing trend as the X increase. Also, E[Z] has larger value than E[Y] which is the tendency we observed in the original Y,Z data. Thus, we can conclude that both expectation catch the trend of the original data reasonably.

Appendix

Assignment 1

```
## 1.1

fx <- function(x){
  term1 <- ((x^2)/exp(x))
  term2 <- (2*exp((-9*sin(x))/(x^2 + x + 1)))
  res <- term1 - term2
  return(res)
}

crossover <- function(x,y){
  kid <- (x + y)/2
  return(kid)
}

mutate <- function(x){
  res <- (x^2)%%30
  return(res)
}

somefun <- function(maxiter,mutprob){
## 1.4a
x <- seq(0,30,0.1)
y <- fx(x)
plot.new()
plot(x = x,y = y,type = 'l',ylab = "f(x)", xlab = "Population",
     main = paste("maxiterations: ",maxiter,"","mutation prob: ",mutprob))
points(max(y),col = 'red',pch = 16)

## 1.4b
X  <- seq(0,30,5)
X
# y <- fx(X)

# plot(x = x,y = y)
# points(max(y),col = 'red')

values <- fx(X)
max_val <- max(values)
for (it in 1:maxiter) {
  parents <- sample(X,2,replace = F)
  # cat("parents:",parents,"\n")


  small_index <- order(values)[1]
  # cat("small_index:", small_index,"\n")

  victim <- X[small_index]
  # print(victim)
```

```
  kid <- crossover(parents[1],parents[2])
  # print(kid)

  s <- sample(1:2,1,prob = c(mutprob,1-mutprob))
  # print(s)

  kid <- ifelse(s==1,mutate(kid),kid)
  # print(kid)

  X[small_index] <- kid
  values <- fx(X)
  max_val[it+1] <- max(values)
  # plot(x = X,y = values,col = 'blue',pch  = 16)
  # Sys.sleep(0.5)
}

# points(x = X,y = values,col = 'blue',pch  = 16)
points(x = X[which.max(values)],y= max(values),col = 'blue',pch  = 16)
 Sys.sleep(0.5)
 return(max(max_val))
}

maxv <- c()
# par(mfrow = c(3,2))
maxv[1] <- somefun(10,0.1)
maxv[2] <- somefun(10,0.5)
maxv[3] <- somefun(10,0.9)
maxv[4] <- somefun(100,0.1)
maxv[5] <- somefun(100,0.5)
maxv[6] <- somefun(100,0.9)
```

Assignment 2

```r
#Question 2
library(ggplot2)
physical1 <- read.csv("physical1.csv")
ggplot(data=physical1)+
  geom_line(aes(x=X,y=Y,color="Y"))+
  geom_line(aes(x=X,y=Z,color="Z"))+ylab("value")


X <- physical1$X
Y <- physical1$Y
Z <- physical1$Z
X_obs <- X[!is.na(Z)]
Z_obs <- Z[!is.na(Z)]
Z_miss <- Z[is.na(Z)]


EM_function<-function(X,Y,Z,X_obs,Z_obs,kmax,eps){
  n <- length(X)
  m <- length(Z_miss)

  lambda_prev <- 0
  lambda_current <- 100
  k <- 0
  while ((abs(lambda_prev-lambda_current)>eps) && (k<(kmax+1))){
    lambda_prev<-lambda_current
    #Usimg Q function
    lambda_current <- (1/(2*n))*(sum(X*Y))+(1/(4*n))*sum(X_obs*Z_obs)+(m/(2*n))*lambda_prev

    k<-k+1

  }

  return(c(lambda_current,k))
}

EM_result <- EM_function(X,Y,Z,X_obs,Z_obs,50,0.001)
# kmax is assign arbitary since we are not given the limit of iteration


optimal_lambda <- EM_result[1]
steps <- EM_result[2]


E_Y <- optimal_lambda/X
E_Z <- 2*optimal_lambda/X
new_df <- data.frame(physical1,E_Y,E_Z)


ggplot(data=new_df)+
  geom_line(aes(x=X,y=E_Y,color="E_Y"))+
  geom_line(aes(x=X,y=E_Z,color="E_Z"))+
```

```
  geom_line(aes(x=X,y=Y,color="Y"))+
  geom_line(aes(x=X,y=Z,color="Z"))
```