Lab 6

Group 6: Shwetha Vandagadde, Suhani Ariga and Hoda Fakharzadeh

December 14, 2020

Contents

Question 1: Genetic algorithm

1.1 *Define the function .*

$$f(x) := \frac{x^2}{e^x} - 2\exp(\frac{-9\sin x}{x^2+x+1})$$

```
#1.1
GeneticFun <- function(x){

  return(x^2/exp(x)-2*exp(-(9*sin(x))/(x^2+x+1)))
}
```

1.2 *Define the function crossover(): for two scalars x and y it returns their kid as* $(x+y)/2$ *.*

```
#1.2
crossover <- function(x,y){
```

```
  kid <- (x+y)/2
  return(kid)
}
```

1.3 *Define the function mutate().for a scalar x returns the result of the integer division x2 mod 30.*

```
#1.3
mutate <- function(x){

  return((x^2 ) %% 30)
}
```

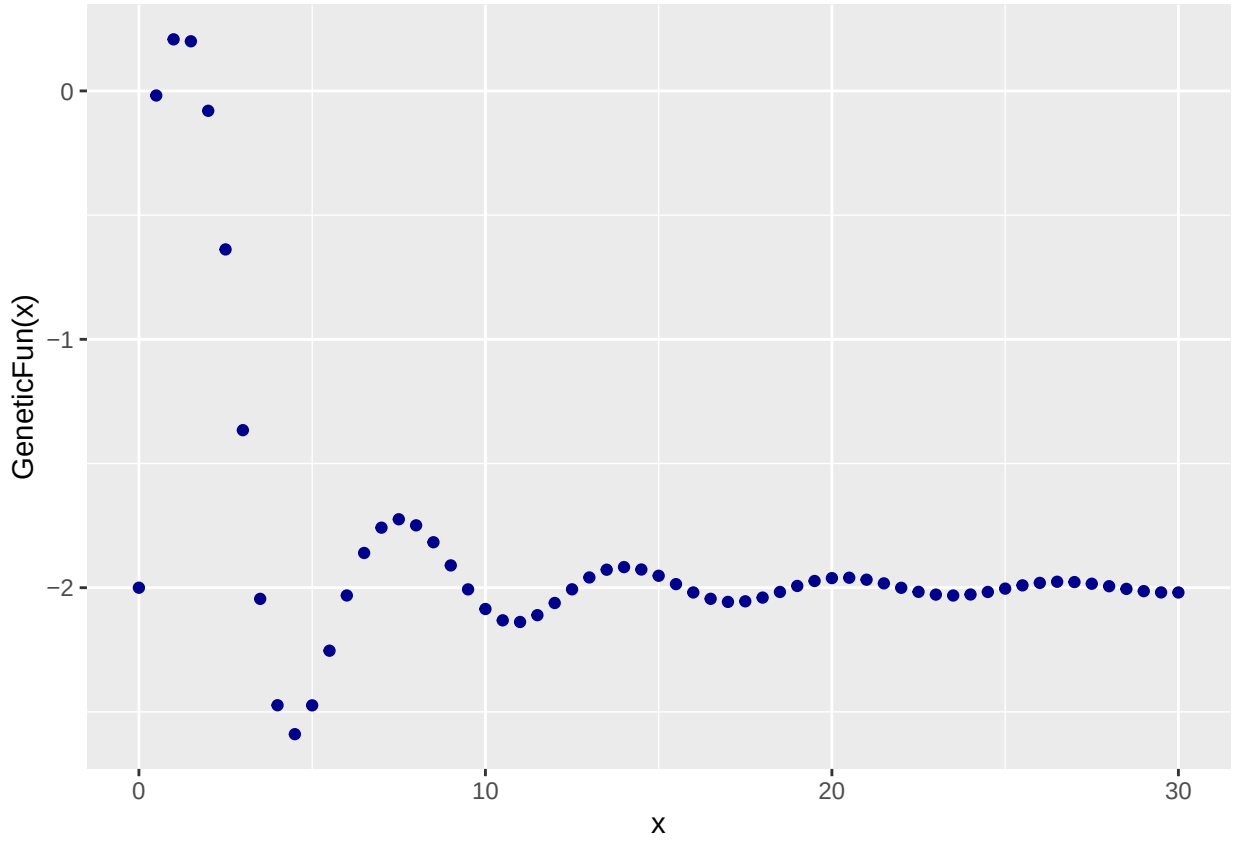
1.4 *Write a function that depends on the parameters* `maxiter`*and* `mutprob`

(a) Plots function f in the range from 0 to 30. Do you see any maximum value?

```
library(ggplot2)
  #4.a plot
  x <- seq(0,30,0.5)
  globalmax <- max(GeneticFun(x))
  cat("global max is",globalmax, "at x = ",
      x[order(GeneticFun(x),decreasing = TRUE)[1] ])
```

```
## global max is 0.2076688 at x =  1
```

```
  #plot(GeneticFun(x))
  gg <- ggplot(data.frame(GeneticFun(x))) +
    geom_point(aes(x,GeneticFun(x)),color='darkblue')
  gg
```



From the graph, It is clear the function is not Convex. This function contains several local maximum and a global maximum. Our goal, is to detect this global maximum with a genetic algorithm.

1.4.

b) Defines an initial population for the genetic algorithm as $X = (0, 5, 10, 15, ..., 30)$

c) Computes vector `Values` *that contains the function values for each population point d) Performs* `maxiter` *iterations where at each iteration*

```
Q4Fun<-function(maxiter,mutprob){


  # 4.b define initial pop
  X <- seq(0,30,5)
  # 4.c Computes vector Values
  Values <- GeneticFun(X)
  init_population <- data.frame("X"=X,"Values" = Values)



  #4.d

  maximal = c()

  for (i in 1:maxiter) {
```

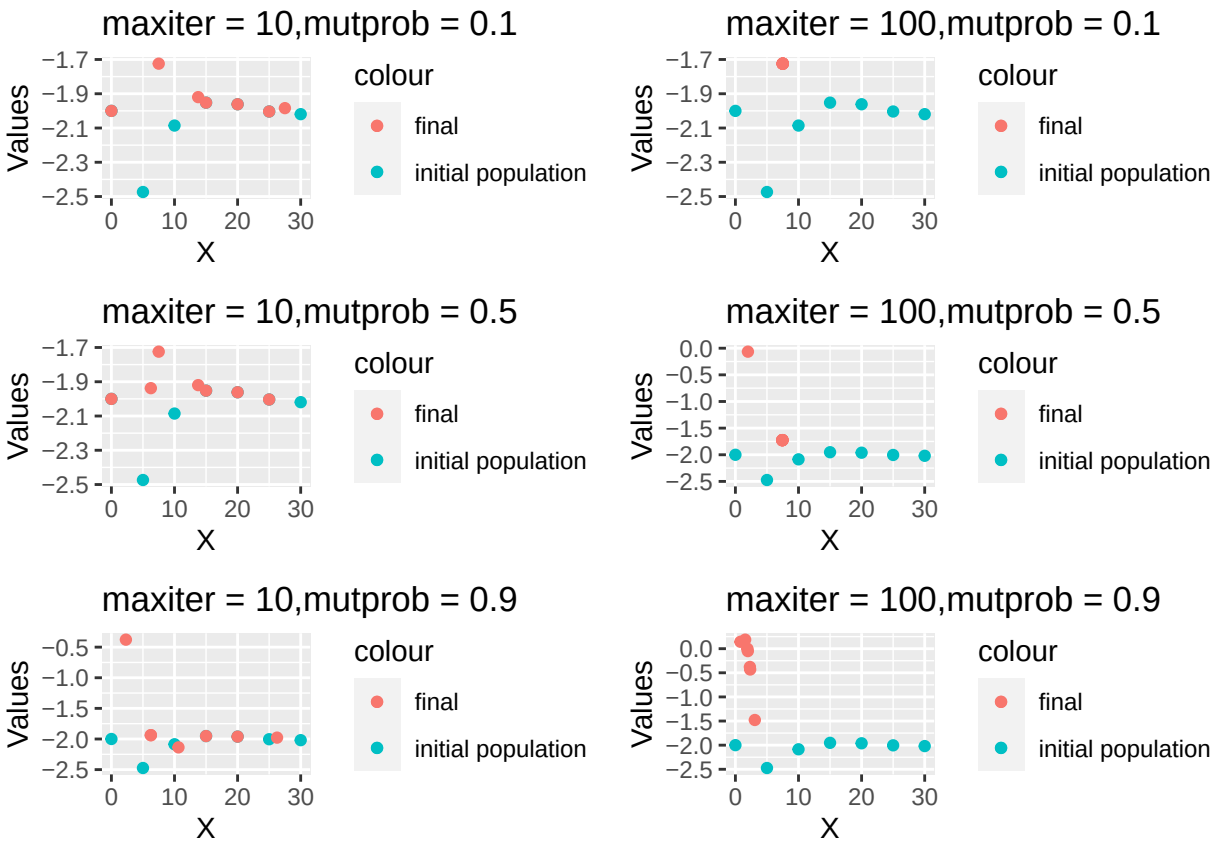
```
    #i
    parents <- sample(x = length(X),size = 2,replace = FALSE)
    ## ii
    victim_idx <-tail(order(Values,decreasing = TRUE),1)
    victim <- X[tail(order(Values,decreasing = TRUE),1)]
    ### iii
    kid <- crossover(X[parents[1]],X[parents[2]])
    if(runif(1)<= mutprob){
      kid<-mutate(kid)
    }
    ## iv
    # victim is replaced by the kid in the population and the vector Values is
    #updated
    victim <- kid
    X[victim_idx] <- kid
    Values[victim_idx] <- GeneticFun(kid)
    #v
    maximal[i] <- sort(Values,decreasing = TRUE)[1]
    ##cat("maximal",maximal,"\n")

    }

  ### V add plot to current Value


  df2 <- data.frame(X=X,Values=Values)
  df2 <- df2[order(df2$Values,decreasing = TRUE),]
  gg4<- ggplot() +
    geom_point(data=init_population, aes(X,Values,colour='initial population'))+
    geom_point(data=df2, aes(X,Values,colour = "final"))
  gg4
  }
```

1.5 *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?*

```
set.seed(12345)
p1 <- Q4Fun(maxiter = 10,mutprob = 0.1) +ggtitle("maxiter = 10,mutprob = 0.1")
set.seed(12345)
p2<- Q4Fun(10,0.5)+ ggtitle("maxiter = 10,mutprob = 0.5")
set.seed(12345)
p3<- Q4Fun(10,0.9)+ ggtitle("maxiter = 10,mutprob = 0.9")
set.seed(12345)
p4<- Q4Fun(100,0.1)+ ggtitle("maxiter = 100,mutprob = 0.1")
set.seed(12345)
p5<- Q4Fun(100,0.5) + ggtitle("maxiter = 100,mutprob = 0.5")
set.seed(12345)
p6<- Q4Fun(100,0.9)+ ggtitle("maxiter = 100,mutprob = 0.9")
library(gridExtra)
```

```
## Warning: package 'gridExtra' was built under R version 4.0.3
```

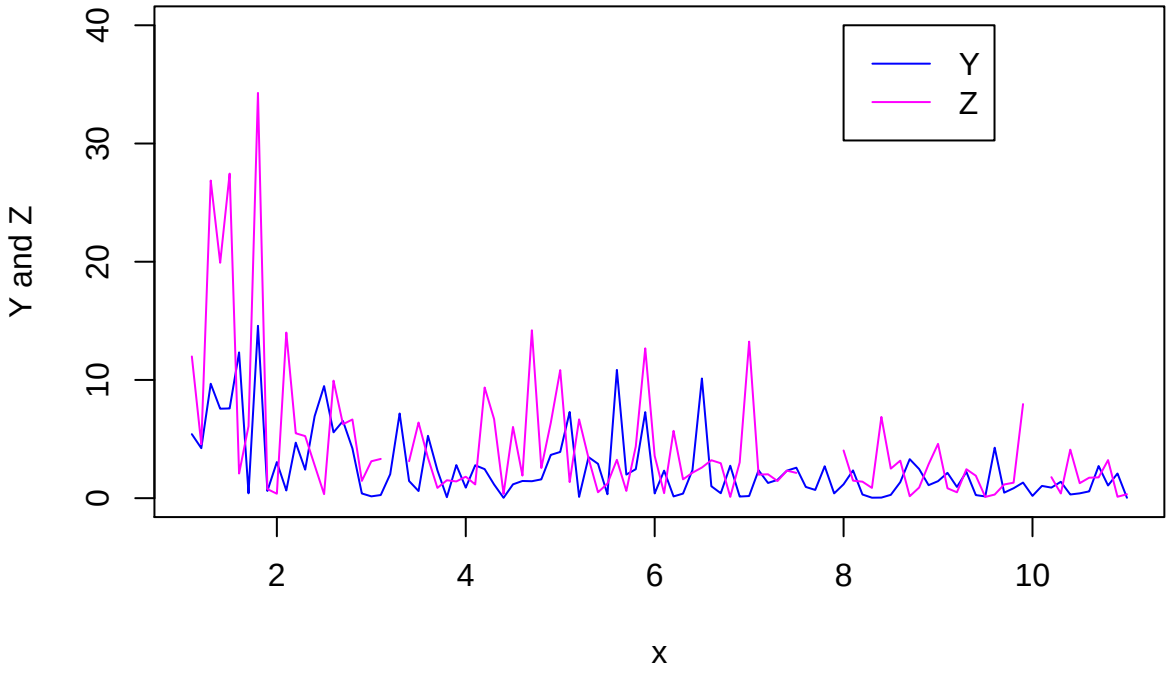
```
genpicture<-grid.arrange(p1,p4,p2,p5,p3,p6,ncol=2)
```



The higher the number of iteration the more accuracy we obtain. when `maxiter` equals to 10 there is a higher chance to stuck at a local `maximum`. Our second observation, is when the value of mutprob is low there is a higher chance for the algorithm to converge to the global maximum.

Question 2: EM algorithm

1. ***The data file physical.csv describes a behavior of two related physical processes*** $Y = Y(X)$ ***and*** $Z = Z(X)$***. Make a time series plot***



From the above plot we can observe that Z has some missing values , hence the discontinuity in plot. Z has a lot of peaks , intensity of peaks are higher at the beginning ie when x values are small and with increase in X the peak intensity reduces. Y also has a similar pattern as Z. For both Y and Z we can say that the variation when compared to X is high for smaller values of X and low for bigger values of X.

2. ***Derive an EM algorithm that estimates*** λ

There are missing values in Z. The model of Y and Z are as follows

$$Y_i \sim exp(\frac{X_i}{\lambda})$$
$$Z_i \sim exp(\frac{X_i}{2\lambda})$$

The goal is to derive an EM algorithm that estimates.

Since Y and Z follow exponential distribution, pdf of these can be stated as follows

$$f(Y|X,\lambda) = \frac{X_i}{\lambda} exp(-\frac{X_i Y_i}{\lambda})$$
$$f(Z|X,\lambda) = \frac{X_i}{2\lambda} exp(-\frac{X_i Z_i}{2\lambda})$$

Liklihood of lambda :

$$L(\lambda) = \prod_{i=1}^{n} \frac{X_i}{\lambda} exp(-\frac{X_i Y_i}{\lambda}) \prod_{i=1}^{n} \frac{X_i}{2\lambda} exp(-\frac{X_i Z_i}{2\lambda})$$

$$L(\lambda) = \prod_{i=1}^{n} \frac{X_i^2}{2\lambda^2} exp(-\frac{X_i Y_i}{\lambda}) exp(-\frac{X_i Z_i}{2\lambda})$$

$$L(\lambda) = (\frac{1}{2\lambda^2})^n exp(-\sum_{i=1}^{n} \frac{X_i Y_i}{\lambda} + \frac{X_i Z_i}{2\lambda}) \prod_{i=1}^{n} X_i^2$$

Log liklihood :

$$\log(L(\lambda)) = -n\log(2) - 2n\log(\lambda) - \sum_{i=1}^{n} \frac{X_i Y_i}{\lambda} - \sum_{i=1}^{n} \frac{X_i Z_i}{2\lambda} + \sum_{i=1}^{n} X_i^2$$

Terms which do not depend on lambda are can be written as C(constants).

$$\log(L(\lambda)) = -2n\log(\lambda) - \sum_{i=1}^{n} \frac{X_i Y_i}{\lambda} - \sum_{i=1}^{n} \frac{X_i Z_i}{2\lambda} + C$$

We know that Z has missing values , so this can be split in to two parts. Z with known values :

$$\sum_{known} \frac{X_i Z_i}{2\lambda}$$

Z with unknown values :

$$\sum_{unknown} \frac{X_j Z_j}{2\lambda}$$

$$\sum_{i=1}^{n} \frac{X_i Z_i}{2\lambda} = \sum_{known} \frac{X_i Z_i}{2\lambda} + \sum_{unknown} \frac{X_j Z_j}{2\lambda}$$

Substituting this to log liklihood :

$$\log(L(\lambda)) = -2n\log(\lambda) - \sum_{i=1}^{n} \frac{X_i Y_i}{\lambda} - \sum_{known} \frac{X_i Z_i}{2\lambda} - \sum_{unknown} \frac{X_j Z_j}{2\lambda} + C$$

Expected log liklihood :

$$E[\log(L(\lambda))] = E\left[-2n\log(\lambda) - \sum_{i=1}^{n} \frac{X_i Y_i}{\lambda} - \sum_{known} \frac{X_i Z_i}{2\lambda} - \sum_{unknown} \frac{X_j Z_j}{2\lambda} + C\right]$$

All terms other than the summation of unknown Z are on observed values. So the uncertainity is only in this term, so we need to calculate expected value for this term and rest are constant values hence expected value is the same. Zj represents the missing Z values , this can be estimated by using exponential distribution as follows:

$$E[Z_j] = \frac{2\lambda_{prev}}{X_j}$$

$$E\left[\sum_{unknown} \frac{X_j Z_j}{2\lambda}\right] = \sum_{unknown} \frac{X_j}{2\lambda} \frac{2\lambda_{prev}}{X_j} = \frac{U\lambda prev}{\lambda}$$

Where U is the number of unknown values in Z.

E-Step is given by :

$$E[\log(L(\lambda))] = -2n\log(\lambda) - \sum_{i=1}^{n} \frac{X_i Y_i}{\lambda} - \sum_{known} \frac{X_i Z_i}{2\lambda} - \frac{U\lambda prev}{\lambda} + C$$

For M step we just need to differentiate expected log liklihood wrt to lambda and equate it to 0 to find the estimate of lambda value, this will be the lambda value for next iteration.

$$\frac{\partial E[\log(L(\lambda))]}{\partial \lambda} = \frac{-2n}{\lambda} + \frac{1}{\lambda^2}\sum_{i=1}^{n} X_i Y_i + \frac{1}{2\lambda^2}\sum_{known} X_i Z_i + \frac{U\lambda_{prev}}{\lambda^2} = 0$$

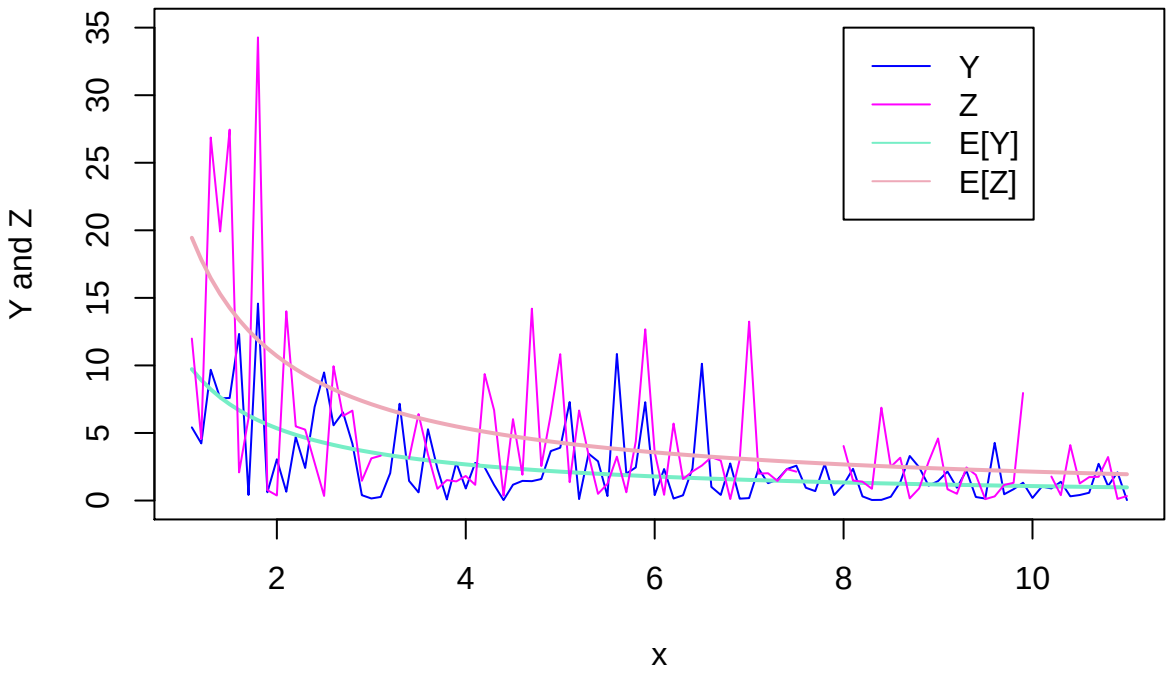
$$\lambda_{mle} = \frac{1}{2n}\sum_{i=1}^{n} X_i Y_i + \frac{1}{4n}\sum_{known} X_i Z_i + \frac{U\lambda_{prev}}{2n}$$

3. *Implement this algorithm in R use* $\lambda_0 = 100$ *and convergence criterion "stop if the change in* λ *is less than 0.001*

```
n = nrow(physical)
U = length(which(is.na(physical$Z)))
known_z = which(!is.na(physical$Z))
count = 1
lambda_diff = Inf
eps = 0.001
lambda = 100
repeat{
  lambda_new = (sum(physical$X * physical$Y) + (0.5 * sum(physical$X[known_z] * physical$Z[known_z])) +
  lambda_diff = abs(lambda - lambda_new)
  lambda = lambda_new
  count = count + 1
  if(lambda_diff < eps){break}
}
```

Optimal lambda is 10.6956555 Number of iterations are 6

4. *Plot* $E[Y]$ *and* $E[Z]$ *versus* X



As Y and Z follow exponential distribution , their expected value will be as follows.

$$E[Y] = \frac{\lambda}{X_i}$$

$$E[Z] = \frac{2\lambda}{X_i}$$

From the graph we can say that the computed lambda seems to be reasonable as both E[Y] and E[Z] lines seem to represent the mean of the actual plot pretty well. The variation with respect to X is higher in Z when compared to Y and even this is captured as the mean line of Z is higher than that of Y

Appendix

```
#1.1
GeneticFun <- function(x){

  return(x^2/exp(x)-2*exp(-(9*sin(x))/(x^2+x+1)))
}
#1.2
crossover <- function(x,y){

  kid <- (x+y)/2
  return(kid)
}
#1.3
mutate <- function(x){

  return((x^2 ) %% 30)
}
library(ggplot2)
  #4.a plot
  x <- seq(0,30,0.5)
  globalmax <- max(GeneticFun(x))
  cat("global max is",globalmax, "at x = ",
      x[order(GeneticFun(x),decreasing = TRUE)[1] ])
  #plot(GeneticFun(x))
  gg <- ggplot(data.frame(GeneticFun(x))) +
    geom_point(aes(x,GeneticFun(x)),color='darkblue')
  gg


Q4Fun<-function(maxiter,mutprob){


  # 4.b define initial pop
  X <- seq(0,30,5)
  # 4.c Computes vector Values
  Values <- GeneticFun(X)
  init_population <- data.frame("X"=X,"Values" = Values)



  #4.d

  maximal = c()

  for (i in 1:maxiter) {
    #i
    parents <- sample(x = length(X),size = 2,replace = FALSE)
    ## ii
    victim_idx <-tail(order(Values,decreasing = TRUE),1)
    victim <- X[tail(order(Values,decreasing = TRUE),1)]
    ### iii
    kid <- crossover(X[parents[1]],X[parents[2]])
```

```
    if(runif(1)<= mutprob){
      kid<-mutate(kid)
    }
    ## iv
    # victim is replaced by the kid in the population and the vector Values is
    #updated
    victim <- kid
    X[victim_idx] <- kid
    Values[victim_idx] <- GeneticFun(kid)
    #v
    maximal[i] <- sort(Values,decreasing = TRUE)[1]
    ##cat("maximal",maximal,"\n")

    }

  ### V add plot to current Value


  df2 <- data.frame(X=X,Values=Values)
  df2 <- df2[order(df2$Values,decreasing = TRUE),]
  gg4<- ggplot() +
    geom_point(data=init_population, aes(X,Values,colour='initial population'))+
    geom_point(data=df2, aes(X,Values,colour = "final"))
  gg4
  }

set.seed(12345)
p1 <- Q4Fun(maxiter = 10,mutprob = 0.1) +ggtitle("maxiter = 10,mutprob = 0.1")
set.seed(12345)
p2<- Q4Fun(10,0.5)+ ggtitle("maxiter = 10,mutprob = 0.5")
set.seed(12345)
p3<- Q4Fun(10,0.9)+ ggtitle("maxiter = 10,mutprob = 0.9")
set.seed(12345)
p4<- Q4Fun(100,0.1)+ ggtitle("maxiter = 100,mutprob = 0.1")
set.seed(12345)
p5<- Q4Fun(100,0.5) + ggtitle("maxiter = 100,mutprob = 0.5")
set.seed(12345)
p6<- Q4Fun(100,0.9)+ ggtitle("maxiter = 100,mutprob = 0.9")
library(gridExtra)
genpicture<-grid.arrange(p1,p4,p2,p5,p3,p6,ncol=2)
physical = read.csv("physical1.csv")
plot(physical$X,physical$Y,xlab = "x", ylab = "Y and Z", col = "blue", type = "l",ylim = c(0,40))
lines(physical$X,physical$Z,col="magenta")
legend(8,40,legend = c("Y","Z"),lty = 1, col = c("blue","magenta"))


n = nrow(physical)
U = length(which(is.na(physical$Z)))
known_z = which(!is.na(physical$Z))
count = 1
lambda_diff = Inf
eps = 0.001
lambda = 100
```

```r
repeat{
  lambda_new = (sum(physical$X * physical$Y) + (0.5 * sum(physical$X[known_z] * physical$Z[known_z])) +
  lambda_diff = abs(lambda - lambda_new)
  lambda = lambda_new
  count = count + 1
  if(lambda_diff < eps){break}
}

e_y = lambda/physical$X
e_z = 2*lambda/physical$X
plot(physical$X,physical$Y,xlab = "x", ylab = "Y and Z", col = "blue", type = "l",ylim = c(0,35))
lines(physical$X,e_y,col="aquamarine2",lwd =2)
lines(physical$X,physical$Z,col="magenta")
lines(physical$X,e_z,col="pink2",lwd = 2)
legend(8,35,legend = c("Y","Z","E[Y]","E[Z]"),lty = 1, col = c("blue","magenta","aquamarine2","pink2"))
```