732A90-Lab6-Report

Pass

Zhixuan Duan(zhidu838) Fengjuan Chen(fench417)

3/6/2020

Question 1: Genetic algorithm

1. Define the function

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

```
f <- function(x){
  f_x=x^2/exp(1)^x-2*exp(-9*sin(x)/(x^2+x+1))
  return(f_x)
}
```

2. Define the function crossover()

For two scalars x and y, crossover() function will return their kid as $(x + y)/2$, namely $crossover(x, y) = \frac{x+y}{2}$.

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

3. Define the function mutate()

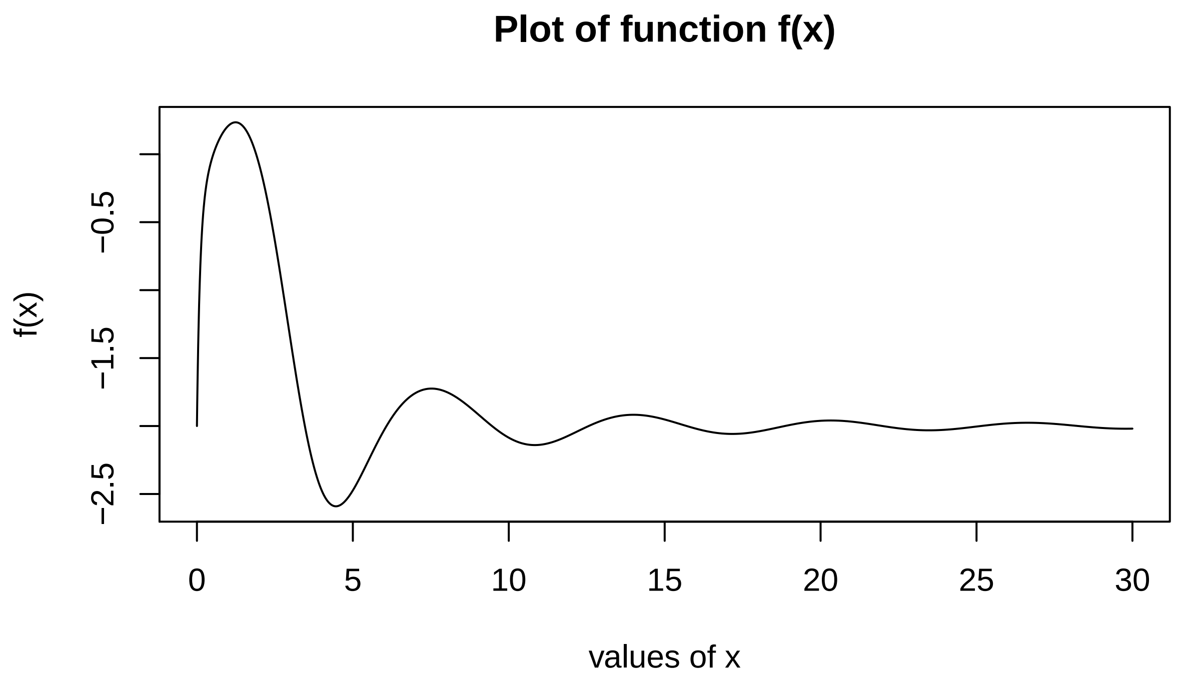
For a scalar x, mutate() will return the result of the integer division x^2 mod 30, namely $mutate(x) = x^2\%\%30$.

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

4. Implement a genetic algorithm

(a) Plot function f(x) in the range from 0 to 30.

```
x_seq=seq(0,30,0.01)
y=f(x_seq)
plot(x_seq,y,type = "l",xlab = "values of x",ylab ="f(x)",main = "Plot of function f(x)")
cat("The maximum of the function in the region [0,30] is",max(y))
```



```
## The maximum of the function in the region [0,30] is 0.2348527
```

From the plot there is an obvious maximum value of $f(x)$ near x is equal to 1.

(b) +(c) Define an initial population and calculate the function values

$X = (0, 5, 10, 15, ..., 30)$.

```
x_init=seq(0,30,5)
values=f(x_init)
```

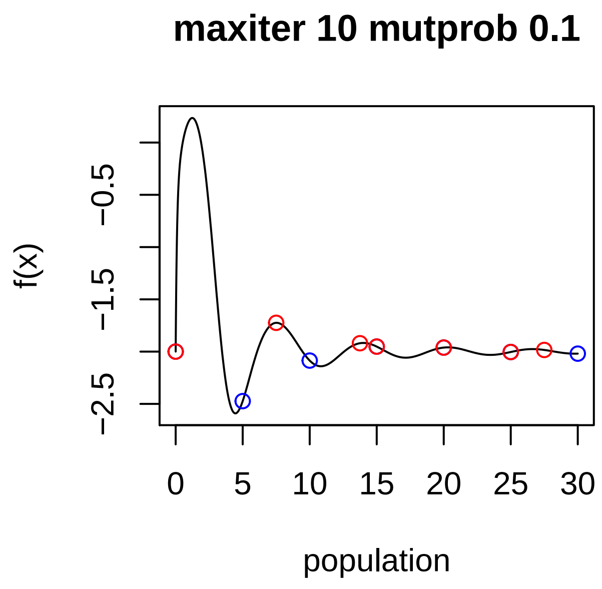
(d)+(e) Function of genetic algorithm

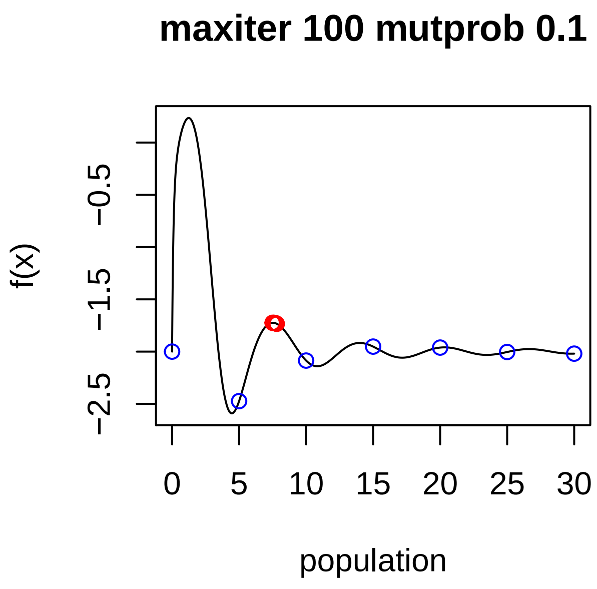
```
set.seed(12345)
genetic <- function(maxiter, mutprob){
  x_seq=seq(0,30,0.01)
  plot(x_seq,f(x_seq),type = "l",
       xlab = "population", ylab = "f(x)"
       main =paste("maxiter",maxiter,"mutprob",mutprob))
  population=seq(0,30,5)
  points(population,f(population),col="blue")
  current_max=max(values)
  for (i in 1:maxiter) {
    ind=sample(population,2)
    victim=order(values)[1]
    new_kid=crossover(ind[1],ind[2])
    u=runif(1)
    if(u<=mutprob){
      new_kid=mutate(new_kid)
    }
    population[victim]=new_kid
    values=f(population)
    current_max=max(values)
  }
  points(population,values,col="red")
  res=list(current_max,population)
  return(res)
}
```

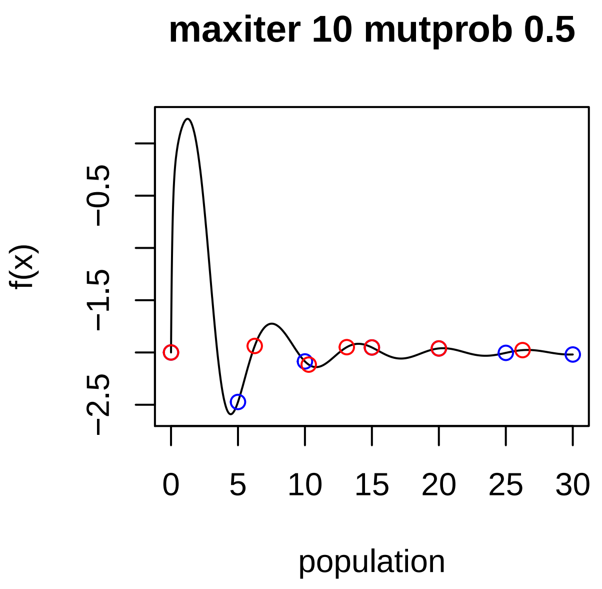
5. Run genetic algorithm with different parameters

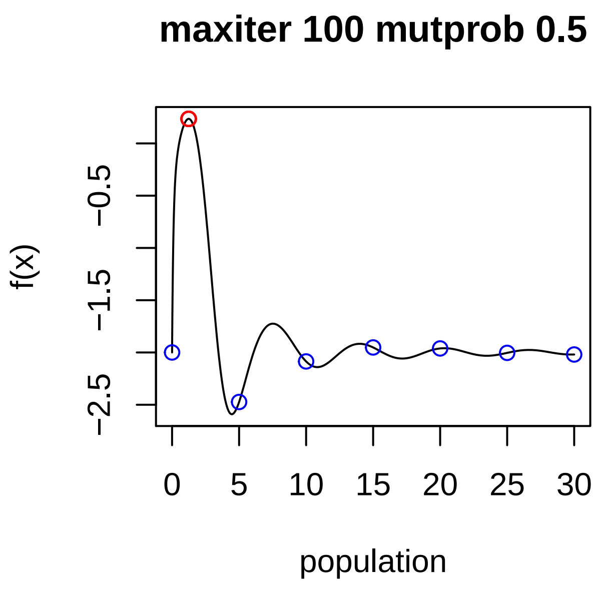
```
for (i in c(0.1,0.5,0.9)) {
  genetic(10,i)
  genetic(100,i)
}
```

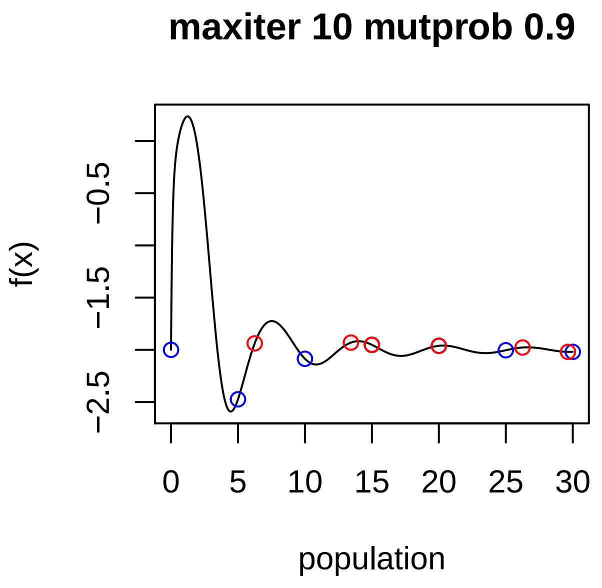
Conclusions: The number of maxiter and the probability of mutation both effect the results of final population. The elements in initial population have lower values of the objective function. Increasing either the number of maxiter or the probability of mutation can both improve the values of the objective functions. This is because these two parameters in our genetic algorithm can both tune the proportion to the mutation. And in our genetic algorithm the child obtained from crossover is the middle of their parents, so this kind of children does not change significantly from their parents. While the mutated child will have a big change comparing with the other non-mutated child. Therefore, if the mutation proportion is increased, the final population will have a significant difference with the initial population.











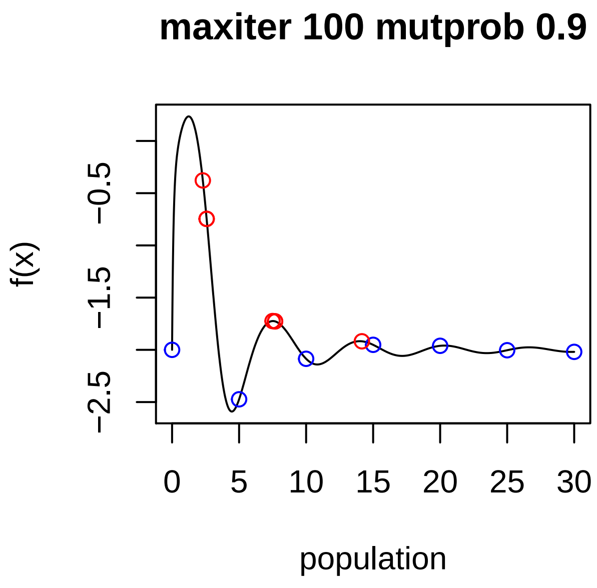


Table 1: the original and the final populations

Good

original	ite10_pro0.1	ite100_pro0.1	ite10_pro0.5	ite100_pro0.5	ite10_pro0.9	ite100_pro0.9
0	0.00	7.705984	0.00000	1.233179	13.43444	7.494428
5	13.75	7.705984	6.25000	1.238986	29.63906	2.578678
10	7.50	7.602992	26.25000	1.240106	15.00000	2.294006
15	15.00	7.705984	15.00000	1.254976	15.00000	7.684867
20	20.00	7.808976	20.00000	1.226252	20.00000	14.160156
25	25.00	7.602992	10.28229	1.236643	26.25000	7.684867
30	27.50	7.500000	13.12500	1.222997	6.25000	2.578678

Question 2: EM algorithm

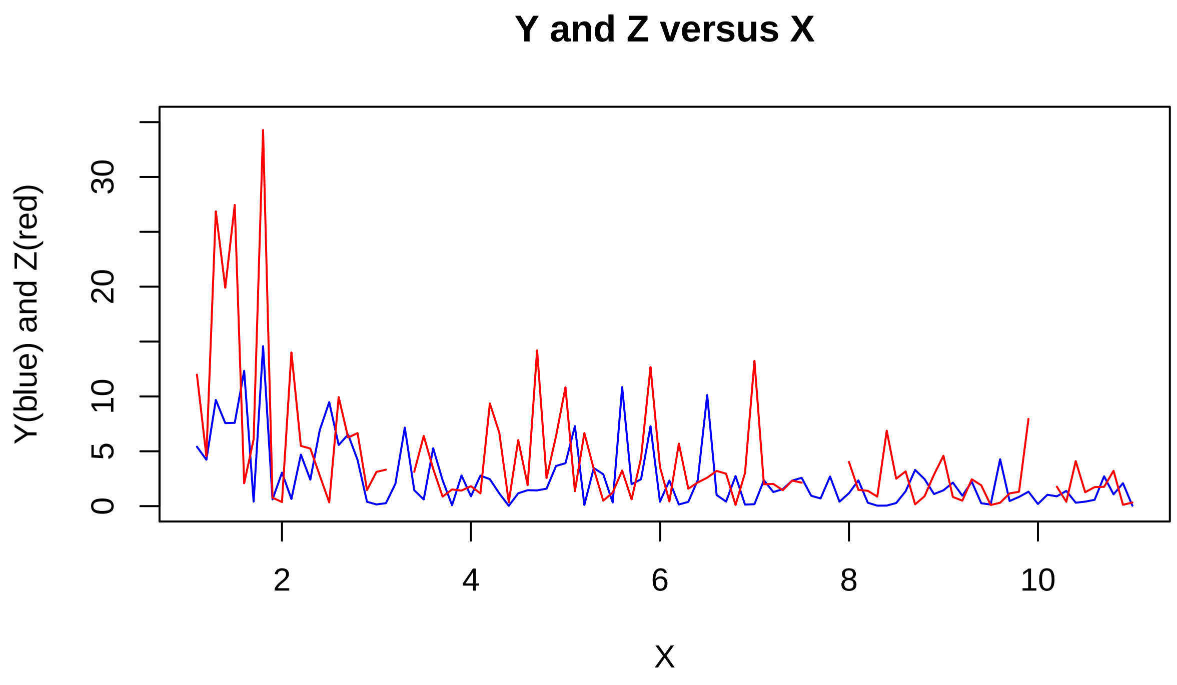
1. Make a time series plot of Z and Y versus X

Import the csv file and plot Y versus X and Z versus X in the same picture.

```r
data=read.csv("physical.csv")
plot(data$X,data$Y,col="blue",type = "l",xlab = "X",
     ylab = "Y(blue) and Z(red)",ylim = c(0,35),
     ,main = "Y and Z versus X")
lines(data$X,data$Z,col="red")
```

From the plot, it seems that two processes Y and Z are related to each other.

The response values with respect to X decayed as time passed.



2. Use models of Y and Z with unknown parameter lambda

The models of Y and Z with unknown parameter λ are:

$Y_i \sim exp(X_i/\lambda)$

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

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

We calculate the likelihood of λ:

$L(\lambda) = \prod_{i=1}^{n}(\frac{X_i}{\lambda}e^{-\frac{X_i}{\lambda}Y_i} \cdot \frac{X_i}{2\lambda}e^{-\frac{X_i}{2\lambda}Z_i}) = \frac{\sum_{i=1}^{n} X_i^2}{2^n\lambda^{2n}}e^{-\frac{1}{\lambda}\sum_{i=1}^{n}(X_iY_i+\frac{1}{2}X_iZ_i)}$ Wrong. The summation outside the X_i should be product instead.

The log likelihood of λ is

Still wrong here at the first term But you are lucky because when you take the derivative the entire first term become zero either ways...

$logL(\lambda) = ln(\sum_{i=1}^n X_i^2) - nln2 - 2nln\lambda - \frac{1}{\lambda}\sum_{i=1}^n(X_iY_i + \frac{1}{2}X_iZ_i)$

$= ln(\sum_{i=1}^n X_i^2) - nln2 - 2nln\lambda - \frac{1}{\lambda}\sum_{i=1}^n X_iY_i - \frac{1}{\lambda}\sum_{i=1}^n(\frac{1}{2}X_iZ_i)$

If we have all observations about Y_i and Z_i, we can use MLE to directly calculate the estiamtion of λ. However, there are some missing values of Z in the data. To obtain the solutions of MLE, we use the EM method.

Since Z_i is an exponential distribution with parameter $\frac{X_i}{2\lambda}$, the mean of Z_i is equal to $\frac{2\lambda}{X_i}$. Each time we estimate a new λ from the EM algorithm, we can obtain the mean of each missing Z_i. We can use these means to replace the missing data in Z and then further evaluate the next new λ with the updated data of Z. We use λ_{old} to denote the estimated λ from the last iteration of EM algorithm and it is a constant.

With this idea, we divide the original Z into two parts, the observed Z and the missing Z, denoted by Z_{obs} and Z_{mis} respectively. And the missing data Zi will be represented by the mean of Z_i, namely $\frac{2\lambda_{old}}{X_i}$, where λ_{old} is a constant obtained by the last iteration of EM algorithm.

In the formula of $logL(\lambda)$, $\sum_{i=1}^n(\frac{1}{2}X_iZ_i)$ becomes the new formula as following:

$\sum_{i=1}^n(\frac{1}{2}X_iZ_i) = \frac{1}{2}\sum_{i\in obs}^n X_iZ_i + \sum_{i\in mis}^n(\frac{1}{2}X_iZ_i) = \frac{1}{2}\sum_{i\in obs}^n X_iZ_i + \sum_{i\in mis}^n(\frac{1}{2}X_i \cdot \frac{2\lambda_{old}}{X_i})$

$= \frac{1}{2}\sum_{i\in obs}^n X_iZ_i + \sum_{i\in mis}^n \lambda_{old}$

This equality doesn't hold. The entire LHS is a random variable but RHS is constant. You have to take the expected value of the LHS to get RHS.

Put this new form of $\sum_{i=1}^n(\frac{1}{2}X_iZ_i)$ back into the $logL(\lambda)$, we have

$logL(\lambda) = ln(\sum_{i=1}^n X_i^2) - nln2 - 2nln\lambda - \frac{1}{\lambda}\sum_{i=1}^n X_iY_i - \frac{1}{\lambda}\frac{1}{2}\sum_{i\in obs}^n X_iZ_i - \frac{1}{\lambda}\sum_{i\in mis}^n \lambda_{old}$. This is the formula to calculate the log likelihood of λ if we want to use the change of log likelihood as the convergence criterion .

Take the partial derivative on λ, we have

$\frac{\partial logL(\lambda)}{\partial\lambda} = -2n\frac{1}{\lambda} + \frac{1}{\lambda^2}\sum_{i=1}^n X_iY_i + \frac{1}{2\lambda^2}\sum_{i\in obs}^n X_iZ_i + \frac{1}{\lambda^2}\sum_{i\in mis}^n \lambda_{old}$.

Let this partial derivative equal to 0 and multiply λ^2 on both sides ($\lambda \neq 0$). We have

$-2n\lambda + \sum_{i=1}^n X_iY_i + \frac{1}{2}\sum_{i\in obs}^n X_iZ_i + \sum_{i\in mis}^n \lambda_{old} = 0$

$2n\lambda = \sum_{i=1}^n X_iY_i + \frac{1}{2}\sum_{i\in obs}^n X_iZ_i + \sum_{i\in mis}^n \lambda_{old}$

$\lambda = \frac{1}{2n}(\sum_{i=1}^n X_iY_i + \frac{1}{2}\sum_{i\in obs}^n X_iZ_i + \sum_{i\in mis}^n \lambda_{old})$ Correct but the steps are wrong.

3. Implement EM algorithm in R

We implement EM algorithm in R, using $\lambda_0 = 100$ and convergence criterion that change of λ is less than 0.001.

We use the formula

$\lambda = \frac{1}{2n}(\sum_{i=1}^n X_iY_i + \frac{1}{2}\sum_{i\in obs}^n X_iZ_i + \sum_{i\in mis}^n \lambda_{old})$

to estimate the parameter λ in M-step and assign the result of it to the λ_{old} if the difference between the old lambda and new lambda is less than the given value, 0.001. And the starting λ_{old} is the given value, $\lambda_0 = 100$.

The code of this EM algorithm is shown below.

```
z_obs=data$Z[!is.na(data$Z)]
x_obs=data$X[!is.na(data$Z)]
n=nrow(data)
mis=n - length(z_obs)

lambda_old=100
```

```
lambda_new=(sum(data$X*data$Y)+
            sum(x_obs*z_obs)/2+
            mis*lambda_old  ) / (2*n)
dif_lam=abs(lambda_new - lambda_old)
iter=1
cat("old_lambda   new_lambda    difference  iteration_number\n")
cat(lambda_old, lambda_new,dif_lam,iter,"\n")
while(dif_lam>=0.001){
  lambda_old= lambda_new
  lambda_new=(sum(data$X*data$Y)+
                sum(x_obs*z_obs)/2+
                mis*lambda_old  ) / (2*n)
  dif_lam=abs(lambda_new - lambda_old)
  iter=iter+1
  cat(lambda_old, lambda_new,dif_lam,iter,"\n")
}
```

```
## old_lambda   new_lambda    difference  iteration_number

## 100 14.26782 85.73218 1

## 14.26782 10.83853 3.429287 2
## 10.83853 10.70136 0.1371715 3
## 10.70136 10.69587 0.005486859 4
## 10.69587 10.69566 0.0002194744 5

## The optimal lambda is: 10.69566
```

Correct value

```
## The number of iterations is: 5
```

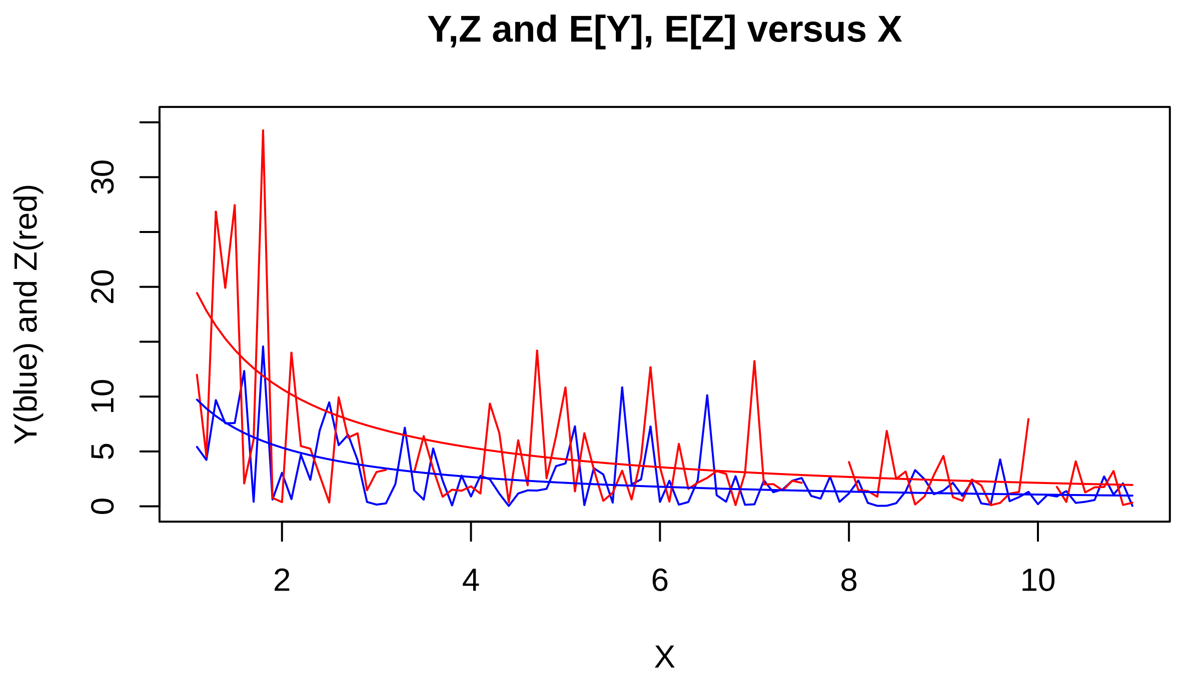
4. Plot E[Y] and E[Z] versus X in the same plot as Y and Z versus X

Because $Y_i \sim exp(X_i/\lambda)$ and $Z_i \sim exp(X_i/(2\lambda))$, we have
$E[Y] = \frac{\lambda}{X_i}$ and $E[Z] = \frac{2\lambda}{X_i}$ where λ is the final lambda we obtained from the EM algorithm.

```
EY=lambda_new/data$X
EZ=2*lambda_new/data$X

plot(data$X,data$Y,col="blue",type = "l",
     xlab = "X",ylab = "Y(blue) and Z(red)",
     ylim = c(0,35),main = "Y,Z and E[Y], E[Z] versus X")
lines(data$X,data$Z,col="red")
lines(data$X,EY,col="blue")
lines(data$X,EZ,col="red")
```

From this plot, we can see that the E[Y] and E[Z] are in the middle of the data Y and Z. We think the λ computed from EM algorithm is reasonable.



Appendix

```
# Question 1: Genetic algorithm

## 1. Define the function

f <- function(x){
  f_x=x^2/exp(1)^x-2*exp(-9*sin(x)/(x^2+x+1))
  return(f_x)
}


## 2. Define the function crossover()

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


## 3. Define the function mutate()

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

## 4. Implement the genetic algorithm
```

```r
### (a) Plot function in the range from 0 to 30.

x_seq=seq(0,30,0.01)
y=f(x_seq)
plot(x_seq,y,type = "l")
max(y)


### (b) Define an initial population for the genetic algorithm

x_init=seq(0,30,5)

### (c) Compute vector Values that contains the function values for each population point
values=f(x_init)


### (d)+(e) Function of the genetic algorithm

set.seed(12345)

genetic <- function(maxiter, mutprob){
  x_seq=seq(0,30,0.01)
  plot(x_seq,f(x_seq),type = "l",
       main = paste("maxiter",maxiter,"mutprob",mutprob))
  population=seq(0,30,5)
  points(population,f(population),col="blue")

  current_max=max(values)
  for (i in 1:maxiter) {
    ind=sample(population,2)
    victim=order(values)[1]
    new_kid=crossover(ind[1],ind[2])
    u=runif(1)
    if(u<=mutprob){
      new_kid=mutate(new_kid)
    }
    population[victim]=new_kid
    values=f(population)
    current_max=max(values)
  }

  points(population,values,col="red")
  res=list(current_max,population)
  return(res)
}

## 5. Run genetic algorithm with different parameters


comp=data.frame(population=x_init)
max_values=rep(0,6)
j=1
for (i in c(0.1,0.5,0.9)) {
  res1=genetic(10,i)
  comp=cbind(comp,res1[[2]])
  max_values[j]=res1[[1]]
  res2=genetic(100,i)
  comp=cbind(comp,res2[[2]])
  max_values[j+1]=res2[[1]]
  j=j+2
}
colnames(comp)=c("original","ite10_pro0.1",
                 "ite100_pro0.1","ite10_pro0.5",
                 "ite100_pro0.5","ite10_pro0.9",
                 "ite100_pro0.9")



# Question 2

## 1. Make a time series plot of Z and Y versus X

data=read.csv("physical.csv")

plot(data$X,data$Y,col="blue",type = "l",
     xlab = "X",ylab = "Y(blue) and Z(red)",
     ylim = c(0,35),main = "Y and Z versus X")
lines(data$X,data$Z,col="red")

## 2. Use models of Y and Z with unknow parameter lambda

## 3. Implement EM algorithm in R

z_obs=data$Z[!is.na(data$Z)]
x_obs=data$X[!is.na(data$Z)]
n=nrow(data)
mis=n - length(z_obs)

lambda_old=100
lambda_new=(sum(data$X*data$Y)+
              sum(x_obs*z_obs)/2+
              mis*lambda_old  ) / (2*n)
dif_lam=abs(lambda_new- lambda_old)
i=1
cat("old_lambda   new_lambda    difference  iteration_number\n")
cat(lambda_old, lambda_new,dif_lam,i,"\n")
while(dif_lam>=0.001){
  lambda_old= lambda_new
  lambda_new=(sum(data$X*data$Y)+
                sum(x_obs*z_obs)/2+
                mis*lambda_old  ) / (2*n)
  dif_lam=abs(lambda_new- lambda_old)
  i=i+1
  cat(lambda_old, lambda_new,dif_lam,i,"\n")
}
```

```r
## 4. plot E[Y] and E[Z]

EY=lambda_new/data$X
EZ=2*lambda_new/data$X

plot(data$X,data$Y,col="blue",type = "l",
     xlab = "X",ylab = "Y(blue) and Z(red)",
     ylim = c(0,35),main = "Y,Z and E[Y], E[Z] versus X")
lines(data$X,data$Z,col="red")
lines(data$X,EY,col="blue")
lines(data$X,EZ,col="red")
```