Computer Lab 6

Group-19 - Dinesh and Umamaheswarababu

2022-12-20

Statement of Contribution : Question 1 was mostly done by Dinesh and Question 2 was mostly done by Umamaheswarababu

Question 1: Genetic algorithm

In this assignment, you will try to perform one-dimensional maximization with the help of a genetic algorithm.

Define the function

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

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

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

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

Define the function mutate() that for a scalar x returns the result of the integer division x^2 mod 30. (Operation mod is denoted in R as %%).

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

Write a function that depends on the parameters maxiter and mutprob and:

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

(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

(i)Two indexes are randomly sampled from the current population, they are further used as parents (use sample()).

(ii) One index with the smallest objective function is selected from the current population, the point is referred to as victim (use order()).

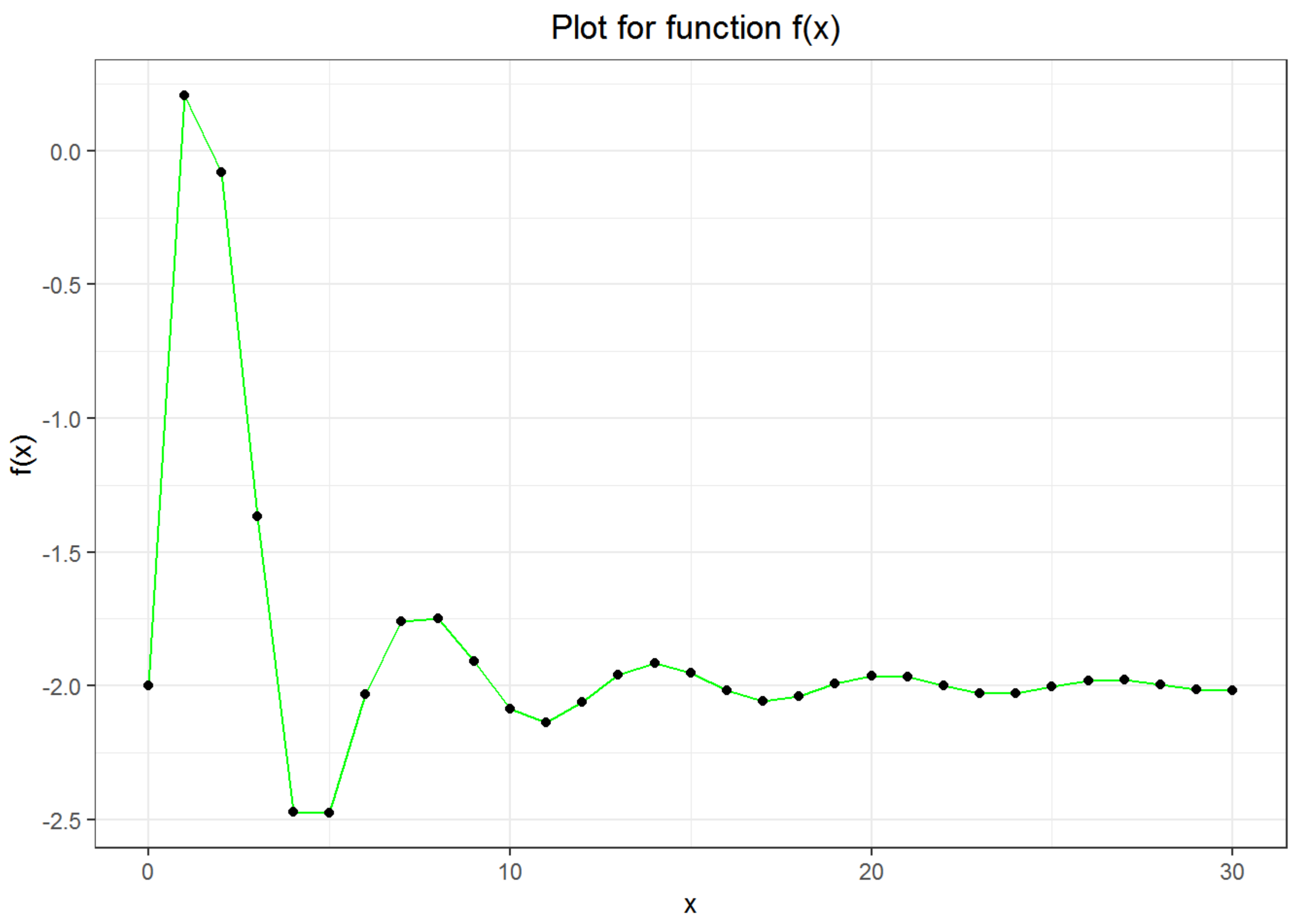
(iii) Parents are used to produce a new kid by crossover. Mutate this kid with probability mutprob (use crossover(), mutate()).

(iv) The victim is replaced by the kid in the population and the vector Values is updated.

(v) The current maximal value of the objective function is saved.

(e) Add the final observations to the current plot in another colour.

```r
#Task4(a)
 set.seed(12345)
 x <- seq(0, 30, by=1)
 y<- f(x)
 df1 <- data.frame(x, y)
library(ggplot2)
ggplot() +
  geom_line(data=df1, aes(x=x, y=f(x)), color="green") +
  geom_point(aes(x=x, y=f(x)), color="black") +
  theme_bw() +
  ggtitle(" Plot for function f(x)") +
  theme(plot.title = element_text(hjust = 0.5), legend.position = "right")
```


Plot for function f(x)
f(x)
0.0
-0.5
-1.0
-1.5
-2.0
-2.5
0
10
20
30
x

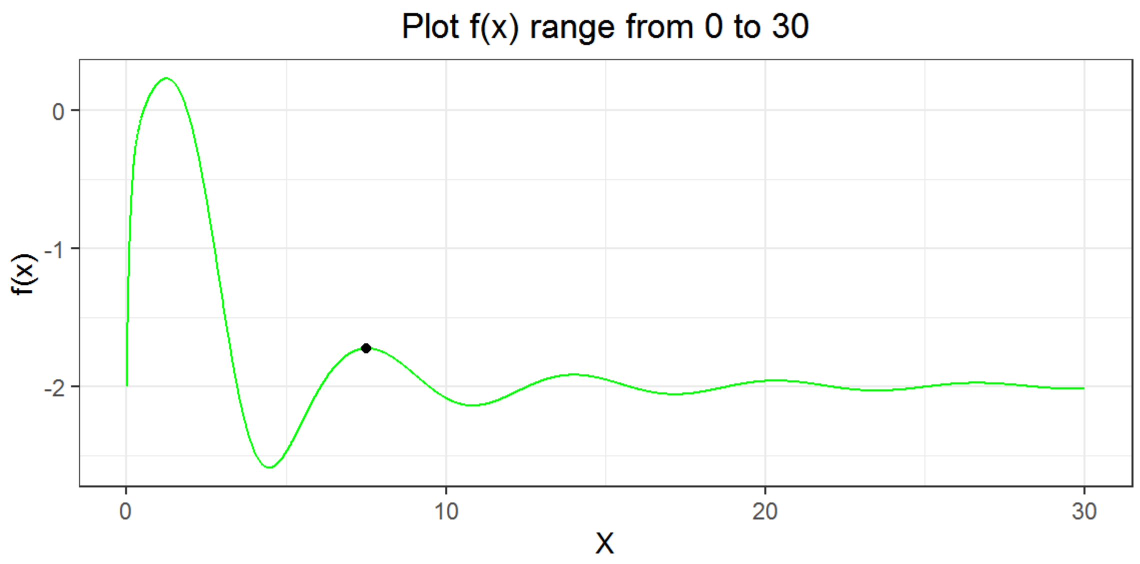
```r
GA<-function(maxiter,mutprob){
  best_val<-vector()
  # Task4(b)
  X<-seq(0,30,5)

  # Task 4(c)
  values<-f(X)
  df<-cbind(X,values)

  # Task 4(d)
  for(i in 1:maxiter){
    parents<-df[,1][sample(c(1:nrow(df)),size = 2)]
    victim<-order(df[,"values"])[1]
    #victim<-which.min(value)
    kid<-crossover(parents[1],parents[2])
    if(mutprob>runif(1)){
      kid<-mutate(kid)
    }
    kid1<-f(kid)
    df[victim,]<-c(kid,kid1)
    best_val<-rbind(best_val,df[order(df[,"values"])[nrow(df)],])
  }

  x_val1 <- seq(0, 30, by=0.01)
  y_val1 <- f(x_val1)
  df1 <- data.frame(x_val1,y_val1)
  x_val2<-best_val[maxiter,1]
  y_val2<-best_val[maxiter,2]

  library(ggplot2)
  g<-ggplot() +
    geom_line(data=df1, aes(x=x_val1, y=y_val1), color="green") +
    geom_point(aes(x=x_val2, y=y_val2), color="black") +
    theme_bw() +
    labs(y = "f(x)",x='X') +
    ggtitle("Plot f(x) range from 0 to 30") +
    theme(plot.title = element_text(hjust = 0.5), legend.position = "right")
  plot(g)

  return(df[,"X"])
}
```

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
#Initial population:
seq(0,30,5)
```

```
## [1]  0  5 10 15 20 25 30
```

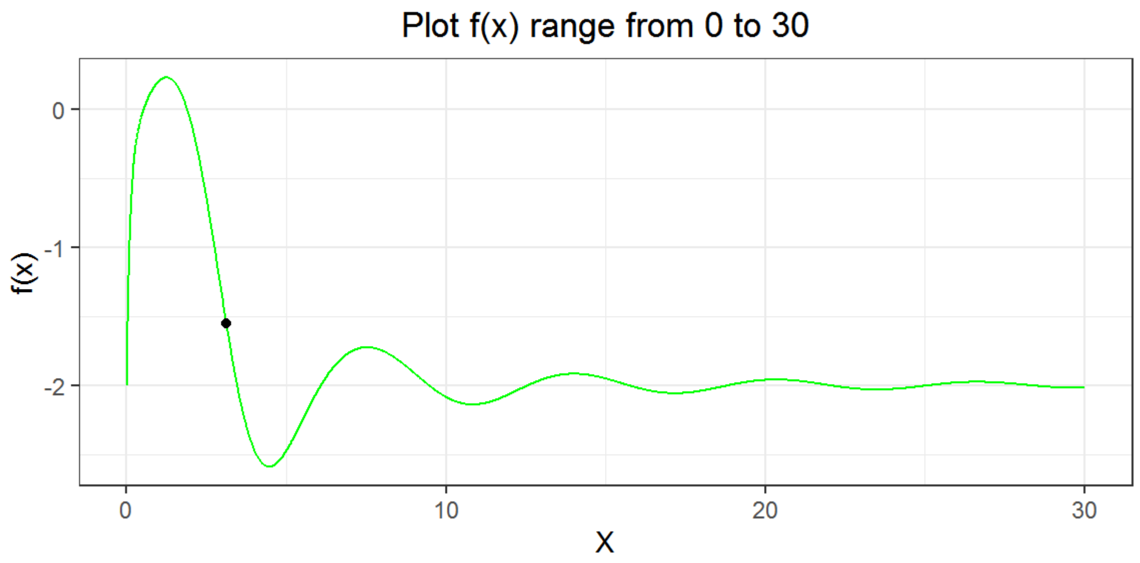
```
#Different combinations
#When maxiter=10, mutprob=0.1 is:
plot_1<-GA(10,0.1)
```



```
plot_1
```

```
## [1]  0.00 13.75  7.50 15.00 20.00 25.00 27.50
```

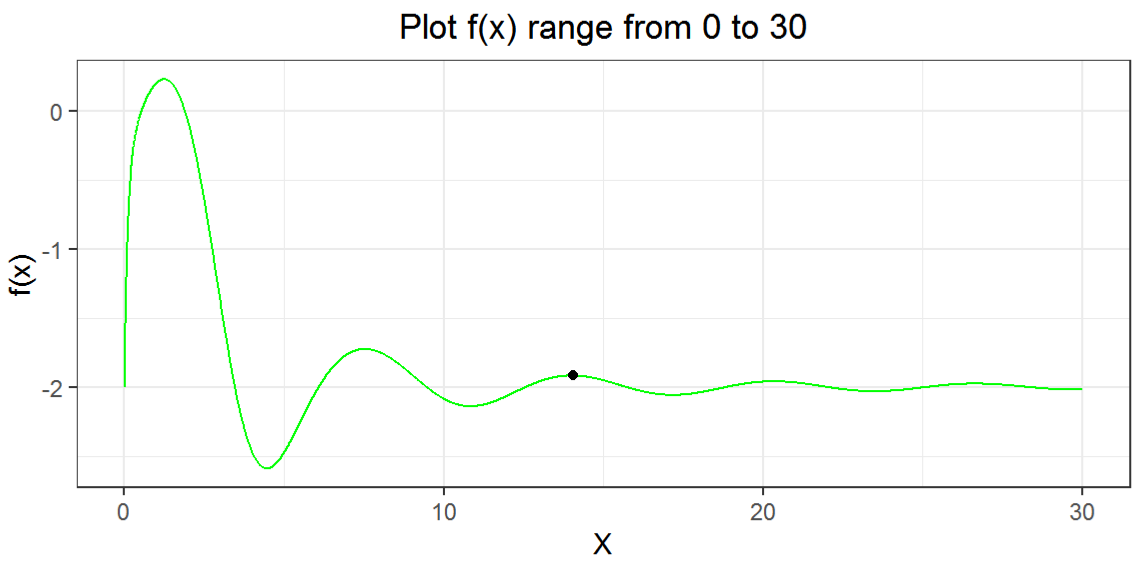
```
#When maxiter=10, mutprob=0.5 is:
plot_2<-GA(10,0.5)
```



```
plot_2
```

```
## [1]  3.12500 15.00000  6.25000 15.00000 21.97266 13.12500  7.50000
```

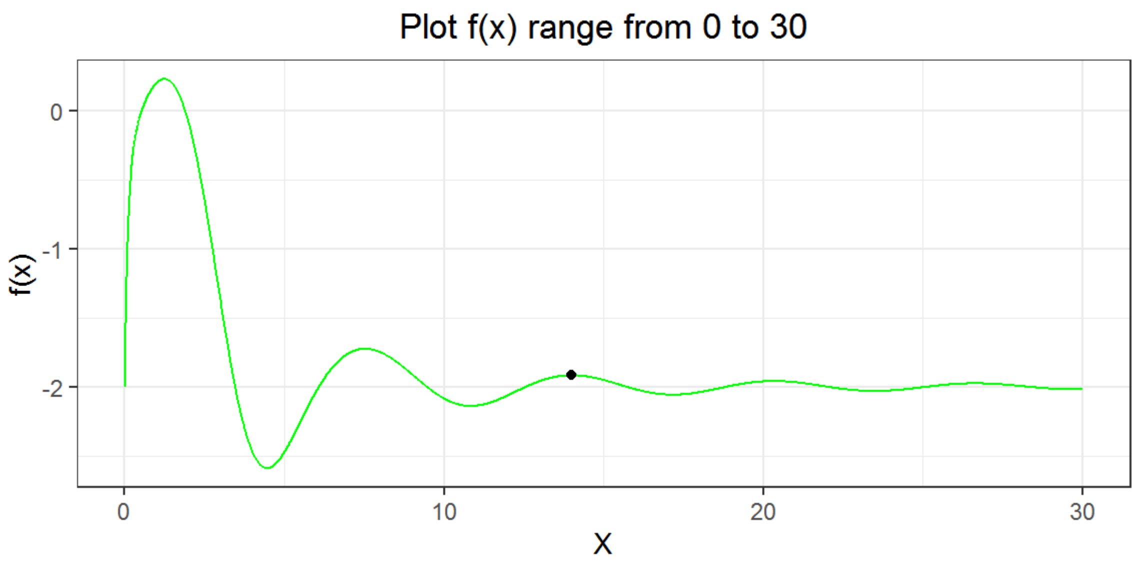
```
#When maxiter=10, mutprob=0.9 is:
plot_3<-GA(10,0.9)
```



```
plot_3
```

```
## [1] 13.125000 14.062500  6.250000 15.000000  3.847656  6.253928  9.227600
```

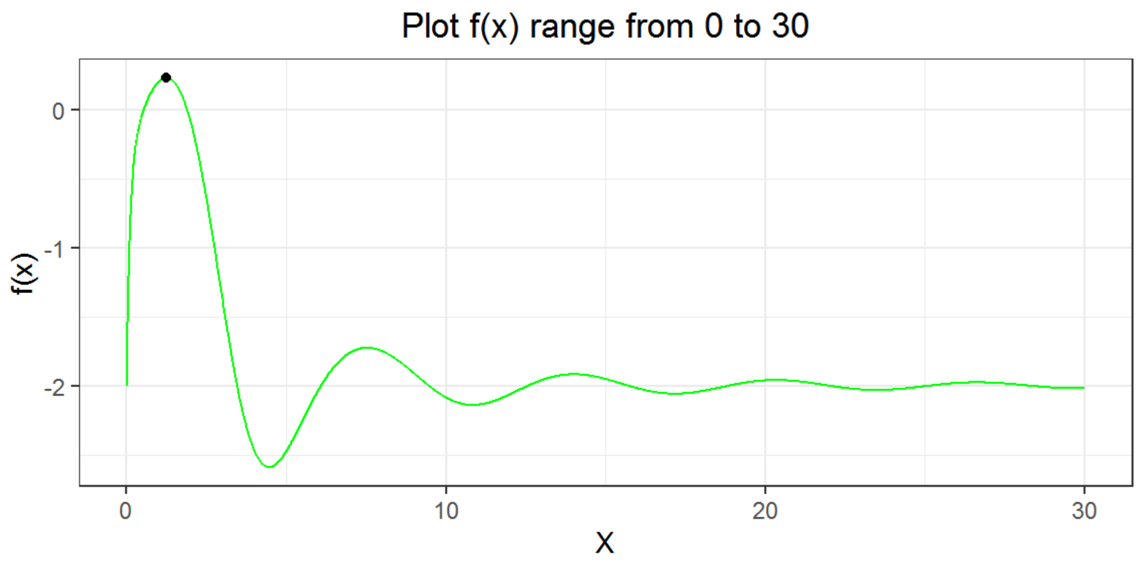
```
#When maxiter=100, mutprob=0.1 is:
plot_4<-GA(100,0.1)
```



```
plot_4
```

```
## [1] 13.99895 13.99895 13.99895 13.99895 13.99895 13.99895 13.99895
```

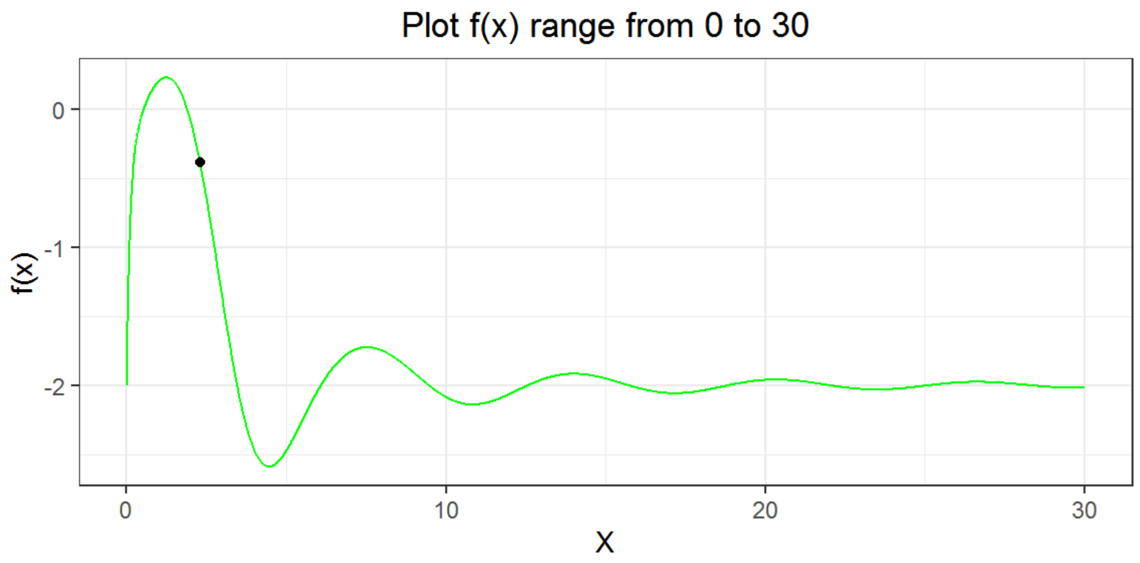
```
#When maxiter=100, mutprob=0.5 is:
plot_5<-GA(100,0.5)
```



```
plot_5
```

```
## [1] 1.239183 1.239153 1.239198 1.239123 1.239213 1.239243 1.239198
```

```
#When maxiter=100, mutprob=0.5 is:
plot_6<-GA(100,0.9)
```



```
plot_6
```

```
## [1]  7.494428  2.578678  2.294006  7.684867 14.160156  7.684867  2.578678
```

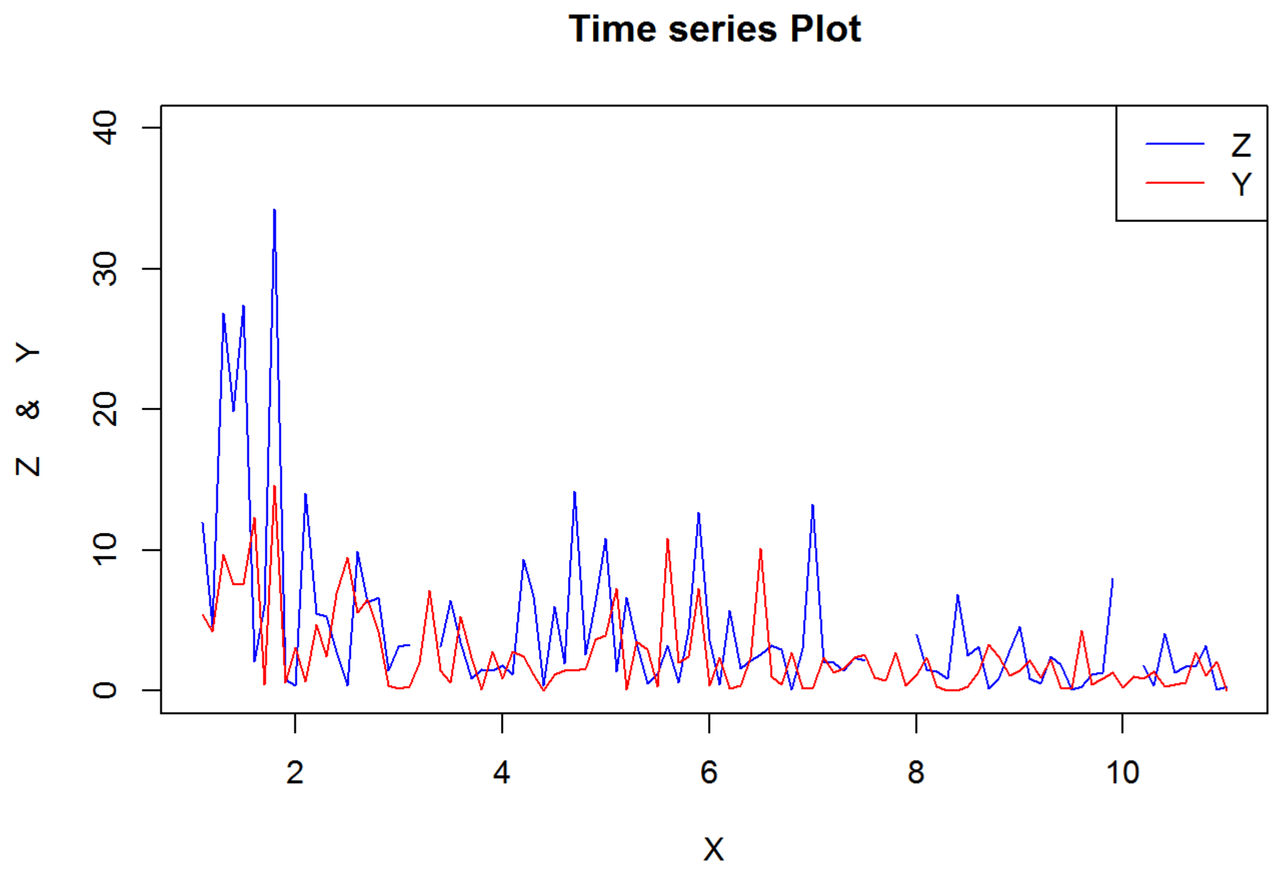
Analysis

Based on the nature of the algorithm we are using, we can see that the frequency of the mutation procedure is directly proportional to the value of the mutprob.The results and plots show that the values of maxiters and mutprobs significantly impact the outcome of iterations. When we observe the behavior of the plots, we can see that increasing the number of iterations increases the probability of the mutation procedure occurring. As a result, we are more likely to find better solutions with maximum values for our populations.

Question 2: EM Algorithm

Question 2.1:Make a time series plot describing dependence of Z and Y versus X. Does it seem that two processes are related to each other? What can you say about the variation of the response values with respect to X?

```
data<-read.csv("physical1.csv") #Loading the data

#Plotting dependence of Z and Y versus X
plot(data$X,data$Z, type='l', col="blue", ylim=c(0,40),xlab="X", ylab="Z    &    Y",main="Time s
eries Plot")
points(data$X,data$Y, type='l',col="red")
legend("topright",col=c("blue","red"),legend = c("Z","Y"),lty = c(1,1))
```



The two processes are related to each other. From the plot we can observe both Y and Z exhibit similar pattern of decayed oscillation. For most of the values of X, Z has higher amplitude compared to Y. We also can observe some missing values of Z.

Question 2.2 : 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 λ is some unknown parameter. **The goal is to derive an EM algorithm that estimates λ.**

Since Y and Z are independent, the probability density function is given by

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

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

Likelihood is given by

$$L(\lambda|X,Y,Z) = \prod P(X,Y,Z|\lambda)$$

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

Log-likelihood is given by

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

Z_i has some missing values so we split it into two parts as below

$$\sum_{i=1}^{m}Z_{obs} - - - - - > observed$$

$$\sum_{i=m+1}^{n}Z_{miss} - - - - > missing$$

The Log-Likelihood can be represented as

$$ln(L(\lambda)) = 2ln(\prod_{i=1}^{n}X_i) - 2nln(2\lambda) - \frac{1}{\lambda}(\sum_{i=1}^{n}X_iY_i + \sum_{i=1}^{m}\frac{X_iZ_{obs}}{2} + \sum_{i=m+1}^{n}\frac{X_iZ_{miss}}{2})$$

EM Algorithm to estimate λ :

The Q function is given by

$$Q(\lambda,\lambda_k) = E(loglik(\lambda|X,Y,Z)|\lambda_k,X,Y,Z)$$

$$Q(\lambda,\lambda_k) = E(ln(L(\lambda)))$$

$$= 2ln(\prod_{i=1}^{n}X_i) - 2nln(2\lambda) - \frac{1}{\lambda}(\sum_{i=1}^{n}X_iY_i + \sum_{i=1}^{m}\frac{X_iZ_{obs}}{2} + \sum_{i=m+1}^{n}\lambda_k)$$

To get λ, we differentiate above expresion with respect to λ and equal it to 0.

$$\frac{\partial E(ln(L(\lambda)))}{\partial \lambda} = 0$$

$$0 - \frac{2n}{\lambda} + \frac{1}{\lambda^2}(\sum_{i=1}^{n} X_i Y_i + \sum_{i=1}^{m} \frac{X_i Z_{obs}}{2} + (n-m)\lambda_k) = 0$$

where (n-m) indicates number of missing values of Z_i

Solving the above equation for λ, we get

$$\lambda = \frac{\sum_{i=1}^{n} X_i Y_i + \sum_{i=1}^{m} \frac{X_i Z_{obs}}{2} + (n-m)\lambda_k}{2n}$$

Question 2.3: Implement this algorithm in R, use λ_0 = 100 and convergence criterion “stop if the change in λ is less than 0.001”. What is the optimal λ and how many iterations were required to compute it?

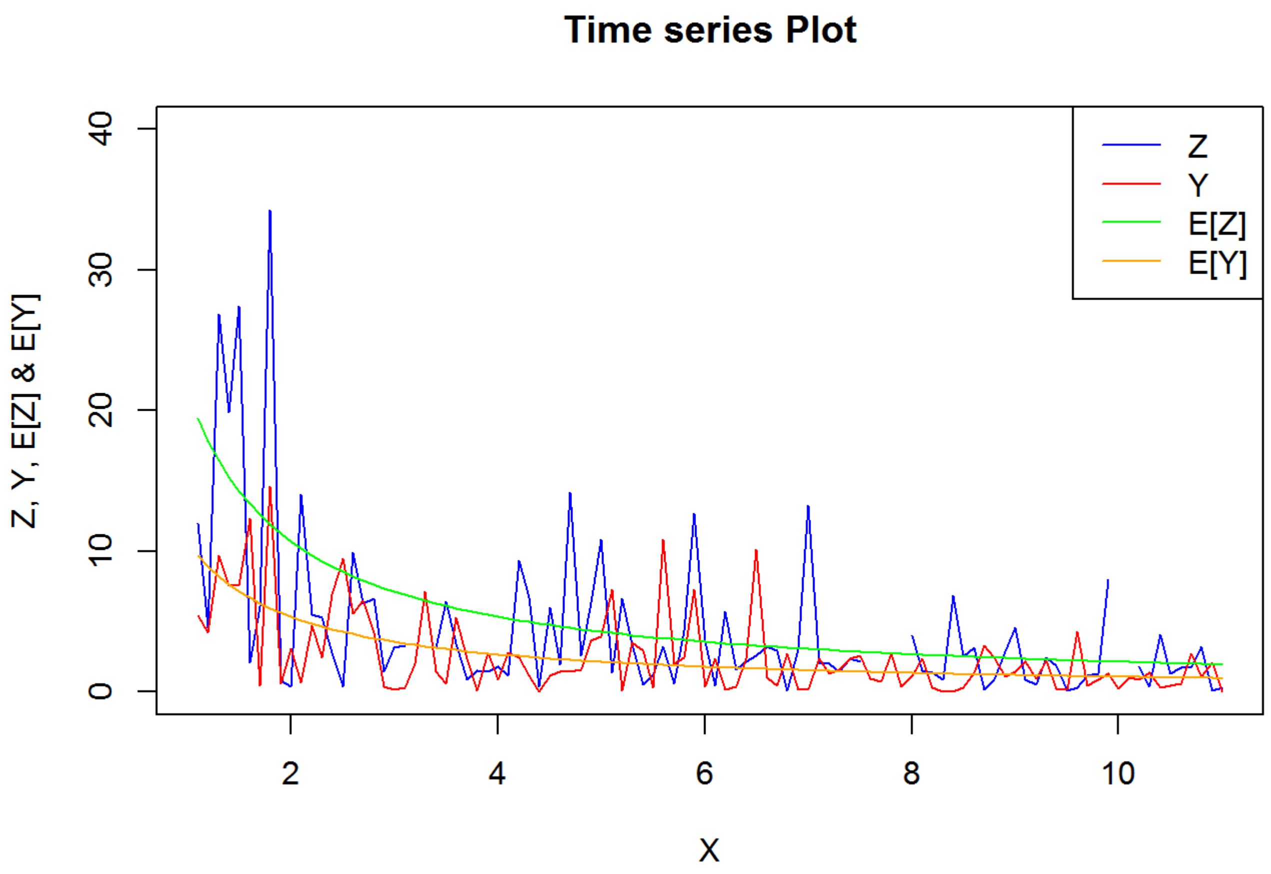
```
EM <-function(data,eps,lambda,kmax){
  X_obs <- data$X[!is.na(data$Z)] #Values of X for which values of Z are not missing
  Z_obs <- data$Z[!is.na(data$Z)] # Observable values of Z
  n <- length(data$X)             # Total number of observations
  m <- length(data$Z[is.na(data$Z)]) #Number of missing Z values
  k <- 0
  llvalprev <- 0
  llvalcurr <- lambda
  print(c(k,llvalprev,llvalcurr)) #Initial values

  #Reference: Lecture06codeSlide15.R
  while ((abs(llvalprev-llvalcurr) > eps && (k<(kmax+1)))){
    llvalprev <- llvalcurr
    EY <- (sum(data$X*data$Y) + sum(X_obs*Z_obs)/2 + m*llvalprev)/(2*n)
    llvalcurr <- EY
    print(c(k,llvalprev,llvalcurr))
    k <- k+1
  }
}
EM(data=data,eps=0.001,lambda=100,kmax=50)
```

```
## [1]   0   0 100
## [1]   0.00000 100.00000  14.26782
## [1]  1.00000 14.26782 10.83853
## [1]  2.00000 10.83853 10.70136
## [1]  3.00000 10.70136 10.69587
## [1]  4.00000 10.69587 10.69566
```

The optimal value of λ is 10.69566 and it took 4 iterations to compute the value.

Question 2.4: 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.

```
opt_lambda<-10.69566
E_Y<-opt_lambda/data$X #E[Y]
E_Z<-2*opt_lambda/data$X #E[Z]
plot(data$X,data$Z, type='l', col="blue", ylim=c(0,40),xlab="X", ylab="Z, Y, E[Z] & E[Y]",main
="Time series Plot")
points(data$X,data$Y, type='l',col="red")
points(data$X,E_Z,type = 'l',col="green")
points(data$X,E_Y,type='l',col="orange")
legend("topright",col=c("blue","red","green","orange"),legend = c("Z","Y","E[Z]","E[Y]"),lty = c
(1,1))
```



The computed λ seems to be reasonable because from the above plot we can observe the expected values E[Y] and E[Z] follow the trend of Y and Z.

Appendix

```r
knitr::opts_chunk$set(echo = TRUE)
f<-function(x)
{
  result <- (x^2/exp(x)) - 2*exp(-(9 * sin(x))/(x^2 + x + 1))
  return(result)
}
crossover<-function(x,y)
{
  return((x+y)/2)
}
mutate<-function(x){
  return(x^2%%30)
}

#Task4(a)
 set.seed(12345)
 x <- seq(0, 30, by=1)
 y<- f(x)
 df1 <- data.frame(x, y)
library(ggplot2)
ggplot() +
  geom_line(data=df1, aes(x=x, y=f(x)), color="green") +
  geom_point(aes(x=x, y=f(x)), color="black") +
  theme_bw() +
  ggtitle(" Plot for function f(x)") +
  theme(plot.title = element_text(hjust = 0.5), legend.position = "right")

GA<-function(maxiter,mutprob){
  best_val<-vector()
  # Task4(b)
  X<-seq(0,30,5)

  # Task 4(c)
  values<-f(X)
  df<-cbind(X,values)

  # Task 4(d)
  for(i in 1:maxiter){
    parents<-df[,1][sample(c(1:nrow(df)),size = 2)]
    victim<-order(df[,"values"])[1]
    #victim<-which.min(value)
    kid<-crossover(parents[1],parents[2])
    if(mutprob>runif(1)){
      kid<-mutate(kid)
    }
    kid1<-f(kid)
    df[victim,]<-c(kid,kid1)
    best_val<-rbind(best_val,df[order(df[,"values"])[nrow(df)],])
  }

  x_val1 <- seq(0, 30, by=0.01)
  y_val1 <- f(x_val1)
  df1 <- data.frame(x_val1,y_val1)
  x_val2<-best_val[maxiter,1]
  y_val2<-best_val[maxiter,2]

  library(ggplot2)
  g<-ggplot() +
    geom_line(data=df1, aes(x=x_val1, y=y_val1), color="green") +
    geom_point(aes(x=x_val2, y=y_val2), color="black") +
    theme_bw() +
    labs(y = "f(x)",x='X') +
    ggtitle("Plot f(x) range from 0 to 30") +
    theme(plot.title = element_text(hjust = 0.5), legend.position = "right")
  plot(g)

  return(df[,"X"])
}

#Initial population:
seq(0,30,5)
#Different combinations
#When maxiter=10, mutprob=0.1 is:
plot_1<-GA(10,0.1)
plot_1
#When maxiter=10, mutprob=0.5 is:
plot_2<-GA(10,0.5)
plot_2
#When maxiter=10, mutprob=0.9 is:
plot_3<-GA(10,0.9)
plot_3
#When maxiter=100, mutprob=0.1 is:
plot_4<-GA(100,0.1)
plot_4
#When maxiter=100, mutprob=0.5 is:
plot_5<-GA(100,0.5)
plot_5
#When maxiter=100, mutprob=0.5 is:
plot_6<-GA(100,0.9)
plot_6

data<-read.csv("physical1.csv") #Loading the data

#Plotting dependence of Z and Y versus X
plot(data$X,data$Z, type='l', col="blue", ylim=c(0,40),xlab="X", ylab="Z    &    Y",main="Time s
eries Plot")
points(data$X,data$Y, type='l',col="red")
legend("topright",col=c("blue","red"),legend = c("Z","Y"),lty = c(1,1))
EM <-function(data,eps,lambda,kmax){
  X_obs <- data$X[!is.na(data$Z)] #Values of X for which values of Z are not missing
  Z_obs <- data$Z[!is.na(data$Z)] # Observable values of Z
  n <- length(data$X)             # Total number of observations
  m <- length(data$Z[is.na(data$Z)]) #Number of missing Z values
  k <- 0
  llvalprev <- 0
  llvalcurr <- lambda
  print(c(k,llvalprev,llvalcurr)) #Initial values

  #Reference: Lecture06codeSlide15.R
  while ((abs(llvalprev-llvalcurr) > eps && (k<(kmax+1)))){
    llvalprev <- llvalcurr
    EY <- (sum(data$X*data$Y) + sum(X_obs*Z_obs)/2 + m*llvalprev)/(2*n)
    llvalcurr <- EY
    print(c(k,llvalprev,llvalcurr))
    k <- k+1
  }
}
EM(data=data,eps=0.001,lambda=100,kmax=50)

opt_lambda<-10.69566
E_Y<-opt_lambda/data$X #E[Y]
E_Z<-2*opt_lambda/data$X #E[Z]
plot(data$X,data$Z, type='l', col="blue", ylim=c(0,40),xlab="X", ylab="Z, Y, E[Z] & E[Y]",main
="Time series Plot")
points(data$X,data$Y, type='l',col="red")
points(data$X,E_Z,type = 'l',col="green")
points(data$X,E_Y,type='l',col="orange")
legend("topright",col=c("blue","red","green","orange"),legend = c("Z","Y","E[Z]","E[Y]"),lty = c
(1,1))
```