com_stats_lab06_group14

Akshath Srinivas,Samira Goudarzi

2022-12-14

Question 1: Genetic algorithm

Task 1-3

```r
############# Question 1 ##############

#task 1
fitness_func<-function(x){
  f_f<-((x^2)/exp(x))-(2*exp(-(9*sin(x))/(x^2+x+1)))
  return(f_f)
}

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

#task 3
mutate<-function(x){
  m<-x^2 %% 30
  return(m)
}
```

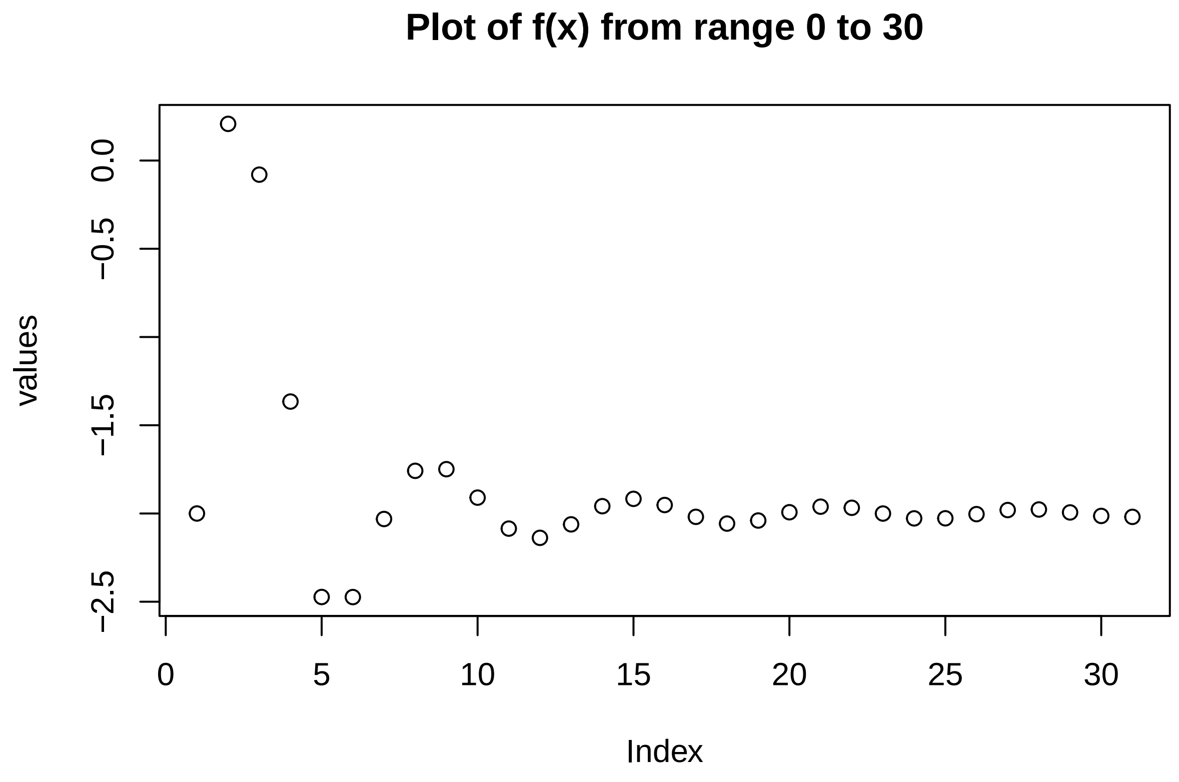
Task 4

we have implemented the function based on parameters **Maximum iterations** and **Mutation rate**.

The plot for the function $f(x)$ ranging from 0 to 30 is shown below and also we can see the maximum value below.

```r
# genetic algorithm
genetic_algo<-function(maxiter,mutprob){
  max_values<-c()
  initial_population<-seq(0,30,5)
  values<-fitness_func(initial_population)
  ini<-fitness_func(c(0:30))
  ini_df<-data.frame(initial=c(0:30),values=ini)
  for(i in 1:maxiter){
    parents<-sample(initial_population,2)
    victim<-order(values)[1]
    kid<-crossover(parents[1],parents[2])
    if(runif(1)<=mutprob){
      kid<-mutate(kid)
    }
    initial_population[victim]<-kid
    values[victim]<-fitness_func(initial_population[victim])
    max_values<-c(max_values,max(values))
  }
  my_df<-data.frame(final_population=initial_population,final_values=values)

  plot<-ggplot(data=ini_df,aes(x=initial,y=values))+
    geom_point() + geom_point(data=my_df,aes(x = final_population,y = final_values),color='blue')+xlab(
  return(list(plot,my_df))
}
```

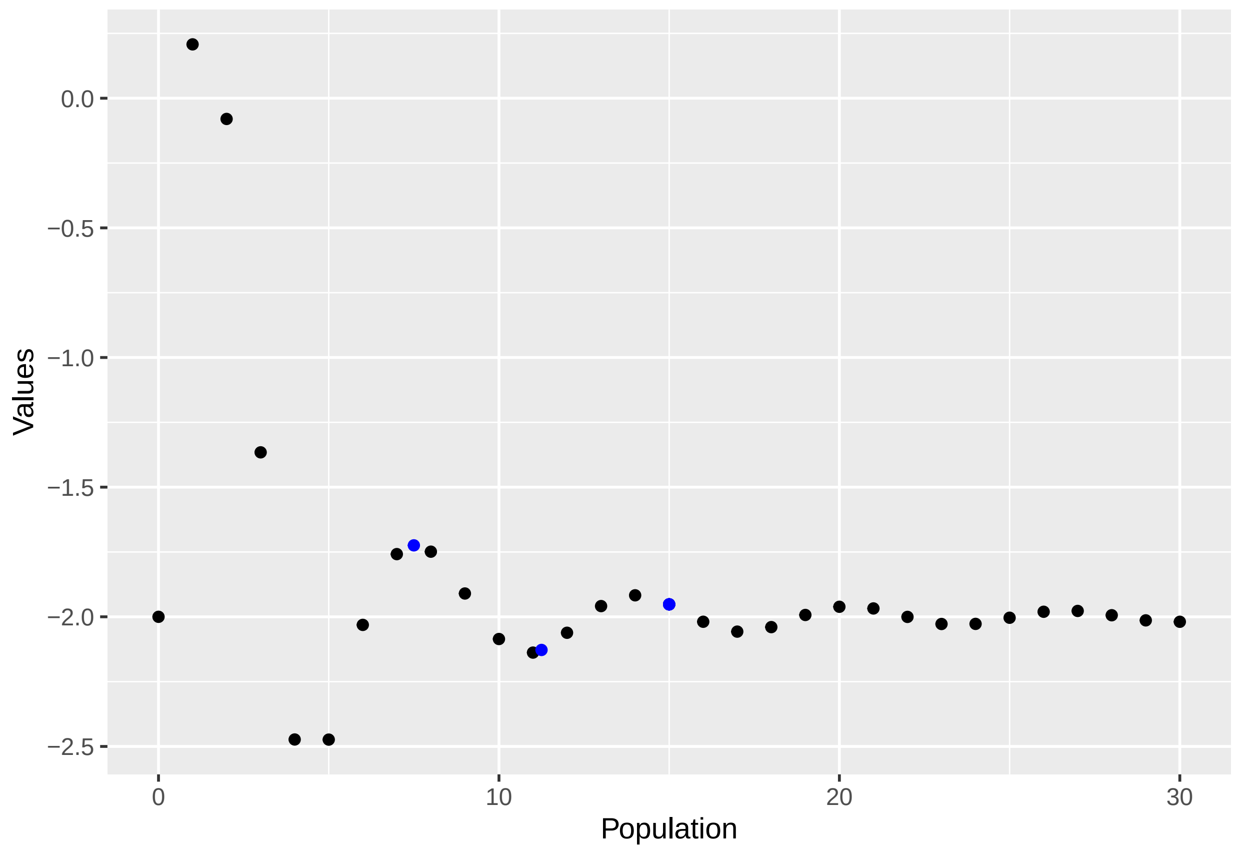


```
## Maximum value is: 0.2076688
```

Task 5

maxiter=10 and mutprob=0.1

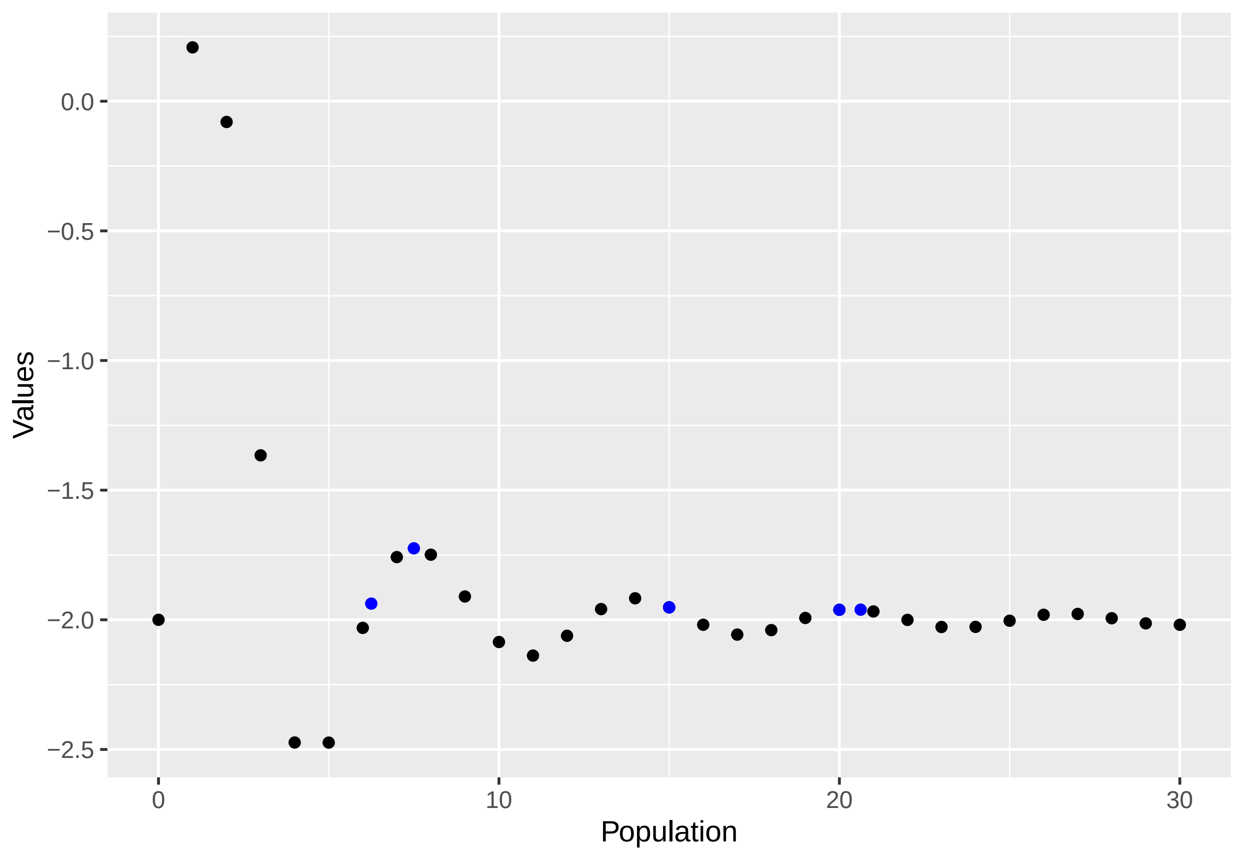
```
## [[1]]
```



```
##
## [[2]]
##   final_population final_values
## 1            15.00    -1.951947
## 2             7.50    -1.724415
## 3            15.00    -1.951947
## 4            15.00    -1.951947
## 5            11.25    -2.127872
## 6            15.00    -1.951947
## 7            15.00    -1.951947
```

maxiter=10 and mutprob=0.5

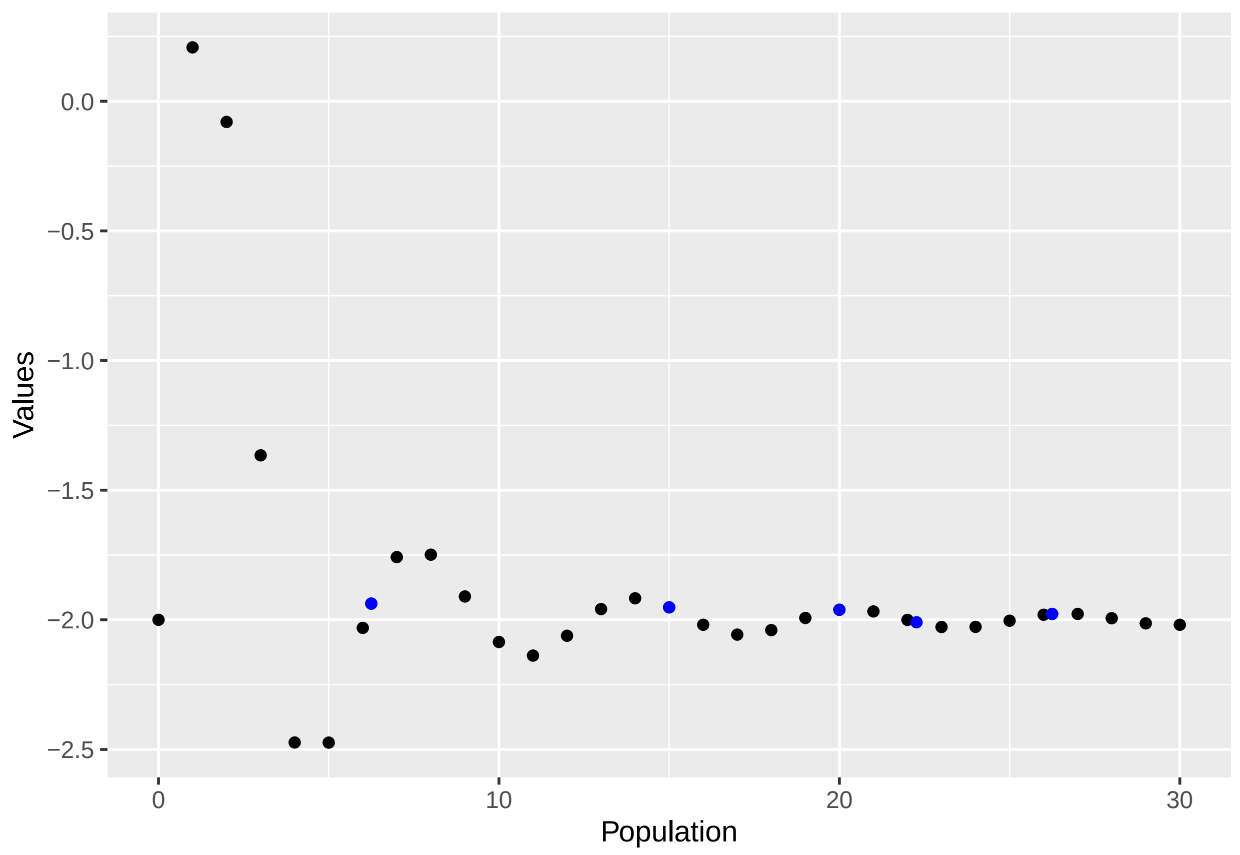
```
## [[1]]
```



```
##
## [[2]]
##   final_population final_values
## 1           15.000    -1.951947
## 2           15.000    -1.951947
## 3           20.625    -1.960959
## 4           15.000    -1.951947
## 5           20.000    -1.961344
## 6            6.250    -1.937529
## 7            7.500    -1.724415
```

maxiter=10 and mutprob=0.9

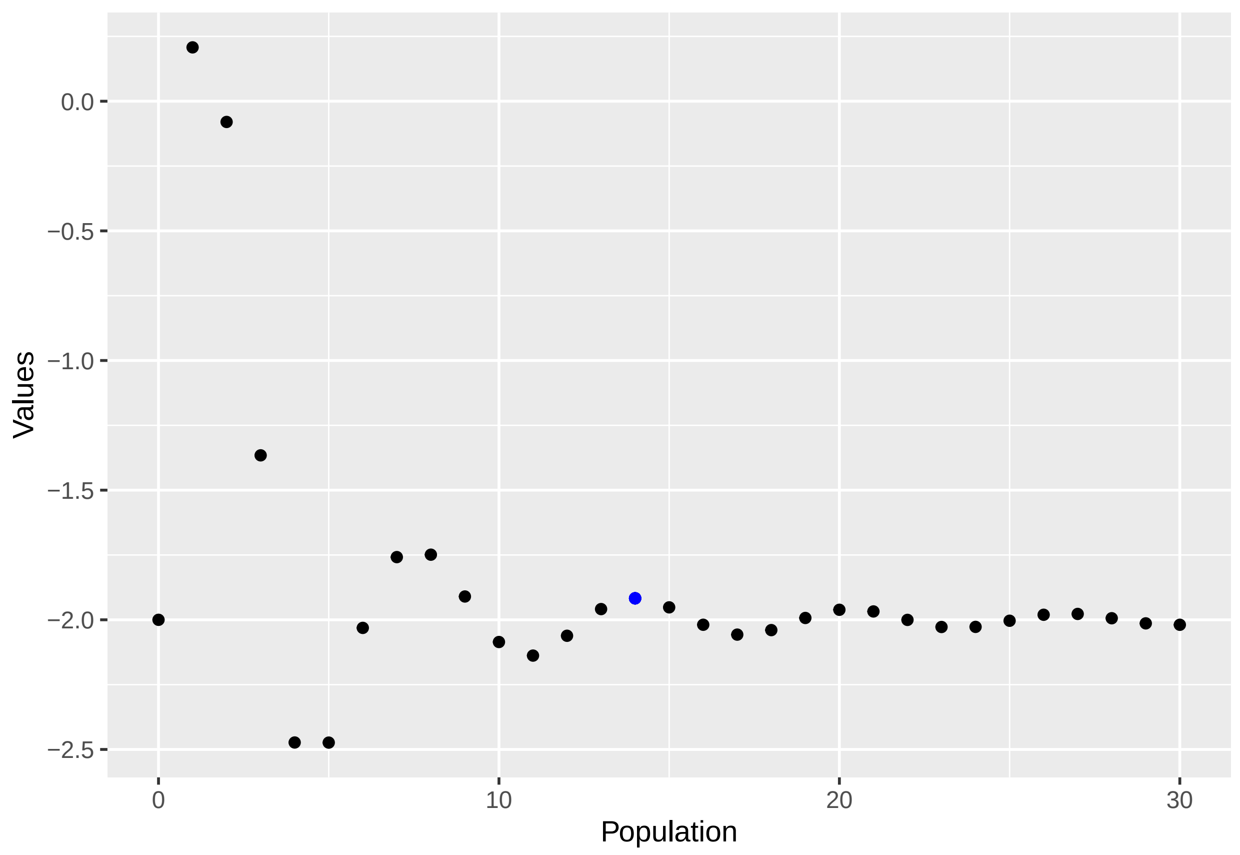
```
## [[1]]
```



```
##
## [[2]]
##   final_population final_values
## 1         22.26562    -2.009422
## 2          6.25000    -1.937529
## 3          6.25000    -1.937529
## 4         15.00000    -1.951947
## 5         20.00000    -1.961344
## 6         26.25000    -1.977539
## 7         26.25000    -1.977539
```

maxiter=100 and mutprob=0.1

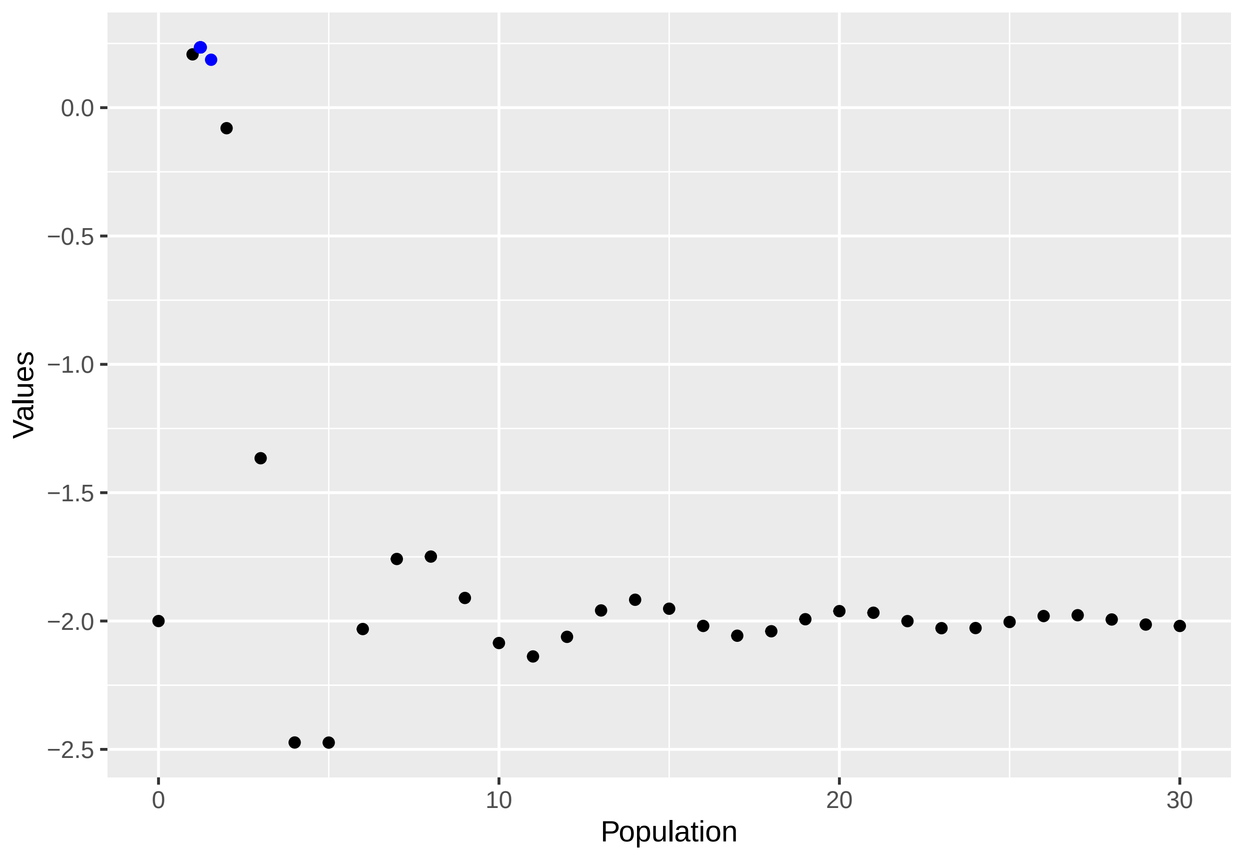
```
## [[1]]
```



```
##
## [[2]]
##   final_population final_values
## 1         13.99979    -1.917091
## 2         13.99979    -1.917091
## 3         13.99979    -1.917091
## 4         13.99979    -1.917091
## 5         13.99979    -1.917091
## 6         13.99979    -1.917091
## 7         13.99979    -1.917091
```

maxiter=100 and mutprob=0.5

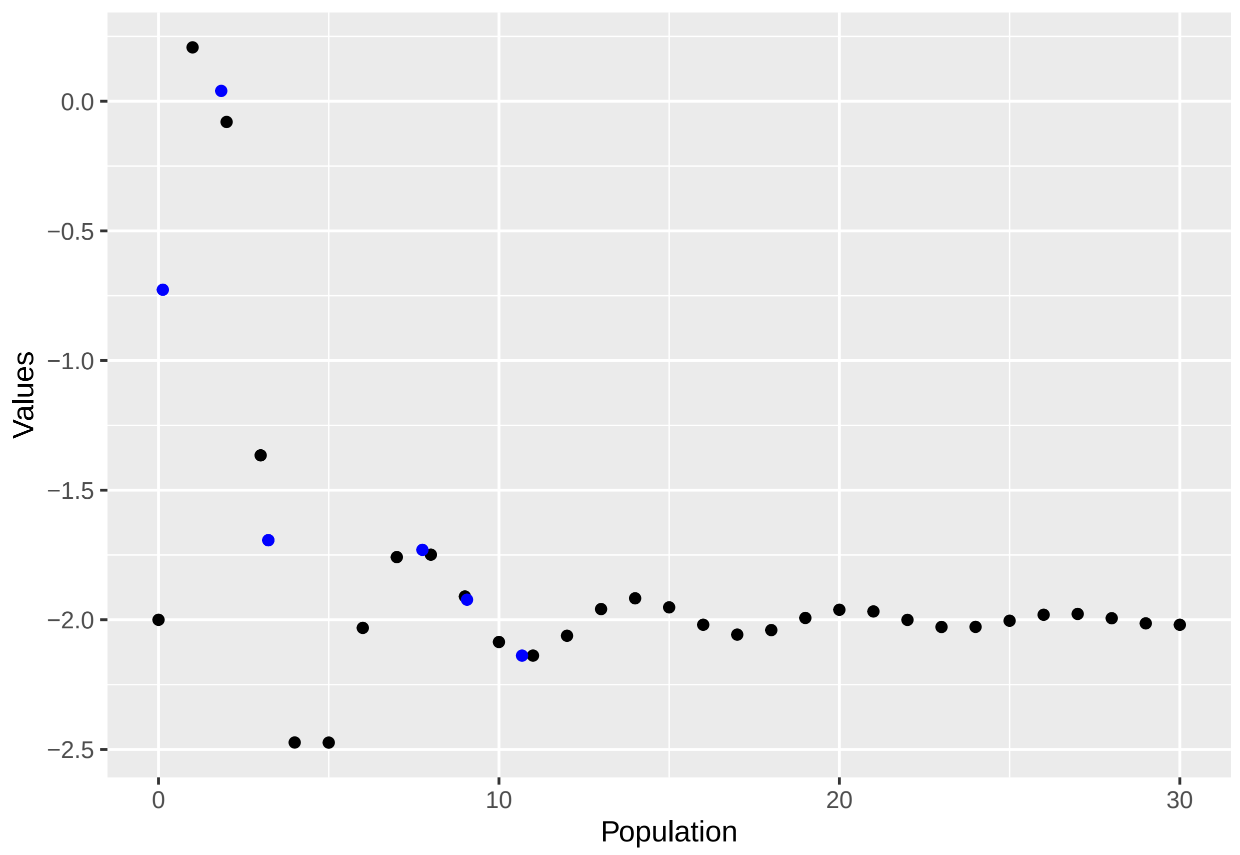
```
## [[1]]
```



```
##
## [[2]]
##   final_population final_values
## 1         1.225894    0.2347663
## 2         1.228471    0.2347967
## 3         1.222730    0.2347201
## 4         1.239368    0.2348530
## 5         1.217575    0.2346238
## 6         1.544504    0.1866480
## 7         1.239368    0.2348530
```

maxiter=100 and mutprob=0.9

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



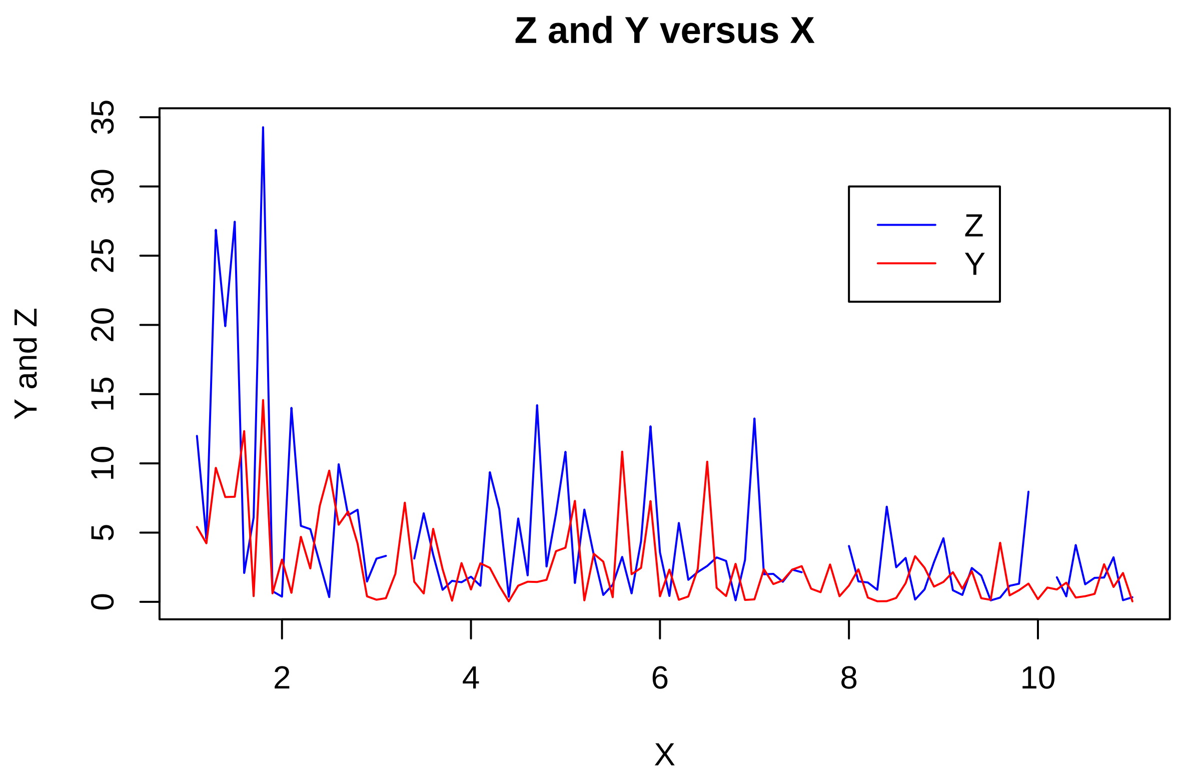
```
##
## [[2]]
##   final_population final_values
## 1        0.1263406  -0.72703062
## 2        3.2259560  -1.69305194
## 3        7.7515375  -1.73024961
## 4        1.8421619   0.03988147
## 5       10.6779642  -2.13814490
## 6        9.0625000  -1.92246571
## 7        3.2259560  -1.69305194
```

Conclusion

We can conclude that as the maximum iterations is increased we are getting the satisfactory solutions. As the mutation probability increases, it increases the probability of searching more areas in search space which may prevent converging to optimum solution. On other side, small probability may result to premature convergence. The best value of mutation probability depends on the problem. In our case, as the mutation rate is increased we are moving towards convergence.

Question 2: EM algorithm

Task 1



We can see from the above plot, both Z and Y has same trend when plotted versus X. We can observe variation of Y and Z is decreasing as the X is increased.

Task 2

Given models.

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

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

Density functions can be written as below (based on exponential distribution).

$$f(Y_i) = \frac{X_i}{\lambda} exp((-X_i/\lambda) \times Y_i)$$
$$f(Z_i) = \frac{X_i}{2\lambda} exp((-X_i/2\lambda) \times Z_i)$$

Likelihood function can be given by.

$$L(\lambda|Y,Z) = \prod_{i=1}^{n} P(Y_i|\lambda) * \prod_{i=1}^{n} P(Z_i|\lambda)$$

$$L(\lambda|Y,Z) = \prod_{i=1}^{n} \frac{X_i}{\lambda} \cdot \exp(-\frac{X_i}{\lambda}Y_i) * \prod_{i=1}^{n} \frac{X_i}{2\lambda} \cdot \exp(-\frac{X_i}{2\lambda}Z_i)$$

Applying log on both sides and solving further.

$$lnL(\lambda|Y,Z) = \sum_{i=1}^{n} \ln(X_i) - n\ln(\lambda) - \sum_{i=1}^{n} \frac{X_i}{\lambda}Y_i + \sum_{i=1}^{n} \ln(X_i) - n\ln(2\lambda) - \sum_{i=1}^{n} \frac{X_i}{2\lambda}Z_i$$

Now to find E step we know.

$$Q(\theta,\theta^k) = E[loglik(\theta|Y,Z)|\theta^k, Y]$$

we can split the Z into two parts, in which one part contains missing values and other part contains non missing values.

$$E[loglik(\lambda|Y,Z)|\lambda^k, Y] = E[\sum_{i=1}^{n} \ln(X_i) - n\ln(\lambda) - \sum_{i=1}^{n} \frac{X_i}{\lambda}Y_i + \sum_{i=1}^{n} \ln(X_i) - n\ln(2\lambda) - \sum_{i=1}^{n} \frac{X_i}{2\lambda}Z_i]$$

$$Q(\lambda,\lambda^k) = \sum_{i=1}^{n} \ln(X_i) - n\ln(\lambda) - \sum_{i=1}^{n} \frac{X_i}{\lambda}Y_i + \sum_{i=1}^{n} \ln(X_i) - n\ln(2\lambda) - \sum_{i=1}^{r} \frac{X_i}{2\lambda}Z_i - \sum_{i=r+1}^{n} \frac{\lambda^k}{\lambda}$$

Next step is to maximize, for that we take derivative with respect to λ and equate it to 0.

M-step:$\lambda^{k+1} = argmax_{\lambda} Q(\lambda,\lambda^k)$

$$-\frac{n}{\lambda} + \sum_{i=1}^{n} \frac{X_i}{\lambda^2}Y_i - \frac{n}{\lambda} + \sum_{i=1}^{r} \frac{X_i}{2\lambda^2}Z_i + \frac{(n-r)\lambda^k}{\lambda^2} = 0$$

where r is length of non missing values in Z and $(n-r)$ is length of missing values in Z.

On further solving we get λ^{k+1} as.

$$\lambda^{k+1} = \frac{1}{2n}(\sum_{i=1}^{n} X_iY_i + \sum_{i=1}^{r} \frac{1}{2}X_iZ_i + (n-r)\lambda^k)$$

We will use this equation in next task to estimate λ.

Task 3

```
##   iterations lamda_previous lamda_current log_likelihood
## 1          1      100.00000      14.26782      -683.9621
## 2          2       14.26782      10.83853      -678.9197
## 3          3       10.83853      10.70136      -679.0048
## 4          4       10.70136      10.69587      -679.0089
## 5          5       10.69587      10.69566      -679.0090
```

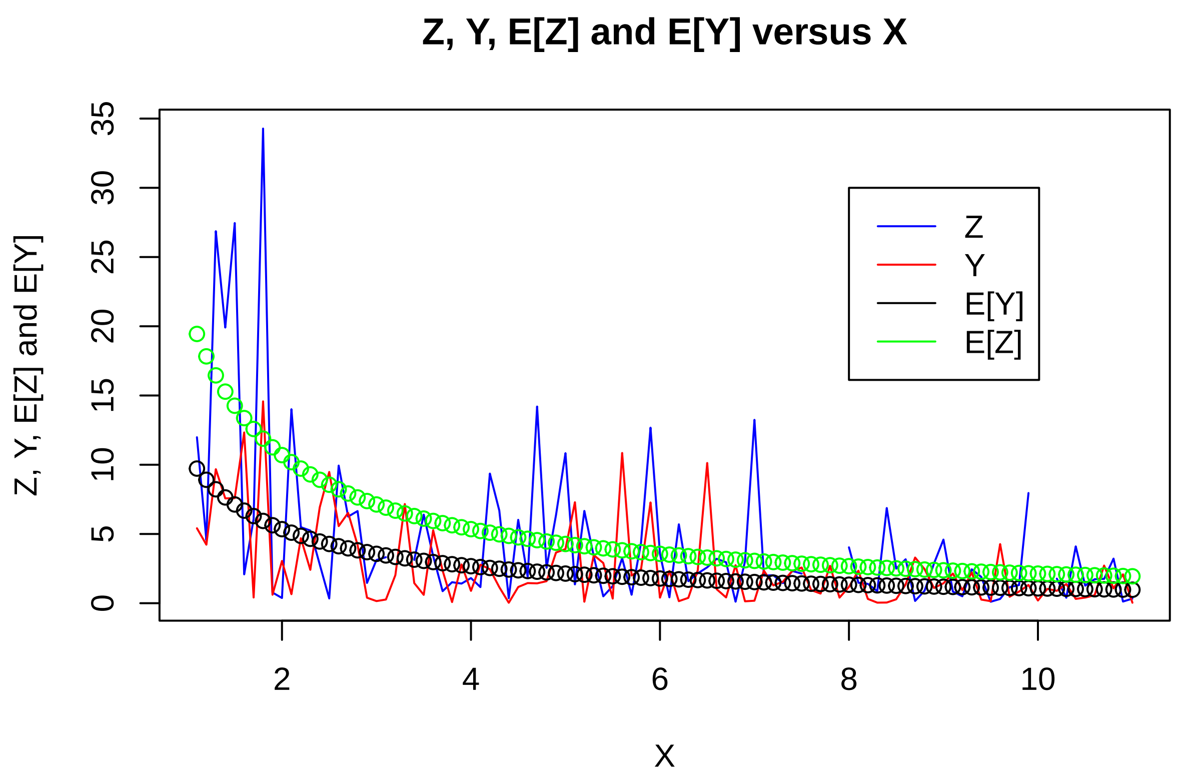
```
## Optimal lambda is : 10.69566
```

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

Task 4

Task is to find out $E[Y]$ and $E[z]$.

Since Y and Z follows exponential distribution, we can find out the expected values by $\frac{\lambda}{X}$ and $\frac{2\lambda}{X}$.



We can conclude that our computed λ is reasonable. The expected values plotted captures the original trend of the data values and also the missing values can be filled or drawn using this distribution.

Appendix

```
knitr::opts_chunk$set(echo = TRUE)
library(ggplot2)

############# Question 1 ##############

#task 1
fitness_func<-function(x){
  f_f<-((x^2)/exp(x))-(2*exp(-(9*sin(x))/(x^2+x+1)))
  return(f_f)
}

#task 2
crossover<-function(x,y){
```

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

#task 3
mutate<-function(x){
  m<-x^2 %% 30
  return(m)
}


# genetic algorithm
genetic_algo<-function(maxiter,mutprob){
  max_values<-c()
  initial_population<-seq(0,30,5)
  values<-fitness_func(initial_population)
  ini<-fitness_func(c(0:30))
  ini_df<-data.frame(initial=c(0:30),values=ini)
  for(i in 1:maxiter){
    parents<-sample(initial_population,2)
    victim<-order(values)[1]
    kid<-crossover(parents[1],parents[2])
    if(runif(1)<=mutprob){
      kid<-mutate(kid)
    }
    initial_population[victim]<-kid
    values[victim]<-fitness_func(initial_population[victim])
    max_values<-c(max_values,max(values))
  }
  my_df<-data.frame(final_population=initial_population,final_values=values)

  plot<-ggplot(data=ini_df,aes(x=initial,y=values))+
    geom_point() + geom_point(data=my_df,aes(x = final_population,y = final_values),color='blue')+xlab(
  return(list(plot,my_df))
}


#plot for function f, range from 0 to 30
ini_pl<-fitness_func(c(0:30))
plot(ini_pl,ylab='values',main='Plot of f(x) from range 0 to 30')
cat('Maximum value is:',max(ini_pl))


genetic_algo(10,0.1)


genetic_algo(10,0.5)


genetic_algo(10,0.9)


genetic_algo(100,0.1)
genetic_algo(100,0.5)


genetic_algo(100,0.9)


############# Question 2 ##############

#plot
phy<-read.csv('physical1.csv')
plot(phy[,1],phy[,3],type='l',xlab='X',ylab='Y and Z',main ='Z and Y versus X',col='blue')
lines(phy[,1],phy[,2],col='red')
legend(x = 8, y = 30,
       legend = c('Z','Y'),
       col = c('blue','red'), lty = c(1,1))


# expectation maximization algorithm
em_algo <- function(data,min_stop_cri,lambda){
  z_obs<-data$Z[!is.na(data$Z)]
  z_missing<-data$Z[is.na(data$Z)]
  x_z_obs<-data$X[which(!is.na(data$Z))]
  n<-nrow(data)
  r<-length(z_obs)
  n_r<-length(z_missing)
  k<-0
  lamda_pre<-0
  lamda_curr<-lambda
  lamda_previous<-c()
  lamda_current<-c()
  K_iterations<-c()
  log_lik<-c()
  while(abs(lamda_curr-lamda_pre)>min_stop_cri){
    lamda_pre<-lamda_curr
    #EM step
    lamda_curr<-(1/(2*n))*(sum(data$X*data$Y)+(1/2)*sum(x_z_obs*z_obs)+
                             n_r*lamda_pre)

    # likelihood calculation
    log_lik_cur<-(2*log(sum(data$X))-(r*log(lamda_curr)+r*log(2*lamda_curr))-
                    ((sum((data$X*data$Y)/lamda_curr))+(sum((x_z_obs*z_obs)/(2*lamda_curr)))))

    log_lik<-c(log_lik,log_lik_cur)
    lamda_previous<-c(lamda_previous,lamda_pre)
    lamda_current<-c(lamda_current,lamda_curr)
    k<-k+1
    K_iterations<-c(K_iterations,k)

  }
  return(data.frame(iterations=K_iterations,lamda_previous=lamda_previous,                lamda_current=
}
em_algo(phy,0.001,100)
```

```
optimal_lambda<-em_algo(phy,0.001,100)[5,3]

cat('Optimal lambda is :',optimal_lambda)
cat('The number of iterations is:', nrow(em_algo(phy,0.001,100)))


#Expected values of Z and Y
expected_y<-optimal_lambda/phy$X
expected_Z<-(2*optimal_lambda)/phy$X

# plot including expected values
plot(phy[,1],phy[,3],type='l',xlab='X',ylab='Z, Y, E[Z] and E[Y]',main ='Z, Y, E[Z] and E[Y] versus X',
lines(phy[,1],phy[,2],col='red')
lines(phy$X,expected_y,type='b')
lines(phy$X,expected_Z,type='b',col='green')
legend(x = 8, y = 30,
       legend = c('Z','Y','E[Y]','E[Z]'),
       col = c('blue','red','black','green'), lty = c(1,1))
```