Computational statistics Lab 06 Report

Karthikeyan Devarajan - Karde799

Question 1: Genetic algorithm

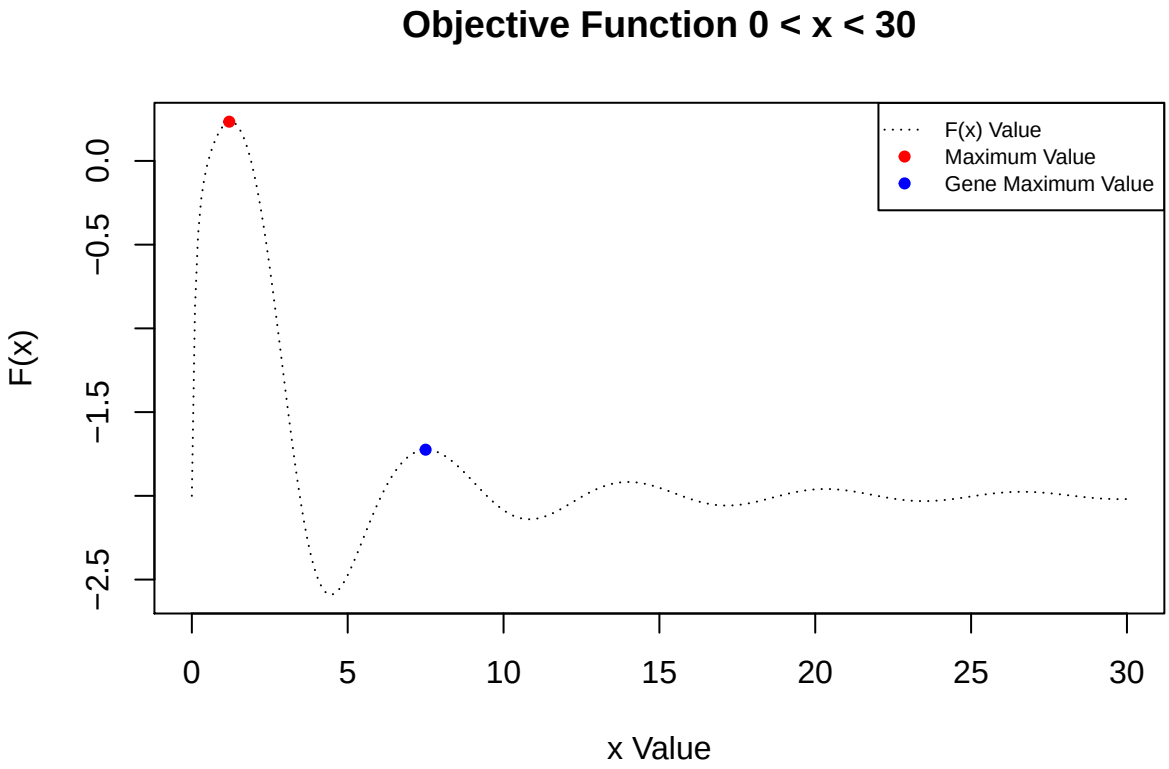
Part 01

Part 02 & Part 03

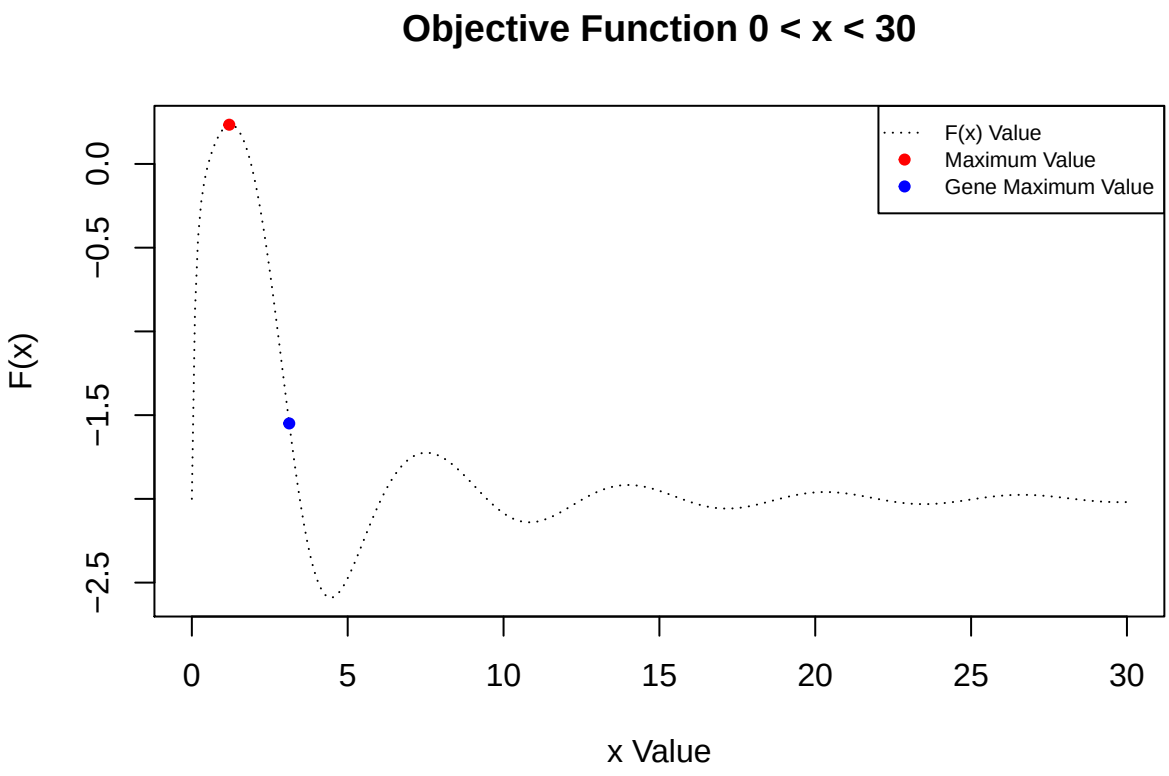
Part 04

Part 05

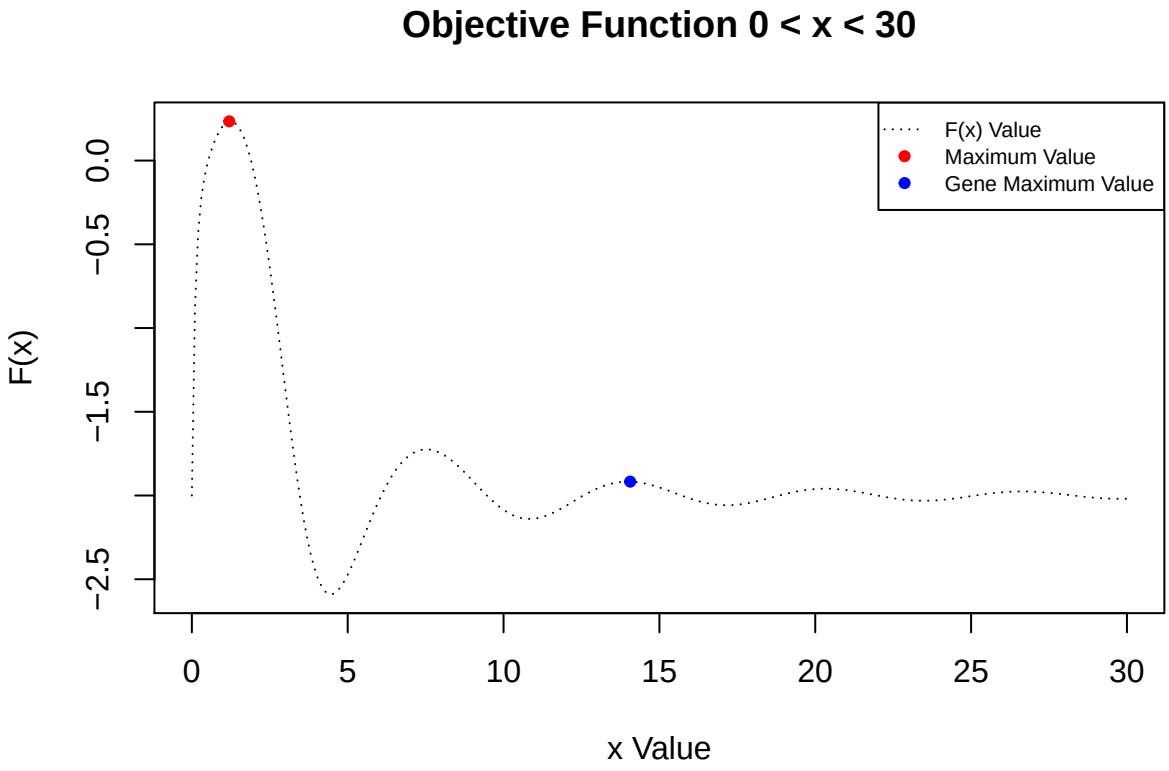
```
## Graphs For  10 and 0.1
## Function Maximum Value is :-  0.2341007
## Genetic Maximum Value is :-  -1.724415
## ** Where 0 < x < 30
```



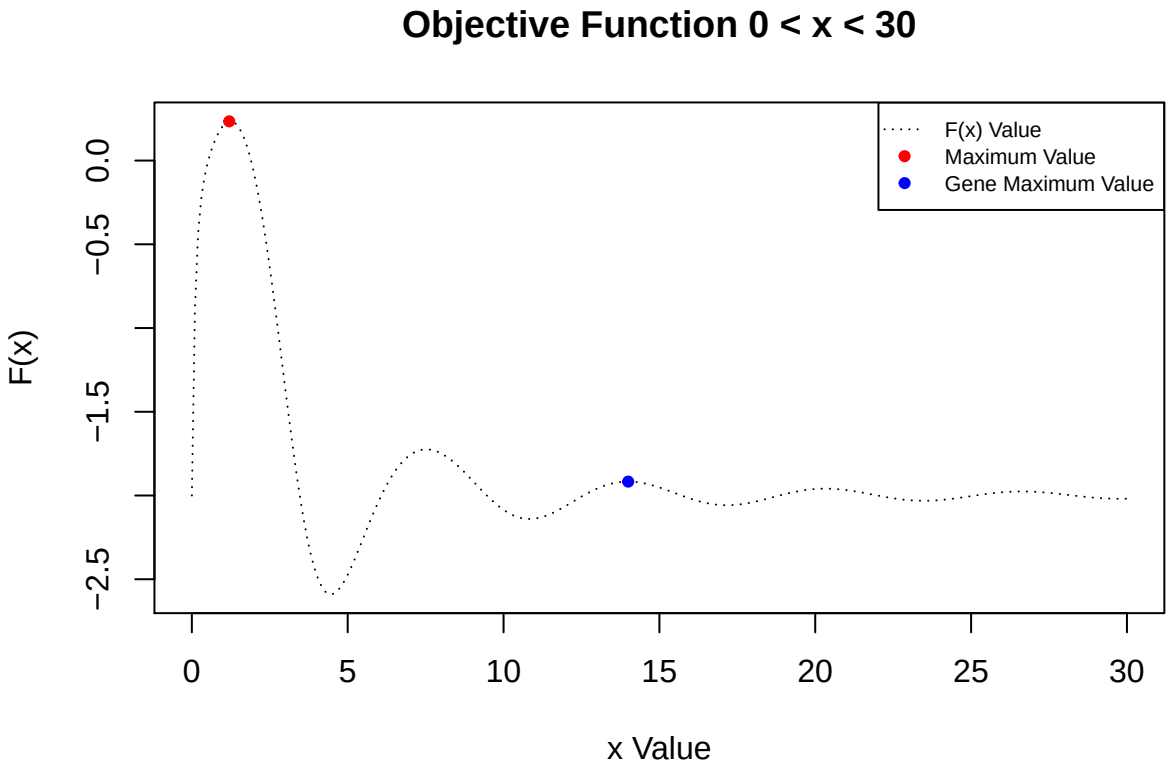
```
## Graphs For  10 and 0.5
## Function Maximum Value is :-  0.2341007
## Genetic Maximum Value is :-  -1.549543
## ** Where 0 < x < 30
```



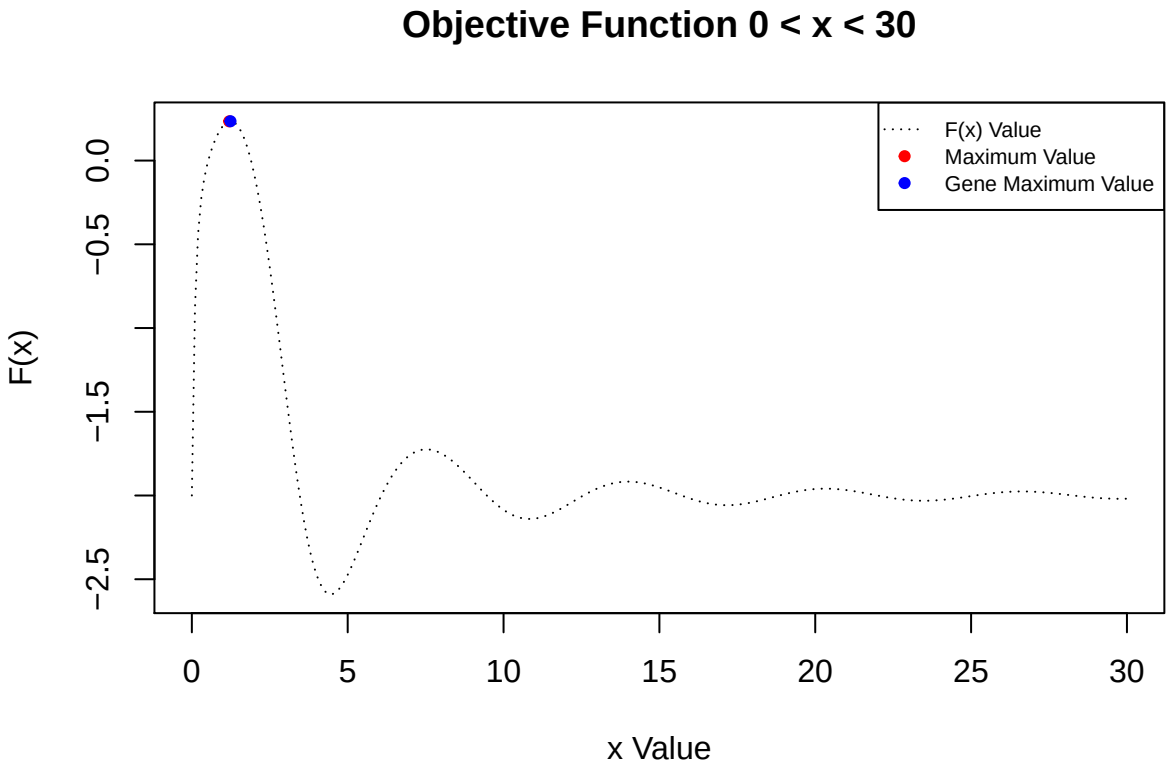

```
## Graphs For  10 and 0.9
## Function Maximum Value is :-  0.2341007
## Genetic Maximum Value is :-  -1.917255
## ** Where 0 < x < 30
```



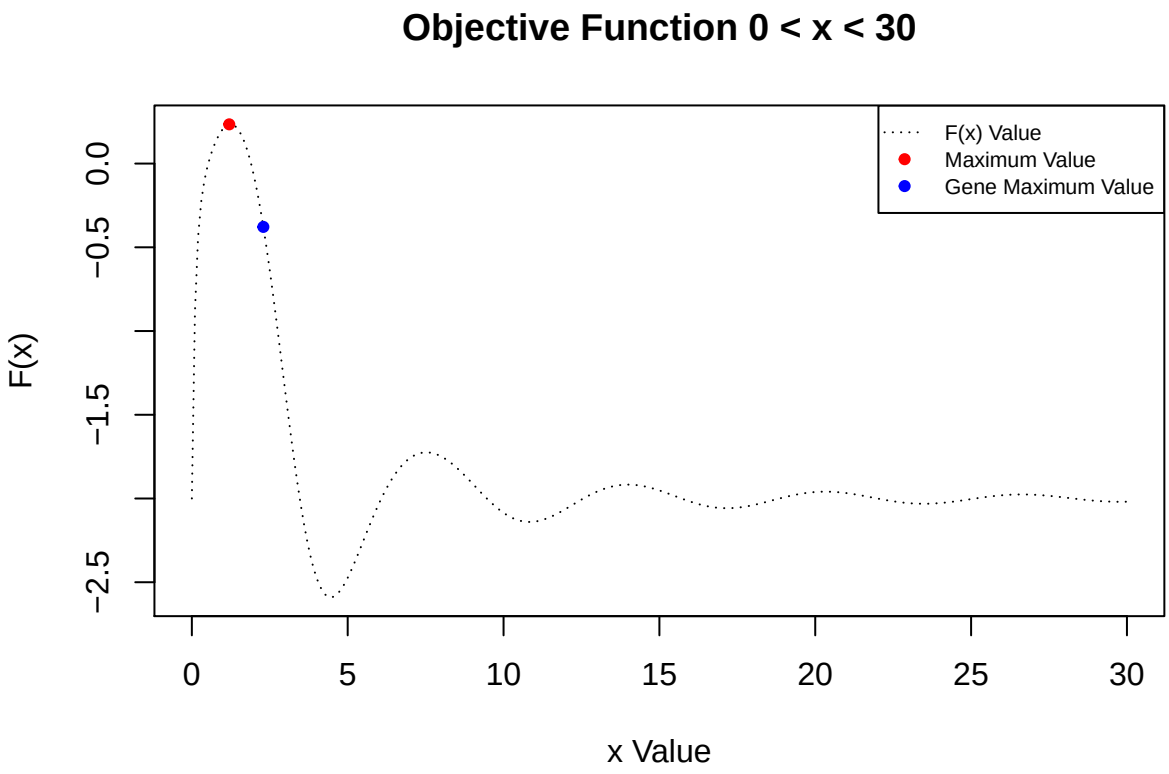

```
## Graphs For  100 and 0.1
## Function Maximum Value is :-  0.2341007
## Genetic Maximum Value is :-  -1.917091
## ** Where 0 < x < 30
```




```
## Graphs For  100 and 0.5
## Function Maximum Value is :-  0.2341007
## Genetic Maximum Value is :-  0.234853
## ** Where 0 < x < 30
```



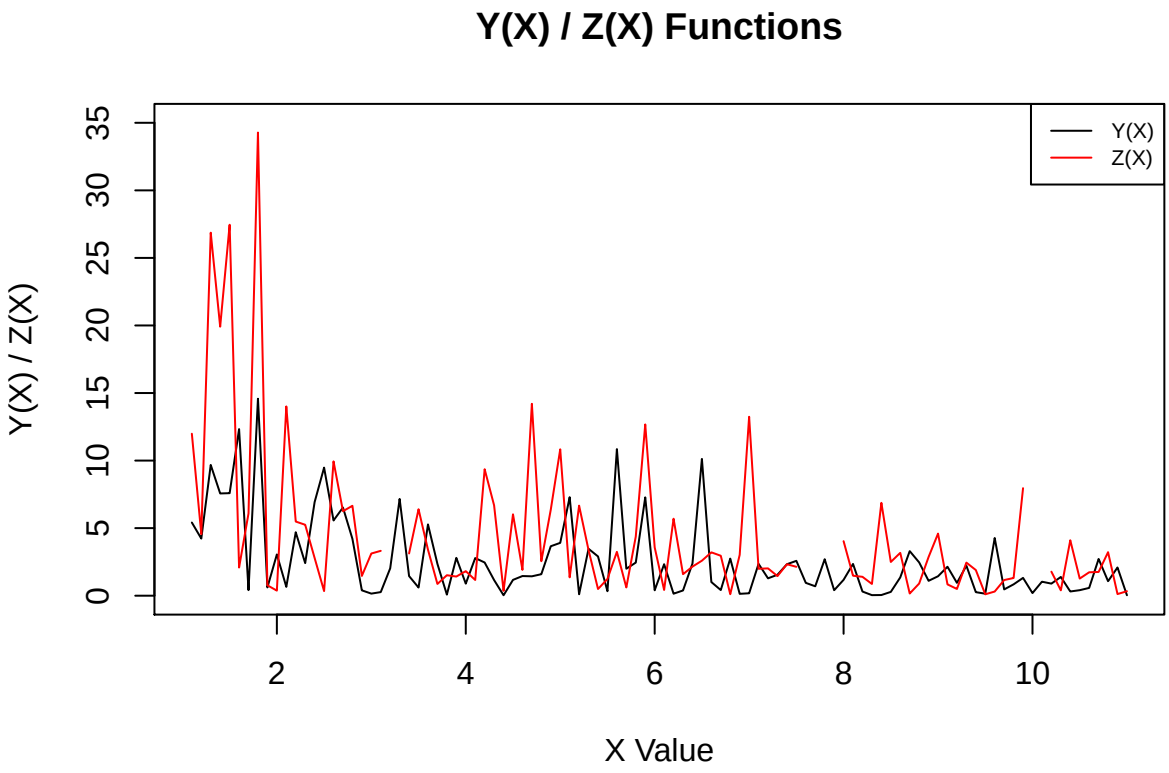

```
## Graphs For  100 and 0.9
## Function Maximum Value is :-  0.2341007
## Genetic Maximum Value is :-  -0.3782303
## ** Where 0 < x < 30
```



Both Mutation Probability and Number of iterations are important. If the Mutation Probability is low, algorithem will converges to the local maxima.And if the the Mutation probability is too high it will take some time or unable to find global maximum. And also it is essential to have decent amount of interations in order to find global maximum.

Question 2: EM algorithm

Part 1




```
## Correlation Between Variables

##            X          Y          Z
## X  1.0000000 -0.4449444 -0.4388769
## Y -0.4449444  1.0000000  0.4876229
## Z -0.4388769  0.4876229  1.0000000

##
##  Variance Between Variables

##           X         Y         Z
## X  8.494835 -3.890104 -7.626895
## Y -3.890104  8.998208  8.721468
## Z -7.626895  8.721468 35.551266
```

It's clear that two process are related to each other since thers is a positive correlation between Y and Z. There is a negative relationship between X and Y and also with X and Z.

Part 2

Since Y and Z are not Independent we can't use Joint distribution of Y(X) and Z(X). So we are considering only Z(X) only.

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

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

$$Lik(Z_i|X_i/2\lambda) = \frac{X_1}{2\lambda} exp(-\frac{X_1 Z_1}{2\lambda}) * \frac{X_2}{2\lambda} exp(-\frac{X_2 Z_2}{2\lambda}) * ...$$

$$Lik(Z_i|X_i/2\lambda) = \frac{\prod_{i=1}^{n} X_i}{(2\lambda)^n} exp(-\frac{X_1 Z_1}{2\lambda} - \frac{X_2 Z_2}{2\lambda} - \frac{X_3 Z_3}{2\lambda} ...)$$

$$Lik(Z_i|X_i/2\lambda) = \frac{\prod_{i=1}^{n} X_i}{(2\lambda)^n} exp(-\frac{\sum_{i=1}^{n} X_i Z_i}{2\lambda})$$

$$LogLik(Z_i|X_i/2\lambda) = ln(\prod_{i=1}^{n} X_i) - ln(2\lambda)^n - \frac{\sum_{i=1}^{n} X_i Z_i}{2\lambda}$$

$$LogLik(Z_i|X_i/2\lambda) = ln(\prod_{i=1}^{n} X_i) - nln(2\lambda) - \frac{\sum_{i=1}^{n} X_i Z_i}{2\lambda}$$

$$E[LogLik(Z_i|X_i/2\lambda)] = ln(\prod_{i=1}^{n} X_i) - nln(2\lambda) - \frac{1}{2\lambda} E[\sum_{i=1}^{n} X_i Z_i]$$

Observed Data $P = y_1, y_2, ..., y_r$

Unobserved Data $Q = y_{r+1}, u_{r+2}, ...y_n$

$$E[ll(Z_i|X_i/2\lambda)] = ln(\prod_{i=1}^{n} X_i) - nln(2\lambda) - \frac{1}{2\lambda} \sum_{i=1}^{r} X_i Z_i - \frac{1}{2\lambda} \sum_{i=r+1}^{n} E[X_i Z_i]$$

$$E[ll(Z_i|X_i/2\lambda)] = ln(\prod_{i=1}^{n} X_i) - nln(2\lambda) - \frac{1}{2\lambda} \sum_{i=1}^{r} X_i Z_i - \frac{1}{2\lambda} \sum_{i=r+1}^{n} \frac{2\lambda_k}{X_i} X_i$$

$$E[ll(Z_i|X_i/2\lambda)] = ln(\prod_{i=1}^{n} X_i) - nln(2\lambda) - \frac{1}{2\lambda} \sum_{i=1}^{r} X_i Z_i - \frac{1}{\lambda}(n-r)\lambda_k$$

$$Q(\theta, \theta_k) = ln(\prod_{i=1}^{n} X_i) - nln(2\lambda) - \frac{1}{2\lambda} \sum_{i=1}^{r} X_i Z_i - \frac{(n-r)\lambda_k}{\lambda}$$

$$Q(\theta, \theta_k)'_\lambda = -\frac{n*2}{2\lambda} + \frac{1}{2\lambda^2} \sum_{i=1}^{r} X_i Z_i + \frac{(n-r)\lambda_k}{\lambda^2}$$

Setting the equation to 0;

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

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

$$\lambda = \frac{1}{2}[\frac{\sum_{i=1}^{r} X_i Z_i}{2} + (n-r)\lambda_k]$$

$$\lambda = \frac{1}{2}[\frac{\sum_{i\in k} X_i Z_i}{2} + (n-r)\lambda_k]$$

K For the observed values.

Part 03

```
## Using Min Logliklihood change  < 0.001
```

```
## [1]  100.0000 -336.3827
## [1]    2.00000   16.83481 -226.20897
## [1]    3.0000   10.1816 -214.2381
## [1]    4.000000    9.649342 -214.084810
## [1]    5.000000    9.606761 -214.083755
## [1]    6.000000    9.603355 -214.083748
```

```
## Optimal Lambda is :-  9.603355
```

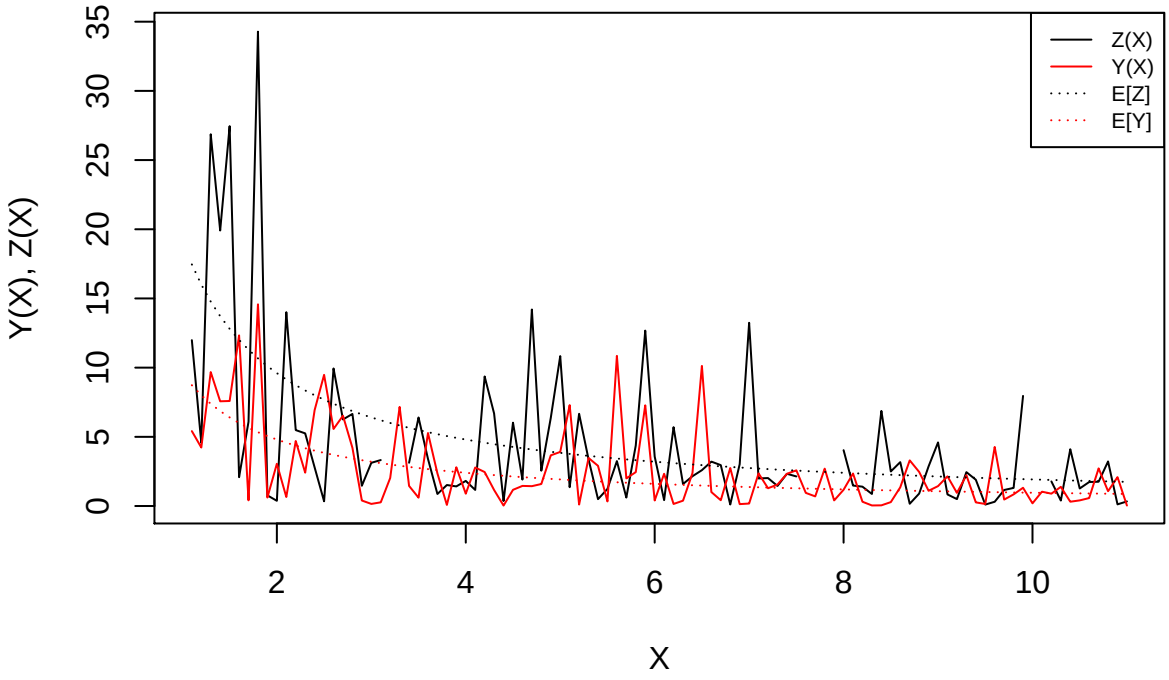
```
##
##  Using Min Lambda change  < 0.001
```

```
## [1] 100   0
## [1]  2.00000  0.00000 16.83481
## [1]  3.000000 16.834814  8.834814
## [1]  4.000000  8.834814 10.181599
## [1]  5.000000 10.181599  9.541599
## [1] 6.000000 9.541599 9.649342
## [1] 7.000000 9.649342 9.598142
## [1] 8.000000 9.598142 9.606761
## [1] 9.000000 9.606761 9.602665
## [1] 10.000000  9.602665  9.603355
```

```
## Optimal Lambda is :-  9.603355
```

It will take 10 iteration to generate optimal λ. Optimal value is $\lambda = 9.603355$.

Part 04



In general λ value seems to be reasonable. Since it follows the same trend of the Y and Z. But it's clear that there are some outliers in the generated Y and Z value. Therfore estimation accuracy seems to be not accurate.

APPENDIX

```
knitr::opts_chunk$set(echo = TRUE)
RNGversion(min(as.character(getRversion()),"3.6.2"))
set.seed(12345, kind = "Mersenne-Twister", normal.kind = "Inversion")
# Part 1
objectiveFunc = function(x) {
  return(((x**2) / exp(x)) -
           (2 * exp(-(9 * sin(x)) / (x**2 + x + 1))))
}
# Part 2

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

# Part 3

mutate = function(x) {
  return(x**2 %% 30)
}
```

```r
# Part 4

geneticsFunc = function(maxiter, mutprob) {

  X_sample = seq(from = 0, to = 30, by = 0.1)
  Y_sample = objectiveFunc(X_sample)

  #genInitialPlot(X_sample,Y_sample)


  genePop = seq(from = 0, to = 30, by = 5)
  values = objectiveFunc(genePop)

  currentBestValue = NULL
  currentBest = NULL

  for (index in 1:maxiter) {
    parentIndexes = sample(length(genePop), 2)

    parents = genePop[parentIndexes]
    parentsValues = values[parentIndexes]

    victimIndex = order(values)[1] # Get Lowest value Index
    victim = genePop[victimIndex] # Get Victim Value

    kid = crossover(parents[1], parents[2]) # Generate Kid

    # Mutation Stage
    rProb = runif(1, 0, 1)
    if (rProb < mutprob) {
      kid = mutate(kid) # Mutate Kid value
    }
    kidValue = objectiveFunc(kid) # Get Kid Value

    # Replace Kid with victim
    genePop[victimIndex] = kid
    values[victimIndex] = kidValue

    # Update Best value
    if (is.null(currentBestValue)) {
      currentBestValue = max(values)
      currentBest = genePop[which(max(values) == values)][1]
    } else if (currentBestValue < max(values)) {
      currentBestValue = max(values)
      currentBest = genePop[which(max(values) == values)][1]
    }

  }

  genFinalPlot(X_sample,Y_sample,currentBest,currentBestValue)
}

genInitialPlot = function(X_data,Y_data) {
```

```
  maxYIndex = which(Y_data == max(Y_data))

  maxValue_Y = Y_data[maxYIndex]
  maxValue_X = X_data[maxYIndex]

  cat('Function Maximum Value is :- ', maxValue_Y, '\n')
  cat('** Where 0 < x < 30', '\n')

  plot(x = X_data,
       y = Y_data,
       lty = 3,
       cex = 0.3,
       type = 'l',
       col = 'black',
       xlab = 'x Value',
       ylab = 'F(x)',
       main = 'Objective Function 0 < x < 30'
  )

  points(x = maxValue_X,
         y = maxValue_Y,
         pch = 19,
         cex = 0.7,
         col = 'red')

  legend("topright",
         legend = c("F(x) Value","Maximum Value"),
         col = c("black","red"),
         pch = c(NA,19),
         lty = c(3,NA),
         cex = 0.7
  )
}

genFinalPlot = function(X_data,Y_data,maxPoint_X,maxPoint_Y) {
  maxYIndex = which(Y_data == max(Y_data))

  maxValue_Y = Y_data[maxYIndex]
  maxValue_X = X_data[maxYIndex]

  cat('Function Maximum Value is :- ', maxValue_Y, '\n')
  cat('Genetic Maximum Value is :- ', maxPoint_Y, '\n')
  cat('** Where 0 < x < 30', '\n')

  plot(x = X_data,
       y = Y_data,
       lty = 3,
       cex = 0.3,
       type = 'l',
       col = 'black',
       xlab = 'x Value',
       ylab = 'F(x)',
       main = 'Objective Function 0 < x < 30'
```

```
  )

  points(x = maxValue_X,
         y = maxValue_Y,
         pch = 19,
         cex = 0.7,
         col = 'red')

  points(x = maxPoint_X,
         y = maxPoint_Y,
         pch = 19,
         cex = 0.7,
         col = 'blue')

  legend("topright",
         legend = c("F(x) Value","Maximum Value", "Gene Maximum Value"),
         col = c("black","red","blue"),
         pch = c(NA,19,19),
         lty = c(3,NA,NA),
         cex = 0.7
  )
}

generateGenetics = function(maxIte,muProb) {
  for (mIter in maxIte) {
    for (mProb in muProb) {
      cat('Graphs For ', mIter , 'and', mProb , '\n')
      geneticsFunc(mIter,mProb)
    }
  }
}

generateGenetics(c(10,100), c(0.1,0.5,0.9))
# Part 01

physicalData = read.csv(file.choose())

plotInitialPhyData = function() {
  plot(x = physicalData$X,
       y = physicalData$Y,
       lty = 1,
       cex = 0.7,
       type = 'l',
       col = 'black',
       ylim = c(0,35),
       xlab = 'X Value',
       ylab = 'Y(X) / Z(X)',
       main = 'Y(X) / Z(X) Functions'
  )

  lines(x = physicalData$X,
        y = physicalData$Z,
        lty = 1,
```

```
        cex = 0.7,
        col = 'red')

  legend("topright",
         legend = c("Y(X)","Z(X)"),
         col = c("black","red"),
         lty = 1,
         cex = 0.7
  )

}

plotInitialPhyData()

cat('Correlation Between Variables','\n')
cor(physicalData, use="complete.obs")

cat('\n','Variance Between Variables','\n')
var(physicalData, na.rm = TRUE)
# Part 03

floglik = function(z, x, lambda, n) {
  return(log(prod(x)) -
           (n * log(2 * lambda)) -
           ((1 / (2 * lambda)) * sum(x * z))
  )
}

EM.Norm = function(Z,X,eps,kmax,initial_k) {

  obsIndex = which(!is.na(Z))
  missIndex = which(is.na(Z))

  Zobs = Z[obsIndex]
  Zmiss = Z[missIndex]
  Xobs = X[obsIndex]
  Xmiss = X[missIndex]

  n <- length(c(Zobs, Zmiss))
  r <- length(Zobs)

  k = 1
  lambda_k = initial_k

  llvalprev = floglik(Zobs,Xobs,lambda_k,r)
  llvalcurr = llvalprev + 10 + 100 * eps
  print(c(lambda_k,llvalcurr))

  while ((abs(llvalprev - llvalcurr) > eps) && (k < (kmax + 1))) {
    llvalprev = llvalcurr

    ## E-step - Derive Q(theta,theta_k)
    lambda = (1 / n) * ((sum(Xobs * Zobs) / 2) + ((n - r) * lambda_k))
```

```
    ## M-step
    lambda_k = lambda

    ## Compute log-likelihood
    llvalcurr = floglik(Zobs,Xobs,lambda_k,r)
    k = k + 1

    print(c(k,lambda_k,llvalcurr))
  }
}

EM.NormMod = function(Z,X,eps,kmax,initial_k) {

  obsIndex = which(!is.na(Z))
  missIndex = which(is.na(Z))

  Zobs = Z[obsIndex]
  Zmiss = Z[missIndex]
  Xobs = X[obsIndex]
  Xmiss = X[missIndex]

  n <- length(c(Zobs, Zmiss))
  r <- length(Zobs)

  k = 1
  lambda_k = initial_k
  lambda_c = 0

  print(c(lambda_k,lambda_c))

  while ((abs(lambda_k - lambda_c) > eps) && (k < (kmax + 1))) {
    #llvalprev = llvalcurr

    ## E-step - Derive Q(theta,theta_k)
    temp = lambda_c
    lambda_c = (1 / n) * ((sum(Xobs * Zobs) / 2) + ((n - r) * lambda_k))

    ## M-step
    lambda_k = temp

    ## Compute log-likelihood
    #llvalcurr = floglik(Zobs,Xobs,lambda_k,r)
    k = k + 1

    print(c(k,lambda_k,lambda_c))
  }
}

testSam = rexp(20,100)
testSam[16:20] = NA


cat('Using Min Logliklihood change  < 0.001','\n')
```

```r
EM.Norm(physicalData$Z,physicalData$X,0.001,50,100)
cat('Optimal Lambda is :- ', 9.603355, '\n')

cat('\n','Using Min Lambda change  < 0.001','\n')
EM.NormMod(physicalData$Z,physicalData$X,0.001,50,100)
cat('Optimal Lambda is :- ', 9.603355, '\n')
# Part 04
plot(x = physicalData$X,
     y = physicalData$Z,
     type = "l",
     col = "black",
     xlab = "X",
     ylab = "Y(X), Z(X)"
)
lines(x = physicalData$X,
      y = physicalData$Y,
      col = "red"
)
lines(x = physicalData$X,
      y = 9.603355/physicalData$X,
      col = "red",
      lty = 3
)
lines(x = physicalData$X,
      y = (2*9.603355)/physicalData$X,
      col = "black",
      lty = 3
)
legend("topright",
       legend = c("Z(X)","Y(X)","E[Z]","E[Y]"),
       col = c("black","red","black","red"),
       lty = c(1,1,3,3),
       cex = 0.7
)
```