Group 12 Lab 6

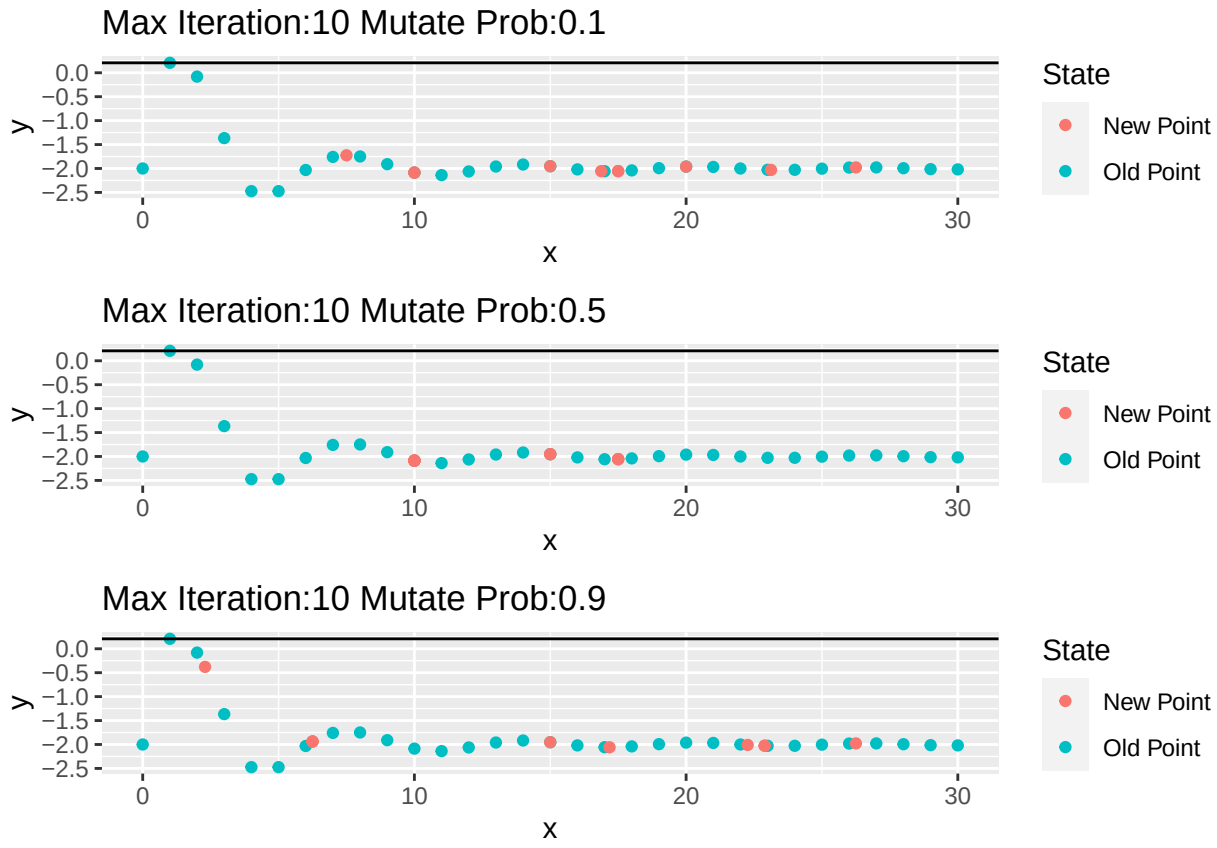
varsi146, aswma317

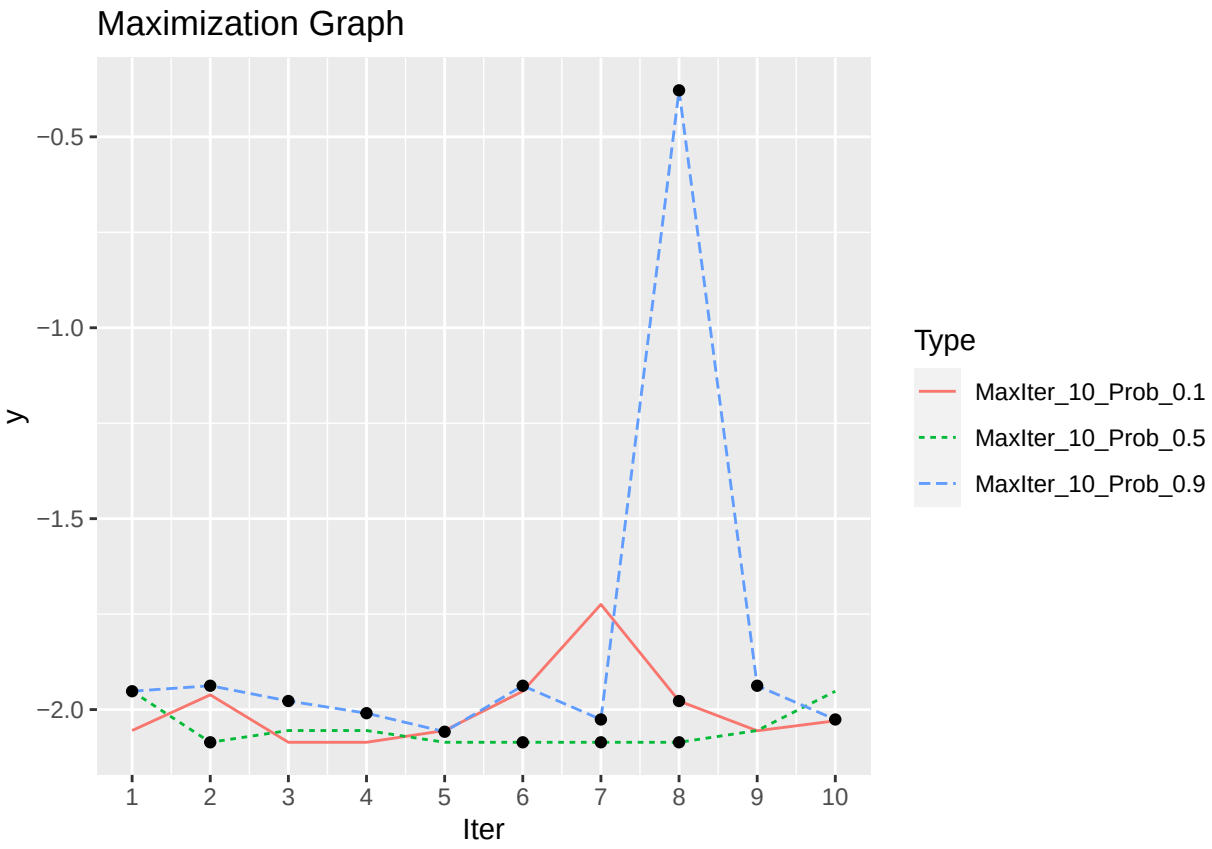
2022-12-19

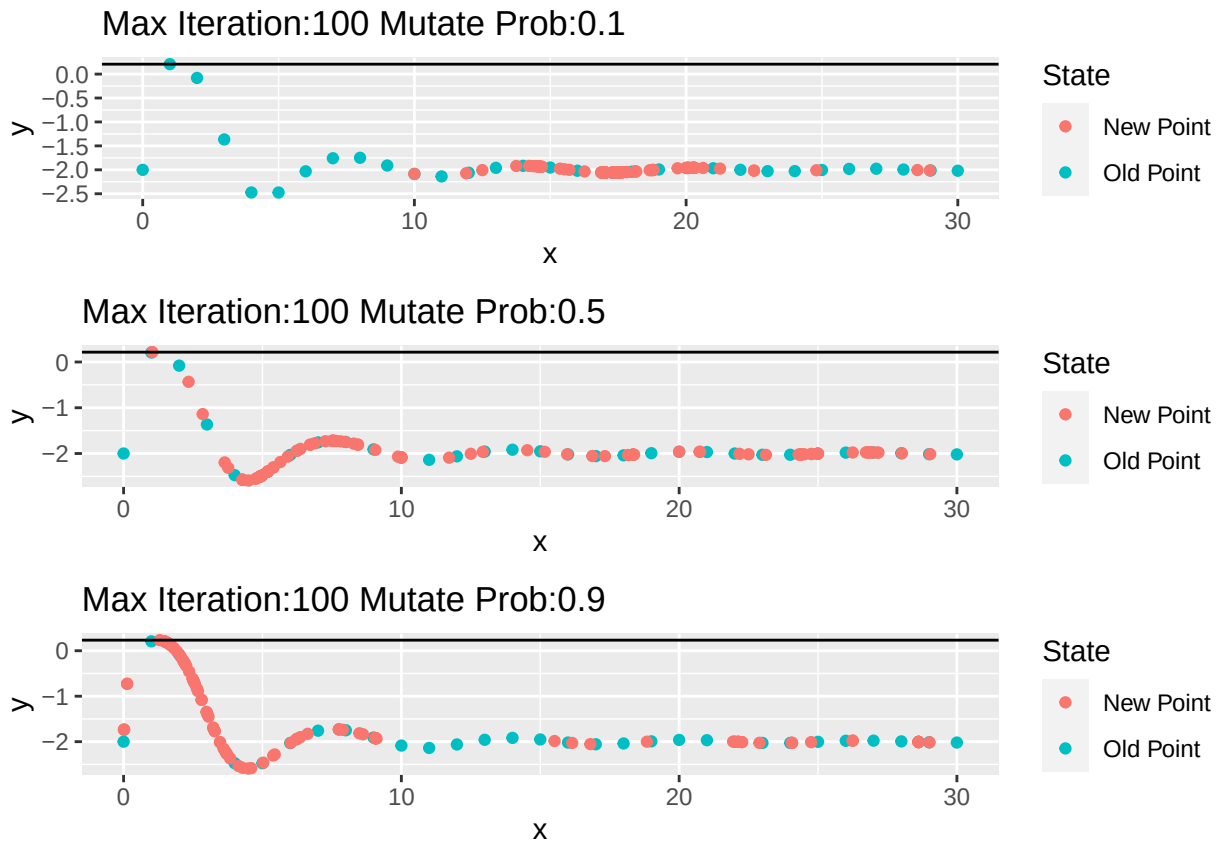
Question 1 : Genetic Algorithm

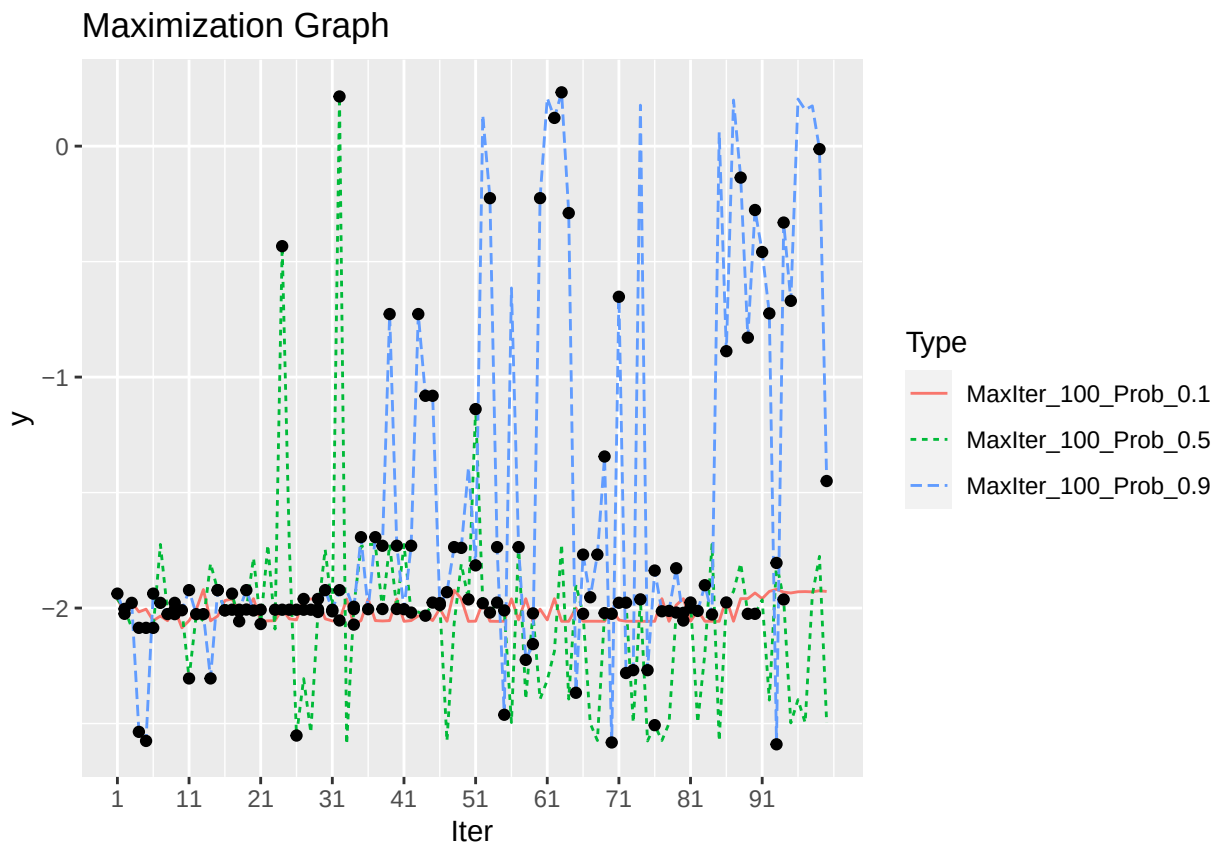
Sub Question 1:

Run your code with different combinations of maxiter= 10; 100 and mutprob= 0.1, 0.5,0.9 and observe the initial population and final population. Conclusions?




Maximization Graph
−0.5
−1.0
−1.5
−2.0
y
1
2
3
4
5
6
7
8
9
10
Iter
Type
MaxIter_10_Prob_0.1
MaxIter_10_Prob_0.5
MaxIter_10_Prob_0.9


Max Iteration:100 Mutate Prob:0.1
0.0
−0.5
−1.0
−1.5
−2.0
−2.5
y
0
10
20
30
x
State
New Point
Old Point
Max Iteration:100 Mutate Prob:0.5
0
−1
−2
y
0
10
20
30
x
State
New Point
Old Point
Max Iteration:100 Mutate Prob:0.9
0
−1
−2
y
0
10
20
30
x
State
New Point
Old Point



As seen in the above graphs, for maximum iteration of 10, we get samples of high y value only with mutation probability of 0.9/0.5, and it may not even be the maximum value.

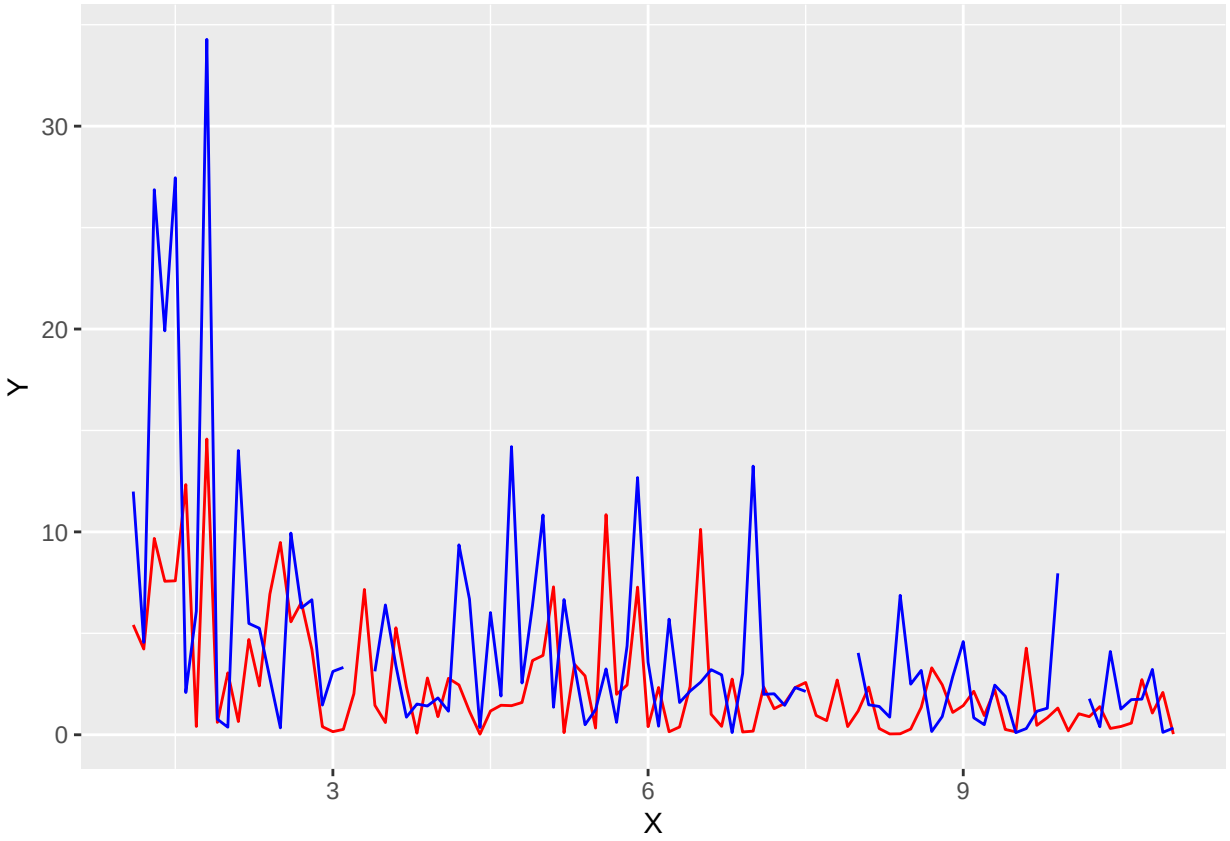
It is also evident from the Maximization graph that the mutation causes the variance in sample values. If there are no mutation, the sample values remain in similar ranges. The mutation causes the samples to be created outside the normal range.

Question 2 : EM Algorithm

Sub Question 1:

Below is time series plot describing the dependence of Z and Y versus X. Based on the plot, we can say that the two processes seem to be related to each other since their trends are similar and that they are both moving in the same direction and that they seem to spike at similar points with respect to X.

From the plot we can say that the variation of the response variables is stabilizing/reducing as the values of X increases.



Sub Question 2:

In our problem statement, we have two variables Y and Z distributed exponentially, and can be represented as the following models:

$$Y_i \sim exp(\frac{X_i}{\lambda}) \tag{1}$$

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

The density function for equation 1:

$$f(Y_i) = \frac{X_i}{\lambda} \cdot exp(Y_i \cdot \frac{X_i}{\lambda}) \tag{3}$$

The density function for equation 2:

$$f(Z_i) = \frac{X_i}{2\lambda} \cdot exp(Z_i \cdot \frac{X_i}{2\lambda}) \tag{4}$$

The likelihood function for equation 3 is written as:

$$L(Y_i|\lambda) = \prod_{i=1}^{n} \frac{X_i}{\lambda} \cdot exp(Y_i \cdot \frac{X_i}{\lambda}) \tag{5}$$

The likelihood function for equation 4 is written as:

$$L(Z_i|\lambda) = \prod_{i=1}^{n} \frac{X_i}{2\lambda} \cdot exp(Z_i \cdot \frac{X_i}{2\lambda}) \tag{6}$$

Applying log to the respective likelihood functions and combining them, we get:

$$log[L(Y_i, Z_i|X_i, \lambda)] = 2 \cdot log \sum_{i=1}^{n} X_i - [n \cdot log(\lambda) + n \cdot log(2 \cdot \lambda)] - \left[\sum_{i=1}^{n} \frac{X_i \cdot Y_i}{\lambda} + \sum_{i=1}^{n} \frac{X_i \cdot Z_i}{2 \cdot \lambda}\right] \tag{7}$$

Equation 7 above is the joint log-likelihood for which we find the expected value in the Expectation step of the EM Algorithm.

Now, we define $Q(\lambda, \lambda^{(k)})$ to be the expectation of the joint log-likelihood for the complete data conditional on the observed data.

$$Q(\lambda, \lambda^{(k)}) = log[L(Y_i, Z_i|X_i, \lambda^{(k)})] \tag{8}$$

Since, Z has non-missing and missing values, we decompose the Z term in equation 7 into two terms(Z_{NM}&Z_M), one for non-missing and missing values respectively and we write it as:

$$Q(\lambda, \lambda^{(k)}) = E[log[L(Y_i, Z_M, Z_{NM}|X_i, \lambda^{(k)})]] \tag{9}$$

$$Q(\lambda, \lambda^{(k)}) = E\left[2 \cdot log \sum_{i=1}^{n} X_i - [n \cdot log(\lambda) + n \cdot log(2 \cdot \lambda)] - \left[\sum_{i=1}^{n} \frac{X_i \cdot Y_i}{\lambda} + \sum_{i=1}^{r} \frac{X_i \cdot Z_{NM}}{2 \cdot \lambda} + \sum_{i=r+1}^{n} \frac{X_i \cdot Z_M}{2 \cdot \lambda}\right]\right] \tag{10}$$

The expected value of an exponential distribution, $(\lambda exp(-\lambda x))$ is known as as its mean, that is $\frac{1}{\lambda}$. Therefore, the expected value of the missing values of Z in the above equation can be written as:

Expectation Step

$$Q(\lambda, \lambda^{(k)}) = 2 \cdot log \sum_{i=1}^{n} X_i - [n \cdot log(\lambda) + n \cdot log(2 \cdot \lambda)] - \left[\sum_{i=1}^{n} \frac{X_i \cdot Y_i}{\lambda} + \sum_{i=1}^{r} \frac{X_i \cdot Z_{NM}}{2 \cdot \lambda} + \sum_{i=r+1}^{n} \frac{\lambda^k}{\lambda}\right] \tag{11}$$

Equation 11 is what we compute to get the expectation of the joint likelihood and in the next step of the EM algorithm we maximize this expectation with respect to λ to obtain $\lambda^{(k+1)}$:

Maximization Step

$$Q(\lambda, \lambda^{(k)})'_\lambda = -\frac{n}{\lambda} + \sum_{i=1}^{n} \frac{X_i \cdot Y_i}{\lambda^2} - -\frac{n}{\lambda} + \sum_{i=1}^{r} \frac{X_i \cdot Z_{NM}}{2 \cdot \lambda^2} + \frac{(n-r) \cdot \lambda^k}{\lambda^2} \tag{12}$$

Setting the above equation to 0 and solving for λ, we get:

$$\lambda^{(k+1)} = \frac{1}{2n}\left[\sum_{i=1}^{n} X_i \cdot Y_i + \sum_{i=1}^{r} \frac{X_i \cdot Z_i}{2} + (n-r) \cdot \lambda^k\right] \tag{13}$$

Sub Question 3:

We are now asked to implement the above analytically derived algorithm by setting $\lambda_0 = 100$ with convergence criterion "stop if the change in λ is less than 0:001"

The corresponding results for optimal λ and number of iterations is shown below:

```
## The optimal lambda is as seen from the data frame, 10.69566 , and was obtained after  6  iterations
```

```
##   LambdaCurrent LambdaPrevious    LLVals
## 1      14.26782      100.00000 -683.9621
## 2      10.83853       14.26782 -678.9197
## 3      10.70136       10.83853 -679.0048
## 4      10.69587       10.70136 -679.0089
## 5      10.69566       10.69587 -679.0090
```

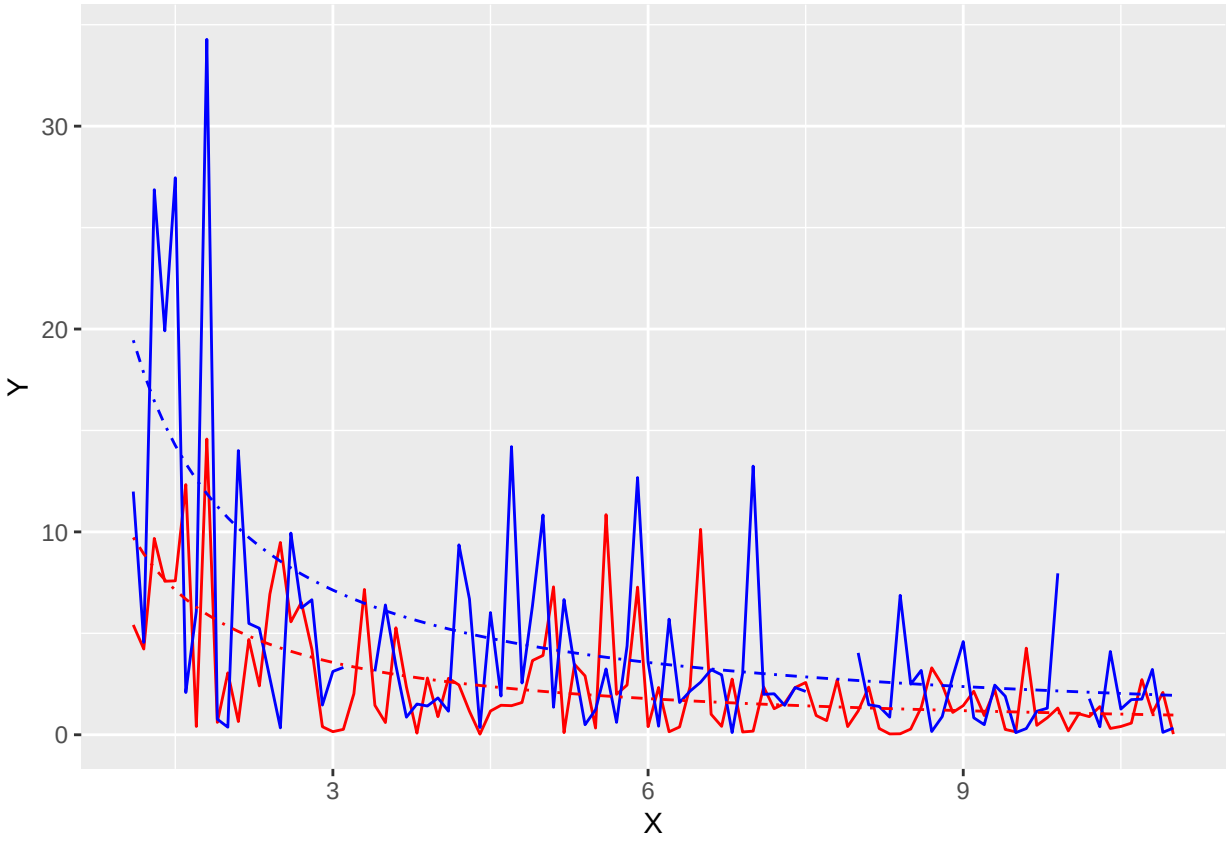
Sub Question 4:

We now compute $E[Y]$ and $E[Z]$ using the obtained optimal lambda value from the algorithm in the following method:

$$E[Y] = \frac{\lambda_{opt}}{X_i}$$

$$E[Z] = \frac{2 \cdot \lambda_{opt}}{X_i}$$

Then, we plot the corresponding values E[Y] and E[Z] in same plot as Y and Z versus X from sub question 1.



The computed λ_{opt}(shown as dot-dashed line) seems to be reasonable as it follows the exponential distribution pattern and also follows the trend of the data.

Appendix

```
library(ggplot2)
library(dplyr)
library(gridExtra)
knitr::opts_chunk$set(echo = TRUE)
maxim_fn <- function(x){
  return((x^2 / exp(x)) - (2 * exp(-(9 * sin(x))/(x^2 + x + 1))))
}
crossover <- function(x, y){
  return((x + y)/2)
}
mutate <- function(x){
  return((x^2) %% 30)
}

run_genetic_fn <- function(maxiter, mutprob){
  data <- data.frame(x = seq(from = 0, to = 30, by = 1),
                     y = maxim_fn(seq(from = 0, to = 30, by = 1)),
                     State = 'Old Point',
                     Mutate = FALSE)
```

```
  # ggplot(data = data, aes(x = x, y = y)) +
  #   geom_point(aes(color = State)) +
  #   ggtitle('Plot of the whole population')
  # cat('Maximum point:', max(data$y))

  init_population <- data.frame(X = 5 * seq(0, 6, 1),
                                Values = maxim_fn(5 * seq(0, 6, 1)),
                                State = 'Old Point')
  # print(init_population)
  for (i in 1:maxiter) {
    # cat(paste('####Iteration', i, '####\n'))
    parents_index <- sample(1:nrow(init_population),2)#Parents
    victim_index <- which.min(init_population$Values)#Victim
    kid <- crossover(init_population$X[parents_index[1]],#Perform crossover
                     init_population$X[parents_index[2]])
    # set.seed(12345)
    is_mutate <- FALSE
    if(mutprob >= runif(1)){#Perform mutation
      kid <- mutate(kid)
      is_mutate <- TRUE
    }
    # cat(paste('Victim:',init_population$X[victim_index],
    #           ' Kid:', kid, '\n'))
    init_population$X[victim_index] <- kid #Override victim
    init_population$Values[victim_index] <- maxim_fn(kid) #Override victim
    init_population$State[victim_index] <- 'New Point' #Override victim
    # print(init_population)
    data <- rbind(data, data.frame(x = kid,
                                   y = maxim_fn(kid),
                                   State = 'New Point',
                                   Mutate = is_mutate))
    # ggplot(init_population %>% arrange(desc(State)), aes(x = X, y = Values)) +
    #   geom_point(aes(color = State))
  }
  # cat('\nPopulation State:\n')
  # print(init_population)
  data <- data %>% arrange(desc(State))
  plot1 <- ggplot(data = data, aes(x = x, y = y)) +
    geom_point(aes(color = State)) +
    ggtitle(paste('Max Iteration:', maxiter, ' Mutate Prob:', mutprob, sep = '')) +
    geom_hline(yintercept = max(data$y))
  new_pop <- data %>% filter(State == 'New Point') %>% select(x, y, Mutate)
  new_pop$Iter <- seq(from = 1, to = maxiter, by = 1)
  new_pop$Type <- rep(paste('MaxIter_',maxiter, '_Prob_', mutprob, sep = ''),nrow(new_pop))
  # plot2 <- ggplot(new_pop, aes(x = Iter, y = y)) +
  #   geom_line() +
  #   ggtitle(paste('Max Iteration:', maxiter, ' Mutate Prob:', mutprob))
  return(list(final_pop = new_pop,
              plot1 = plot1))
}
iter10_prob0.1 <- run_genetic_fn(maxiter = 10, mutprob = 0.1)
iter10_prob0.5 <- run_genetic_fn(maxiter = 10, mutprob = 0.5)
iter10_prob0.9 <- run_genetic_fn(maxiter = 10, mutprob = 0.9)
```

```
grid.arrange(iter10_prob0.1$plot1,
             iter10_prob0.5$plot1,
             iter10_prob0.9$plot1,
             nrow = 3)
maxim_plot_df1 <- rbind(iter10_prob0.1$final_pop,
                        iter10_prob0.5$final_pop,
                        iter10_prob0.9$final_pop)
ggplot(maxim_plot_df1, aes(x = Iter, y = y)) +
  geom_line(aes(color = Type, linetype = Type)) +
  geom_point(data = maxim_plot_df1 %>% filter(Mutate == TRUE),
             aes(x = Iter, y = y)) +
  scale_x_continuous(breaks = seq(from = 1, to = 10, by = 1)) +
  ggtitle('Maximization Graph')

iter100_prob0.1 <- run_genetic_fn(maxiter = 100, mutprob = 0.1)
iter100_prob0.5 <- run_genetic_fn(maxiter = 100, mutprob = 0.5)
iter100_prob0.9 <- run_genetic_fn(maxiter = 100, mutprob = 0.9)
grid.arrange(iter100_prob0.1$plot1,
             iter100_prob0.5$plot1,
             iter100_prob0.9$plot1,
             nrow = 3)
maxim_plot_df <- rbind(iter100_prob0.1$final_pop,
                       iter100_prob0.5$final_pop,
                       iter100_prob0.9$final_pop)
ggplot(maxim_plot_df, aes(x = Iter, y = y)) +
  geom_line(aes(color = Type, linetype = Type)) +
  geom_point(data = maxim_plot_df %>% filter(Mutate == TRUE),
             aes(x = Iter, y = y)) +
  scale_x_continuous(breaks = seq(from = 1, to = 100, by = 10)) +
  ggtitle('Maximization Graph')
physical_data <- read.csv(file = 'physical1.csv')

plot <- ggplot(data = physical_data)+
  geom_line(aes(x = X, y = Y), colour = 'red')+
  geom_line(aes(x = X, y = Z), colour = 'blue')
plot
# The derived log-likelihood for missing and non-missing data is analytically
# derived to be:

exp_loglik <- function(n, y, x, x_z, z, lambda){
  return((2*log(sum(x))-(n*log(lambda)+n*log(2*lambda))-
            ((sum((x*y)/lambda))+(sum((x_z*z)/(2*lambda))))))
}

em_algo <- function(data_req, min, kmax){
  Xobs <- data_req$X
  Yobs <- data_req$Y
  Zobs <- data_req$Z[!is.na(data_req$Z)]
  Zmiss <- data_req$Z[is.na(data_req$Z)]
  x_z <- data_req$X[which(!is.na(data_req$Z))]
  n <- nrow(data_req)
  r <- length(Zobs)
  k <- 1
```

```
  lambda_p <- 0
  lambda_k <- 100
  loglikval_prev <- exp_loglik(n = r, y = Yobs, x = Xobs, x_z = x_z, z = Zobs,
                               lambda = lambda_k)
  loglikval_curr <- loglikval_prev+10+100*min

  lambda_k_vals <- c()
  lambda_p_vals <- c()
  loglikval_curr_vals <- c()
  k_vals <- c()
  while ((abs(lambda_p-lambda_k)>min) && (k<(kmax+1))) {
    loglikval_prev <- loglikval_curr
    lambda_p <- lambda_k

    ## E-step

    E_lambda <- sum(0.5*Zobs*x_z)+(n-r)*lambda_p

    ## M-step
    lambda_k <- (1/(2*n))*(E_lambda+sum(Xobs*Yobs))

    ## Computing current log-likelihood
    loglikval_curr <- exp_loglik(n = r,
                                 y = Yobs,
                                 x = Xobs,
                                 x_z = x_z,
                                 z = Zobs,
                                 lambda = lambda_k)
    k <- k+1
    lambda_k_vals <- c(lambda_k_vals,lambda_k)
    lambda_p_vals <- c(lambda_p_vals, lambda_p)
    loglikval_curr_vals <- c(loglikval_curr_vals, loglikval_curr)

    # print(c(lambda_k, loglikval_curr, k))

  }
  cat('The optimal lambda is as seen from the data frame,',lambda_k,
      ', and was obtained after ', k,' iterations \n')
  em_df <- data.frame('LambdaCurrent' = lambda_k_vals,
                      'LambdaPrevious' = lambda_p_vals,
                      'LLVals' = loglikval_curr_vals)
  return(em_df)
}

em_algo(data_req = physical_data, min = 0.001, kmax = 1000)


em_vals <- em_algo(data_req = physical_data, min = 0.001, kmax = 1000)
optimal_lambda <- em_vals[nrow(em_vals), 1]

expVal_Y <- optimal_lambda/physical_data[,1]
expVal_Z <- (2*optimal_lambda)/physical_data[,1]
```

```
exp_vals <- data.frame('ExpectedValsY' = expVal_Y,
                       'ExpectedValsZ' = expVal_Z,
                       'XVals' = physical_data$X)

plot+
  geom_line(data = exp_vals,aes(x = XVals, y = ExpectedValsY), colour = 'red',
            linetype = 'dotdash')+
  geom_line(data = exp_vals,aes(x = XVals, y = ExpectedValsZ), colour = 'blue',
            linetype = 'dotdash')
```