Computational statistics Lab06

Saewon Jun(saeju204)

Question 1: Genetic algorithm

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

1-1. Define the function

$$f(x) := \frac{x^2}{e^x} - 2e^{-\left(\frac{9\sin x}{x^2+x+1}\right)}$$

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

1-2. Define the function crossover() : for two scalars x and y it returns their "kid" as $\frac{x+y}{2}$.

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

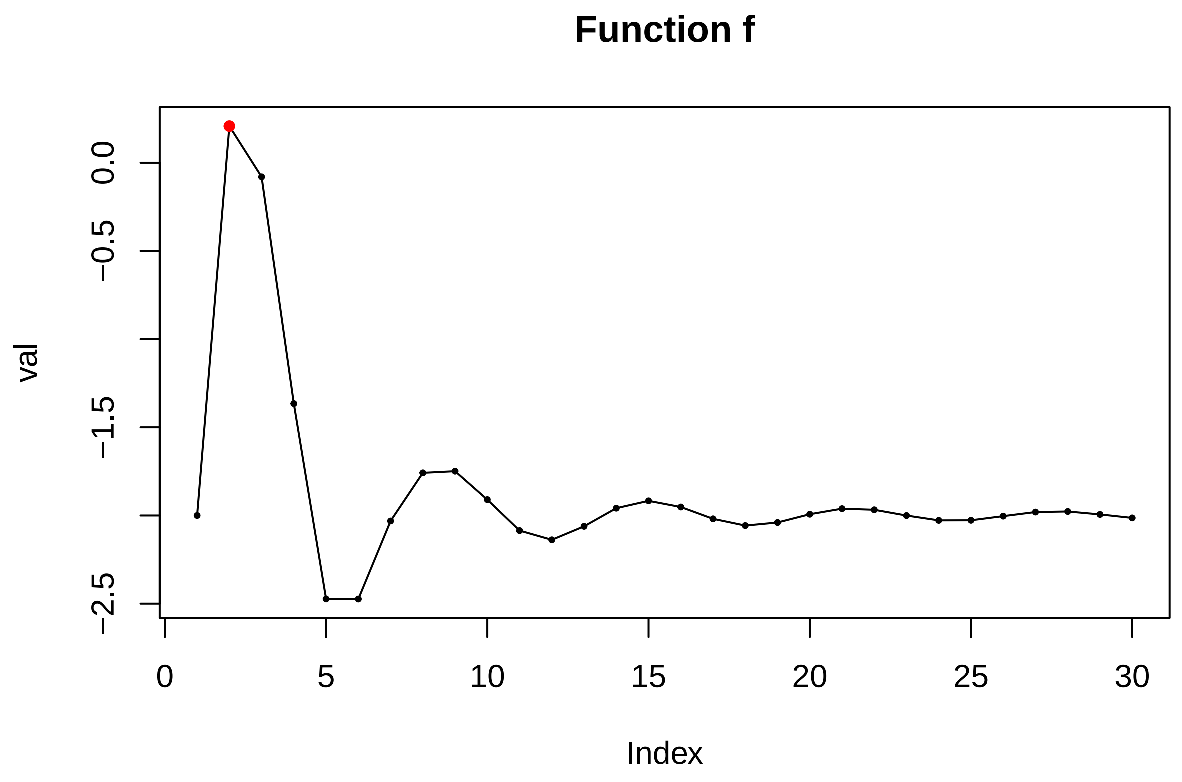
1-3. Define the function mutate() that for a scalar x returns the reslt of the integer division x^2 mod 30.

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

1-4.Write a function that depends on the parameters maxiter and mutprop.

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

```r
val <- numeric()

for (i in 0:29){
        val[i+1] <- f(i)
}

plot(val, main="Function f", type="o", cex=0.5, pch=20, col="black")
points(2,val[2], pch=20, col="red")
```



```r
print("maximum value:")
```

```
## [1] "maximum value:"
```

```r
print(val[2])
```

```
## [1] 0.2076688
```

The maximum value is observed at the very beginning of the range.

(b) Define an initial population for the genetic algorithms as X=(0,5,10,15,. . . 30)

(c) Compute vector Values that contains the function value each populaton point.

(d) Performs maxiter iterarions where at iterations

- Two indexes are randomly sampled from the current popultaion, they are further used as parents.
- One index with the smallest objective function is selected from the current population, the point is referred to as victim.
- Parents are used to produce a new kid by crossover. Mutate this kind with probability mutprob.
- The victim is replaced by the kid *in the population* and the vector *Values* is updated.
- The current maximal value of the objective function is saved.

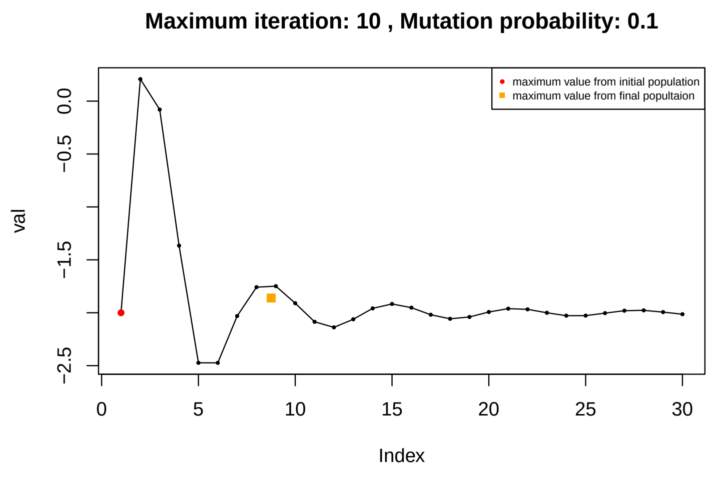
(e) Add the final observations to the current plot in another colour

```r
Genetic <- function(maxiter, mutprob){
        X <- seq(0,30,5) #define initial popultaion
        Values <- f(X) #Values that contains the function value each populaton point.

         #maximums <- numeric()

        set.seed(12345)
        for (i in 1:maxiter){
                x <- sample(X,1)
                y <- sample(X,1) # shgould not be the same
                victim <- which.min(order(Values))
                #this gives you the index of the victim

                kid <- crossover(x,y)

                if(as.numeric(rbinom(1,1,mutprob))==1){
                        kid <- mutate(kid)
                }

                #update the population
                X[victim] <- kid
                #update the Values
                Values <- f(X)

                #save the current maiximal value..?
                #maximums[i] <- max(Values)


        }

        max_val <- max(Values) #is it correct ...?
        index_max_val <- X[which.max(Values)]

        title <- paste("Maximum iteration:",maxiter,
                       ", Mutation probability:",mutprob)
        plot(val, main=title, type="o", cex=0.5, pch=20, col="black")
        points(val[1], pch=20, col="red")
        points(index_max_val, max_val, pch=15, col="orange")
        legend("topright", cex=0.6,
               c("maximum value from initial population",
                 "maximum value from final popultaion"),
               pch=c(20,15),col=c("red","orange"))

        return(list("final_popultaion"=X, "values"=f(X), "maximum_value"=max_val))


}
```

1-5. Run your code with different combinations of maxiter=10,100 and mutprob=0.1,0.5,0.9

maxiter=10, mutprob=0.1

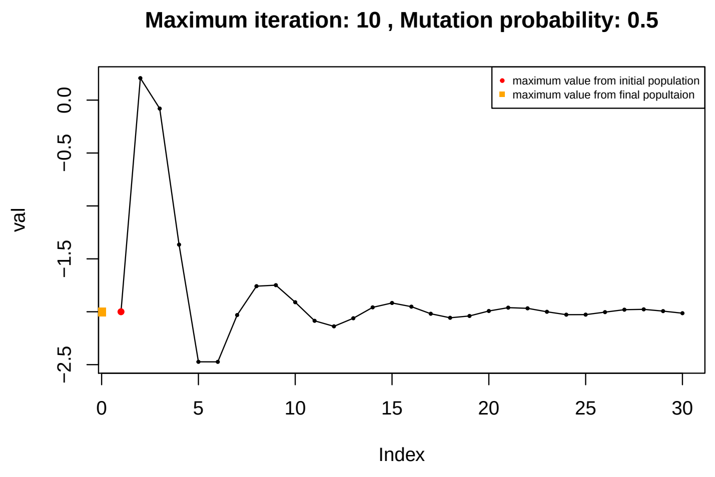
```
Genetic(10,0.1)
```



```
## $final_popultaion
## [1]  0.00  5.00 10.00 12.50 17.50  8.75 30.00
##
## $values
## [1] -2.000000 -2.473573 -2.085654 -2.006463 -2.054806 -1.861738 -2.019194
##
## $maximum_value
## [1] -1.861738
```

maxiter=10, mutprob=0.1

```
Genetic(10,0.5)
```

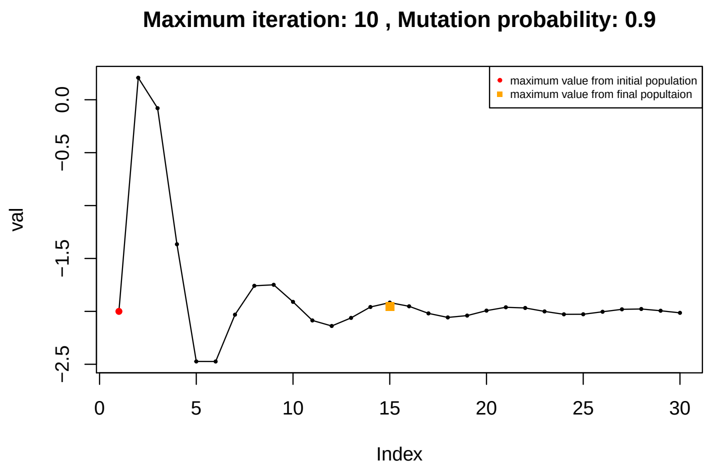


```
## $final_popultaion
## [1]  0.00  5.00 10.00 17.50 10.00 10.00 11.25
##
## $values
## [1] -2.000000 -2.473573 -2.085654 -2.054806 -2.085654 -2.085654 -2.127872
```

```
##
## $maximum_value
## [1] -2
```

maxiter=10, mutprob=0.1

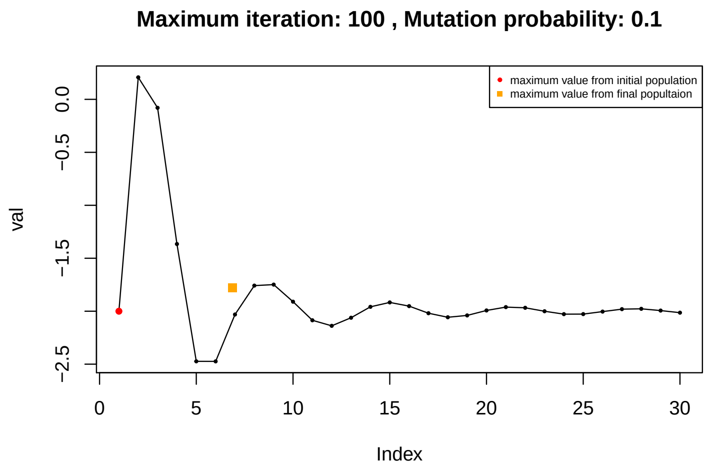
```
Genetic(10,0.9)
```



```
## $final_popultaion
## [1]  0  5 10 15 10 10 30
##
## $values
## [1] -2.000000 -2.473573 -2.085654 -1.951947 -2.085654 -2.085654 -2.019194
##
## $maximum_value
## [1] -1.951947
```

maxiter=100, mutprob=0.1

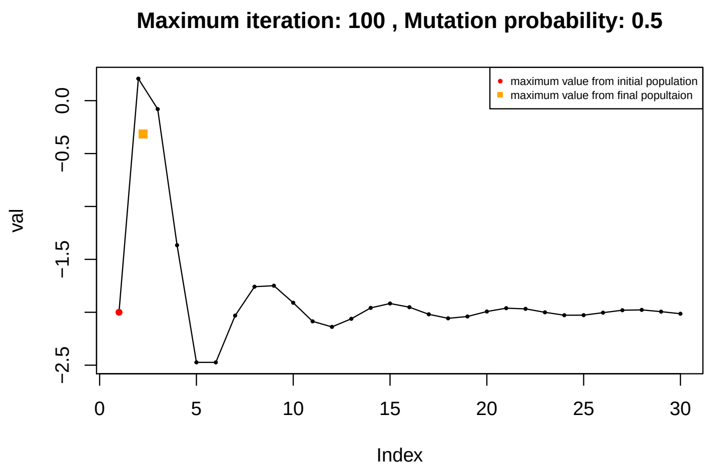
```
Genetic(100,0.1)
```



```
## $final_popultaion
## [1]  6.8750 13.7500 12.7832 13.7500 13.7500 13.7500  6.8750
##
## $values
## [1] -1.777096 -1.919668 -1.977806 -1.919668 -1.919668 -1.919668 -1.777096
##
## $maximum_value
## [1] -1.777096
```

maxiter=100, mutprob=0.5

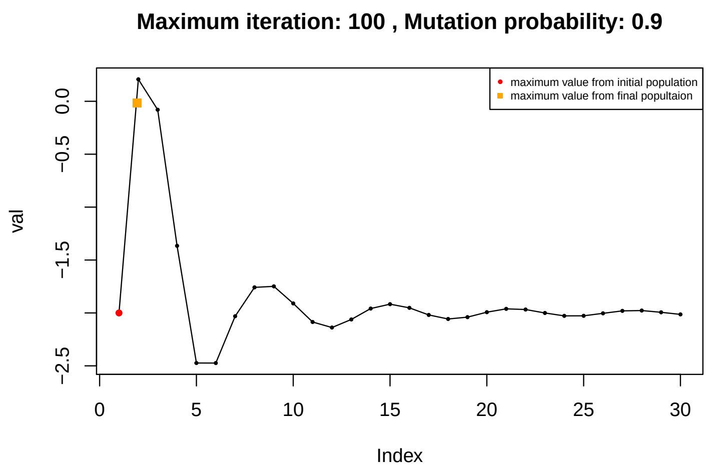
```
Genetic(100,0.5)
```



```
## $final_popultaion
## [1]  2.240781  3.280888 19.988128 11.634508  4.470808  7.457698  2.240781
##
## $values
## [1] -0.3174155 -1.7685282 -1.9615061 -2.0988557 -2.5904022 -1.7248471
## [7] -0.3174155
##
## $maximum_value
## [1] -0.3174155
```

maxiter=100, mutprob=0.9

```
Genetic(100,0.9)
```



```
## $final_popultaion
## [1]  1.922653  2.187682  2.533351  7.743351  2.187682  4.963994 24.656355
##
## $values
## [1] -0.01758621 -0.25962621 -0.68331733 -1.72984356 -0.25962621 -2.48689612
## [7] -2.01307025
##
## $maximum_value
## [1] -0.01758621
```

When maximum iteration number is 10, maximum value from final population does not reach the maximum value from initial poplulation, and this is due to lack of iteration. With maximum iteration number equals to 100, we can observe that the maximum value from final population reaches the maximum value from initial popultaion as you increase the mutation probability. We can conclude that one-dimensional maximaization with the genetic algorithm shows the best perfomrmance when number of iteration is 100 and mutation probability is 0.9 among given combinations.

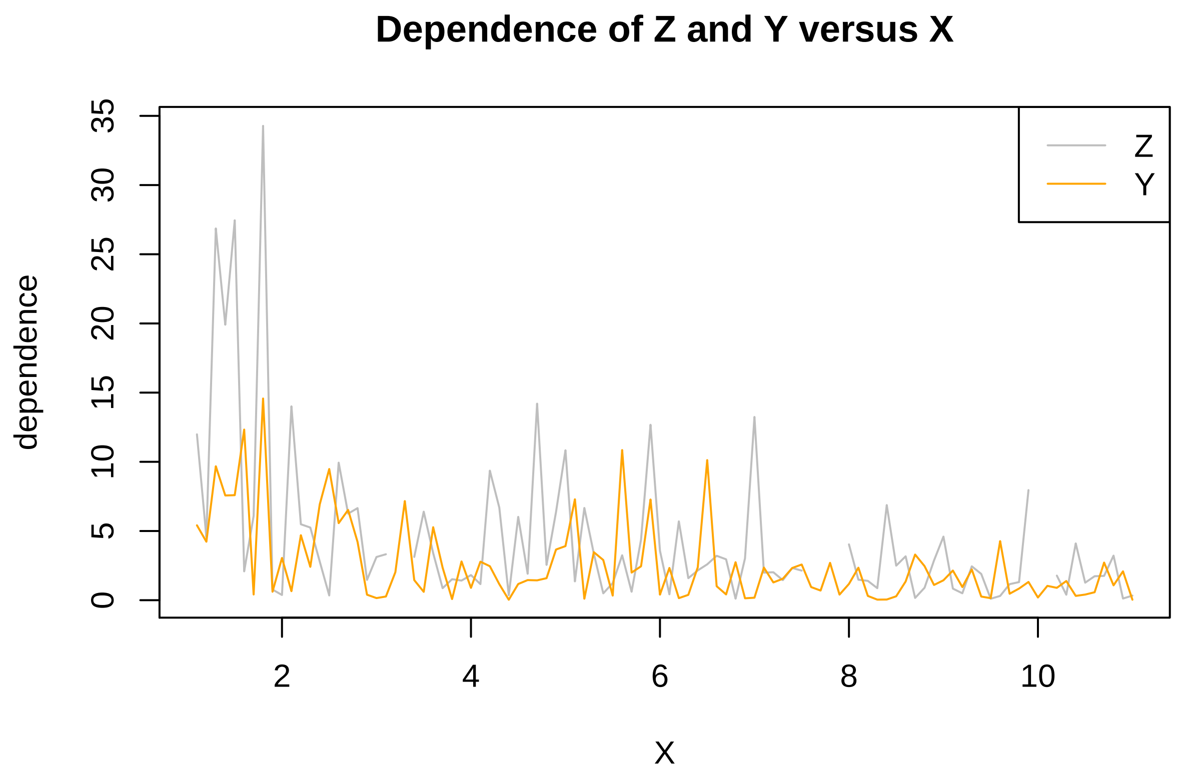
Question 2: EM algorithm

The data file *physical.csv* describes a behavior of two related physical process $Y = Y(X)$ and $Z = Z(X)$.

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")

plot(data$X, data$Z, type="l", col="grey", cex=0.4,
        main="Dependence of Z and Y versus X", xlab="X", ylab="dependence")
points(data$X, data$Y, type="l", col="orange")
legend("topright", legend=c("Z","Y"),
       col = c("grey","orange"), lty = c(1,1))
```



They are not perfectly same, but we can say that both process Z and Y follows similar pattern as X goes. Two processes both show higher variance at the very beginning of the X, which seems to be decrease at the end of the X. In overall, Z has higher variance over entire data then Y.

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

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

where λ is some unknown parameter. X,Y and Z has 100 obersvations each but some missing points(NA) in Z. So for the following steps, we set n to the length Y and Z, and m to the number of missing values in Z. **The goal is to derive an EM algorithm that estimates λ.**

Derive Log-liklihood function

$$\begin{aligned}
L(\lambda|Y,Z) &= \prod_{i=1}^{n} f(Y) \times \prod_{i=1}^{n} f(Z) \\
&= \prod_{i=1}^{n} \frac{X_i}{\lambda} \cdot e^{-\frac{X_i}{\lambda}Y_i} \times \prod_{i=1}^{n} \frac{X_i}{2\lambda} \cdot e^{-\frac{X_i}{\lambda}Z_i} \\
&= \frac{X_1 \cdot \ldots \cdot X_n}{\lambda^n} \times e^{-\frac{1}{\lambda}\sum_1^n X_iY_i} \times \frac{X_1 \cdot \ldots \cdot X_n}{(2\lambda)^n} \times e^{-\frac{1}{2\lambda}\sum_1^n X_iZ_i} \\
lnL(\lambda|Y,Z) &= \sum_{i=1}^{n} ln(X_i) - nln(\lambda) - \frac{1}{\lambda}\sum_{i=1}^{n} X_iY_i \ + \sum_{i=1}^{n} ln(X_i) - nln(2\lambda) - \frac{1}{2\lambda}\sum_{i=1}^{n} X_iZ_i
\end{aligned}$$

E-step : Derive Q function

Obtaining the expected values for the missing data using an initial parameter estimate.

$$
\begin{aligned}
Q(\theta,\theta^k) &= E[\ loglik(\lambda|Y,Z)\mid \lambda^k,(Y,Z)] \\
&= \sum_{i=1}^{n} ln(X_i) - nln(\lambda) - \frac{1}{\lambda}\sum_{i=1}^{n} X_iY_i\ + \sum_{i=1}^{n} ln(X_i) - nln(2\lambda) \\
&\quad - \frac{1}{2\lambda}\left[\sum_{i=1}^{n} X_iZ_i\ + m\cdot X_i\cdot\frac{2\lambda_{k-1}}{X_i}\right]
\end{aligned}
$$

Here, we are taking expectation on the missing values in Z, so we need to seperate the Z_{obs} and Z_{miss}.

M-step

Obtain the maximum likelihood estimate of the parameters by taking the derivative with respect to λ. Repeat till estimate converges.

$$
\begin{aligned}
-\frac{n}{\lambda} - \frac{n}{\lambda}\ + \frac{\sum_{i=1}^{n} X_iY_i}{\lambda^2}\ + \frac{\sum_{i}^{m} X_iZ_i\ + m\cdot 2\lambda_{k-1}}{2\lambda^2} &:= 0 \\
-2\lambda(2n) + 2\sum_{i=1}^{n} X_iY_i\ + \sum_{i=1}^{n} X_iZ_i + m\cdot 2\lambda_{k-1} &:= 0
\end{aligned}
$$

$$
\lambda = \frac{\sum_{i=1}^{n} X_iY_i + \frac{1}{2}\sum_{i=1}^{n} X_iZ_i + m\cdot\lambda_{k-1}}{2n}
$$

Now we implement this to function called *EM.missing()* that depends on the parameters **data** and **eps** and **kmax**.

```
EM_missing <- function(data,eps,kmax,lamb_0){

        X <- data$X
        Y <- data$Y
        Z <- data$Z

        Xobs <- X[!is.na(Z)]
        Zobs <- Z[!is.na(Z)]
        Zmiss <- Z[is.na(Z)]

        n <- length(X)
        m <- length(Zmiss)

        k <<- 0
        llvalprev <- 0
        llvalcurr <- lamb_0

        print(c(llvalprev,llvalcurr,k))

        while ((abs(llvalprev-llvalcurr)>eps) && (k<(kmax+1))){
                llvalprev <- llvalcurr
                llvalcurr <- (sum(X*Y)+sum(Xobs*Zobs)/2+m*llvalprev)/(2*n)
```

```r
                k <<- k+1
        }

        print(c(llvalprev,llvalcurr,k))
}
```

2-3 Implement the algorithm, use $\lambda_0 = 100$, and convergence criterion "stop if the change in lambda is less than 0.001". What is optimal λ and how many iterations were required to compute it?

```r
EM_missing(data,0.001,50,100)
```

```
## [1]   0 100   0
## [1] 10.69587 10.69566  5.00000
```

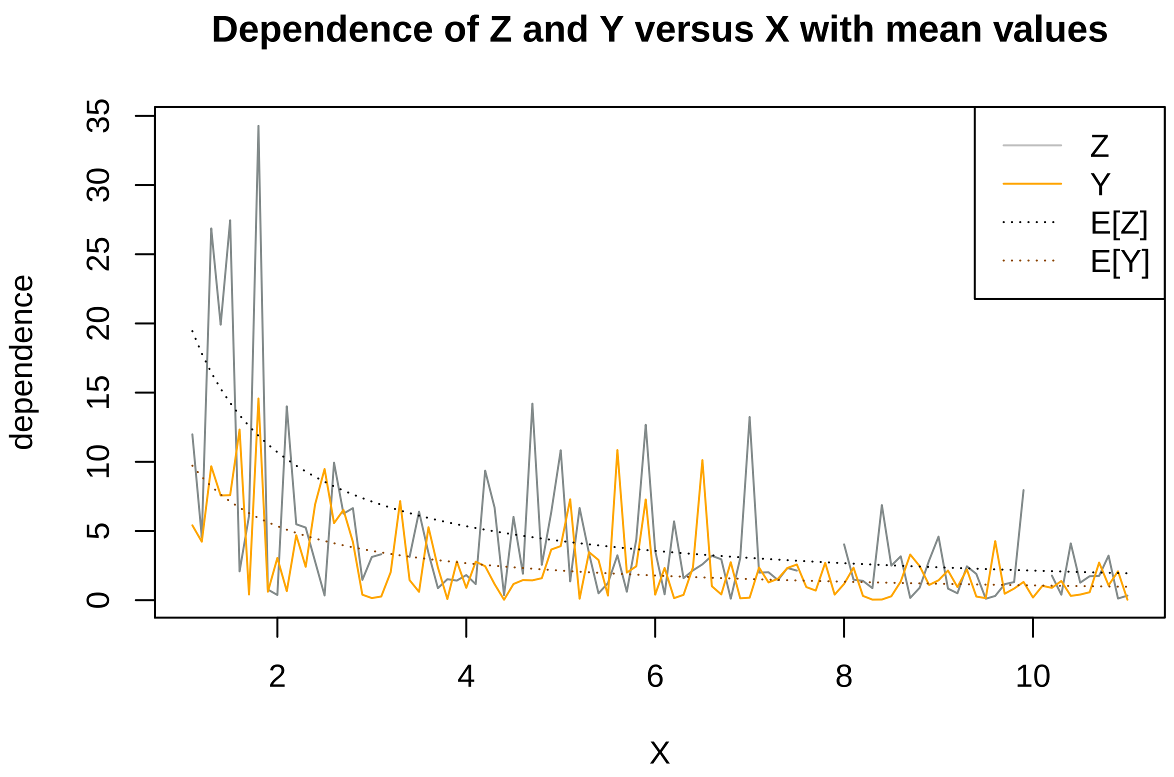
The result indicates that the optimal lambda is 10.69566 and 5 iteration is required to compute it.

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.

Following the given model for Y and Z, we can easily derive the mean value with obtained λ.

$$E[Y] = \frac{\lambda}{X_i}\ , \quad E[Z] = \frac{2\lambda}{X_i}$$

```r
lambda <- 10.69566

plot(data$X, data$Z, type="l", col="azure4", cex=0.4,
        main="Dependence of Z and Y versus X with mean values", xlab="X", ylab="dependence")
points(data$X, data$Y, type="l", col="orange")
points(data$X, lambda/data$X, type="l", lty=3, col="darkorange4")
points(data$X, 2*lambda/data$X, type="l", lty=3)
legend("topright", legend=c("Z","Y","E[Z]","E[Y]"),
       col = c("grey","orange","black","darkorange4"), lty = c(1,1,3,3))
```



From the plot above, we can see that each E[Z] and E[Y] captures the flow of Z and Y on X respectively. So we can say that our computed λ is reasonable enough.