# Computational statistics Lab06

*Saewon Jun(saeju204), Min-chun Shih(shimi077)*

## 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

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

```r
# implement the function above
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
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
mutate <- function(x) {
  res <- (x^2) %%30
  return (res)
}
```

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

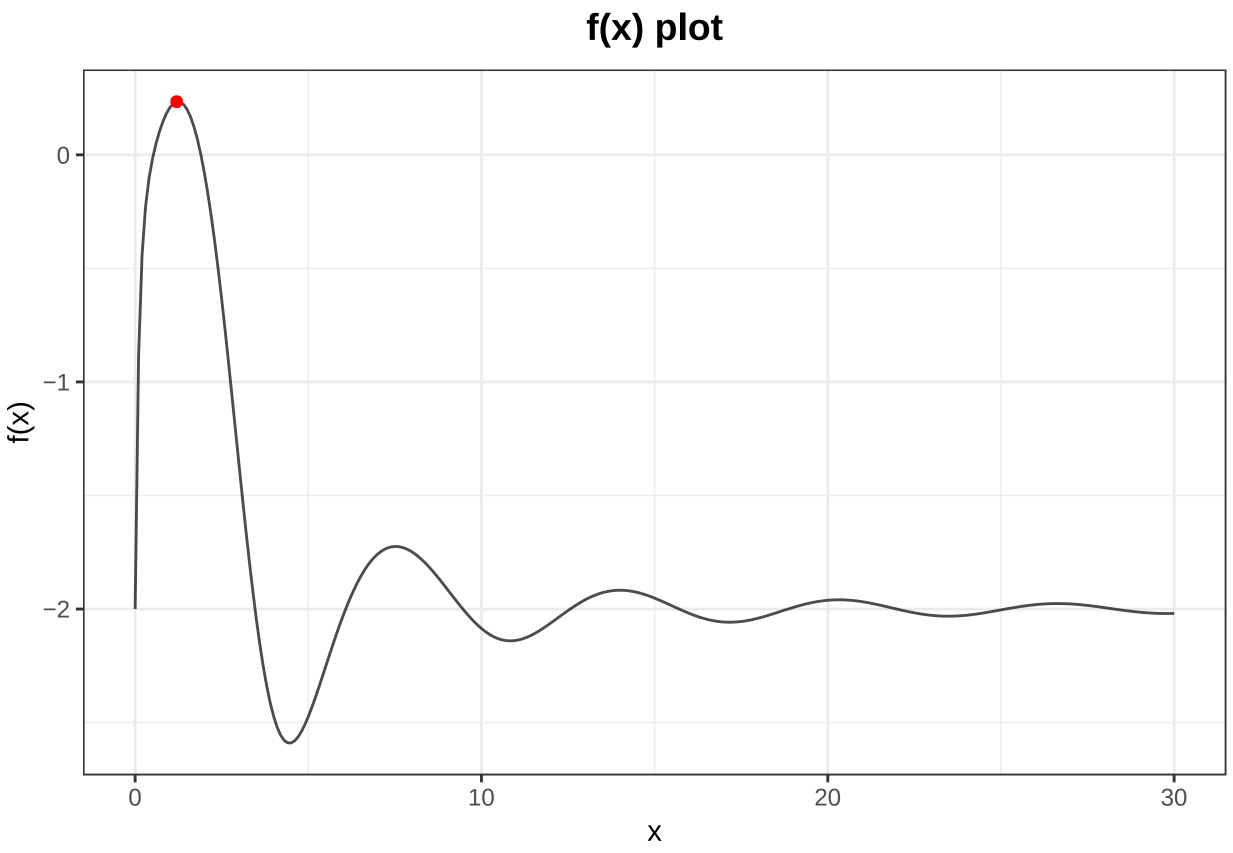
### (a) Plots function $f$ in the range from 0 to 30. Do you see anu maximum value?

```r
library(ggplot2)

x <- seq(0, 30, 0.1)
y <- sapply(x, f)
d <- data.frame(x = x, y = y)
n <- which.max(d$y)

plot <- ggplot(data = d, aes(x = x, y = y)) +
  # geom_point() +
  geom_line(alpha = 0.7) +
  geom_point(aes(d[n, "x"], d[n, "y"]), colour = "red") +
  theme_bw() +
  labs(y = "f(x)", title = "f(x) plot") +
  theme(plot.title = element_text(size = rel(1.3), face = "bold", hjust = 0.5, margin = unit(c(0, 0, 3,

plot
```



The maximum value is y = 0.20766879 when x = 1. The maximum value is marked in red.

**(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 algorithm
Genetic <- function(maxiter, mutprob) {
  # initialize points and value
  X <- seq(0, 30, 5)
  Values <- sapply(X, f)
  # idx for values
  idx <- X

  for (i in 1:maxiter) {
    # (i) parents points
    parents <- sample(idx, size = 2)

    # (ii) minimum value(victim) index
    indx_victim <- order(Values)[1]

    # (iii) get kid point (crossover parents)
    kid <- crossover(parents[1], parents[2])
    # mutate
    if (runif(1) < mutprob) {
      kid <- mutate(kid)
    }
    # (iv) replace minimum value as f(kid) value
    Values[indx_victim] <- f(kid)
    # replace the X value of kid
    idx[indx_victim] <- kid
  }
  # (v)
  max_value <- max(Values)
  n <- which.max(Values)
  # return (list(which.max(Values), max_value))
  return (list(idx[n], max_value, idx))
}
```

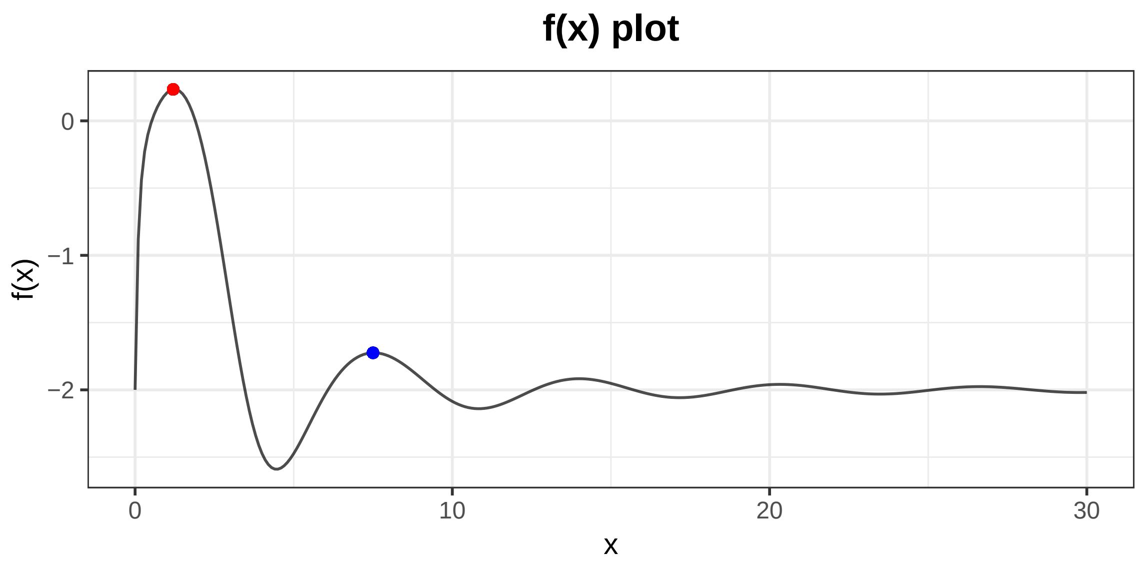
**1-5. Run your code with different combinations of maxiter=10,100 and mutprob=0.1,0.5,0.9**

```r
set.seed(100)
X <- seq(0, 30, 5)

# maxiter = 10, mutprob = 0.1
res <- Genetic(10, 0.1)
indx <- res[[1]]
max_value <- res[[2]]
final <- res[[3]]
final
```

```
## [1]  0.000  7.500 15.000 15.000 20.000 14.375 13.750
```

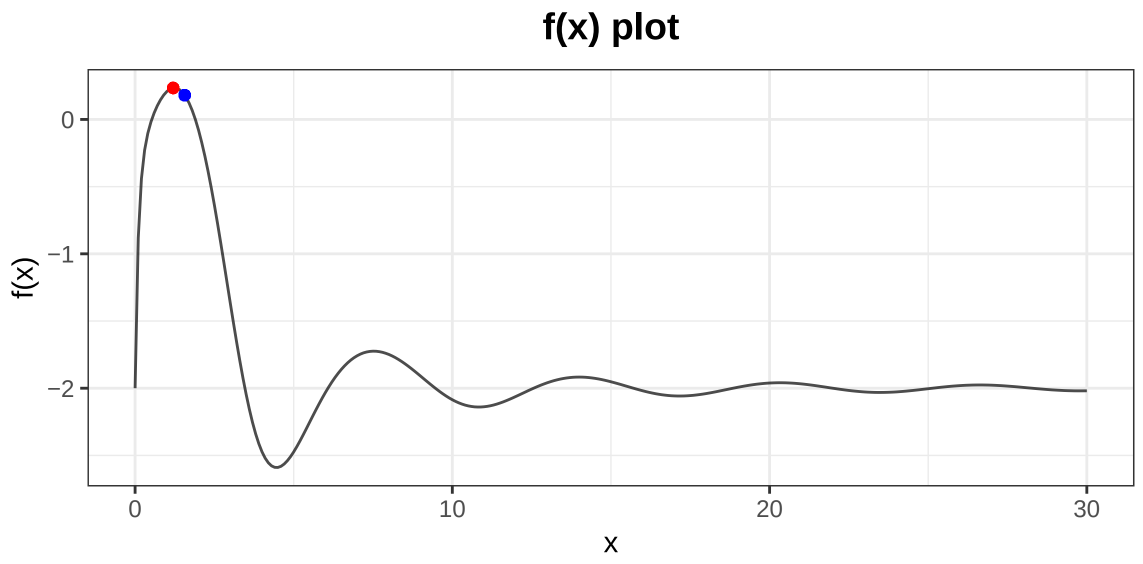
```r
plot + geom_point(aes(x = indx, y = max_value), colour = "blue")
```



```
# maxiter = 10, mutprob = 0.5
res <- Genetic(10, 0.5)
indx <- res[[1]]
max_value <- res[[2]]
final <- res[[3]]
final
```

```
## [1]  8.31617 26.39160 20.00000 15.00000 20.00000 28.04517  1.56250
```

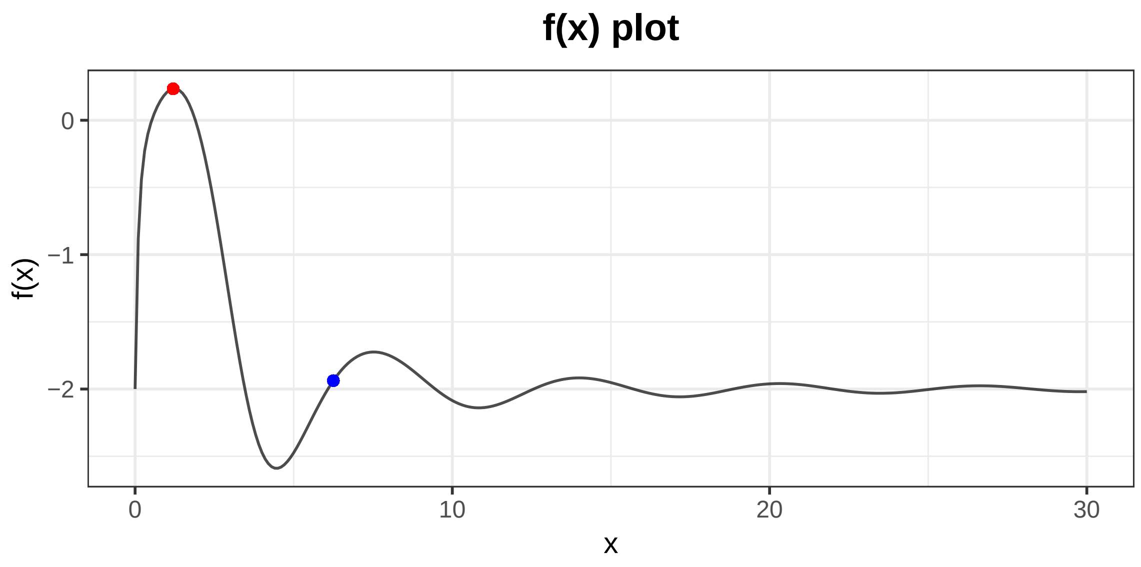
```
plot + geom_point(aes(x = indx, y = max_value), colour = "blue")
```



```
# maxiter = 10, mutprob = 0.9
res <- Genetic(10, 0.9)
indx <- res[[1]]
max_value <- res[[2]]
final <- res[[3]]
final
```

```
## [1] 20.64850  6.25000 15.00000 15.00000 20.00000 22.89062 15.00000
```

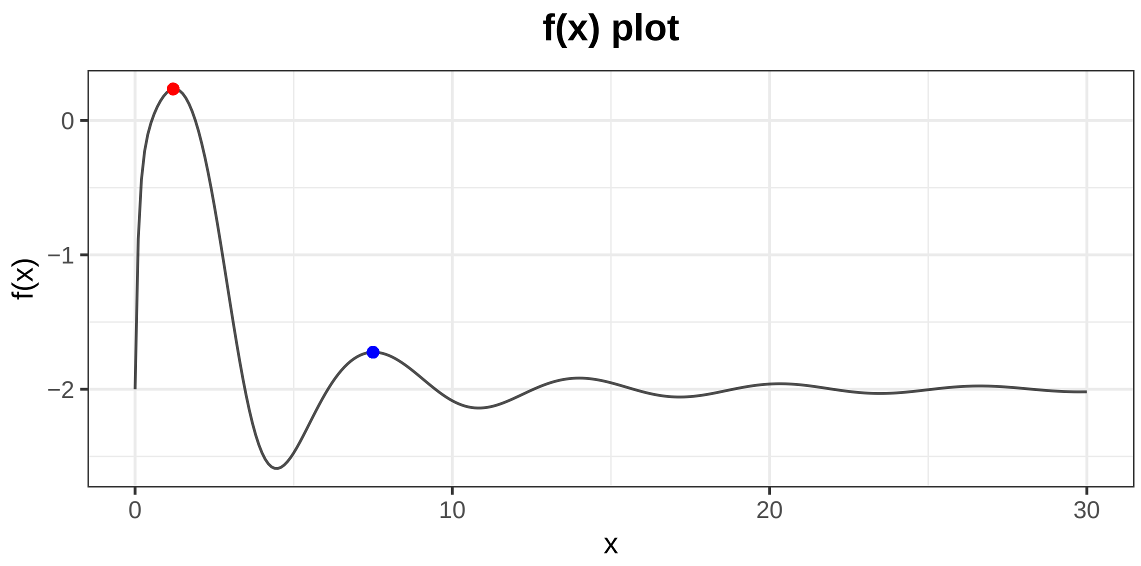
```
plot + geom_point(aes(x = indx, y = max_value), colour = "blue")
```



```
# maxiter = 100, mutprob = 0.1
res <- Genetic(100, 0.1)
indx <- res[[1]]
max_value <- res[[2]]
final <- res[[3]]
final
```

```
## [1] 7.500000 7.483739 7.491869 7.475608 7.487804 7.471543 7.479674
```

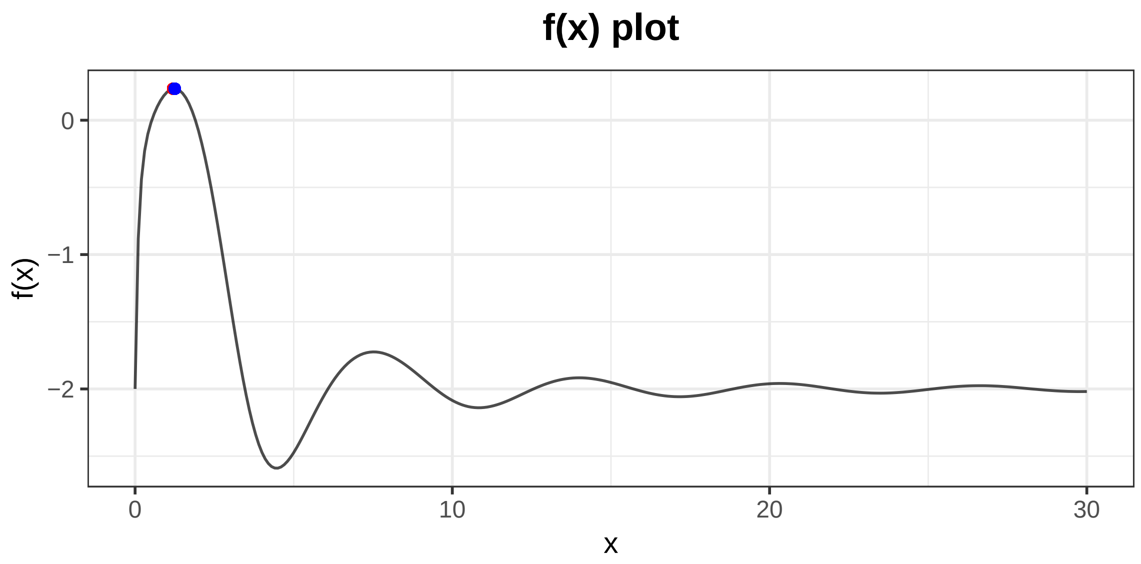
```
plot + geom_point(aes(x = indx, y = max_value), colour = "blue")
```



```
# maxiter = 100, mutprob = 0.5
res <- Genetic(100, 0.5)
indx <- res[[1]]
max_value <- res[[2]]
final <- res[[3]]
final
```

```
## [1] 1.249300 1.252813 1.249300 1.251057 1.249300 1.252813 1.249300
```

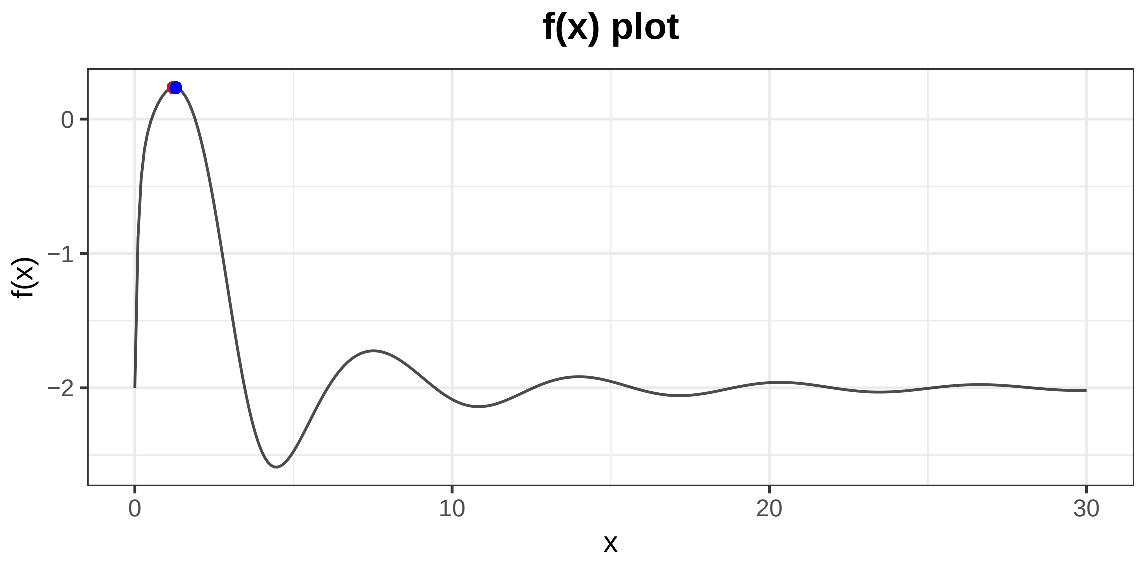
```
plot + geom_point(aes(x = indx, y = max_value), colour = "blue")
```



```
# maxiter = 100, mutprob = 0.9
res <- Genetic(100, 0.9)
indx <- res[[1]]
max_value <- res[[2]]
final <- res[[3]]
final
```

```
## [1] 1.314945 1.341622 1.288894 1.368299 1.314945 1.176695 1.835919
```

```
plot + geom_point(aes(x = indx, y = max_value), colour = "blue")
```



From the plots above, we can see the aspect from iteration first, when the iteration is 100 the plots perform better than when iteration = 10. The final maximum values are closer to the initial maximum value. Hence, we can say, larger iteration will get a better result. (But somehow, the sample() influences the plots)

For the same iteration, higher mutation probability tends to have better performance. When variable mutprob = 0.9, which means there are higherprobability of mutation, the result is closer to the maximum value from initial population.

## Question 2: EM algorithm

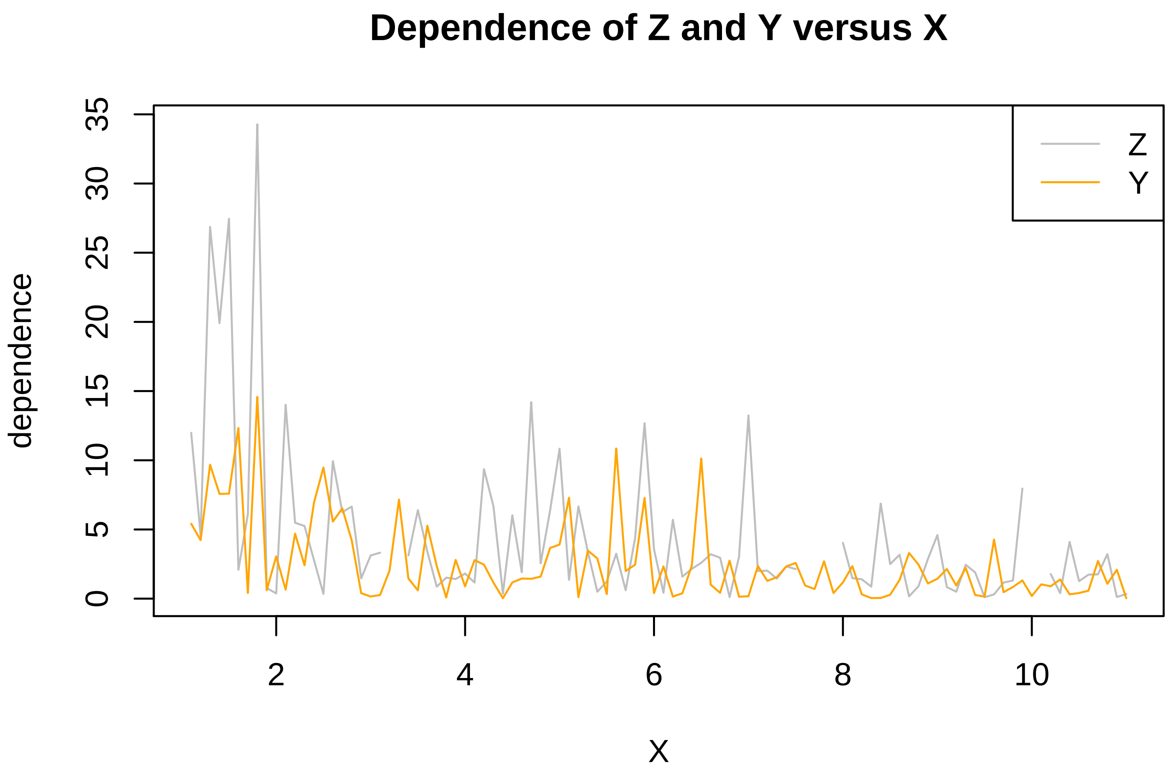
The data file *physical.csv* describes a behavior of two related physical process $Y = Y(X)$ and $Z = Z(X)$.

### 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?

```r
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 $\lambda$ 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 $\lambda$.**

### Derive Log-liklihood function

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

### E-step : Derive Q function

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

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

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 $\lambda$. Repeat till estimate converges.

$$
\begin{aligned}
-\frac{n}{\lambda} - \frac{n}{\lambda} \; + \frac{\sum_{i=1}^{n} X_i Y_i}{\lambda^2} \; + \frac{\sum_i^m X_i Z_i \; + m \cdot 2\lambda_{k-1}}{2\lambda^2} &:= 0 \\
-2\lambda(2n) + 2\sum_{i=1}^{n} X_i Y_i \; + \sum_{i=1}^{n} X_i Z_i + m \cdot 2\lambda_{k-1} &:= 0
\end{aligned}
$$

$$
\lambda = \frac{\sum_{i=1}^{n} X_i Y_i + \frac{1}{2}\sum_{i=1}^{n} X_i Z_i + m \cdot \lambda_{k-1}}{2n}
$$

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)

                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 $\lambda$ and how many iterations were required to compute it?**

```
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.

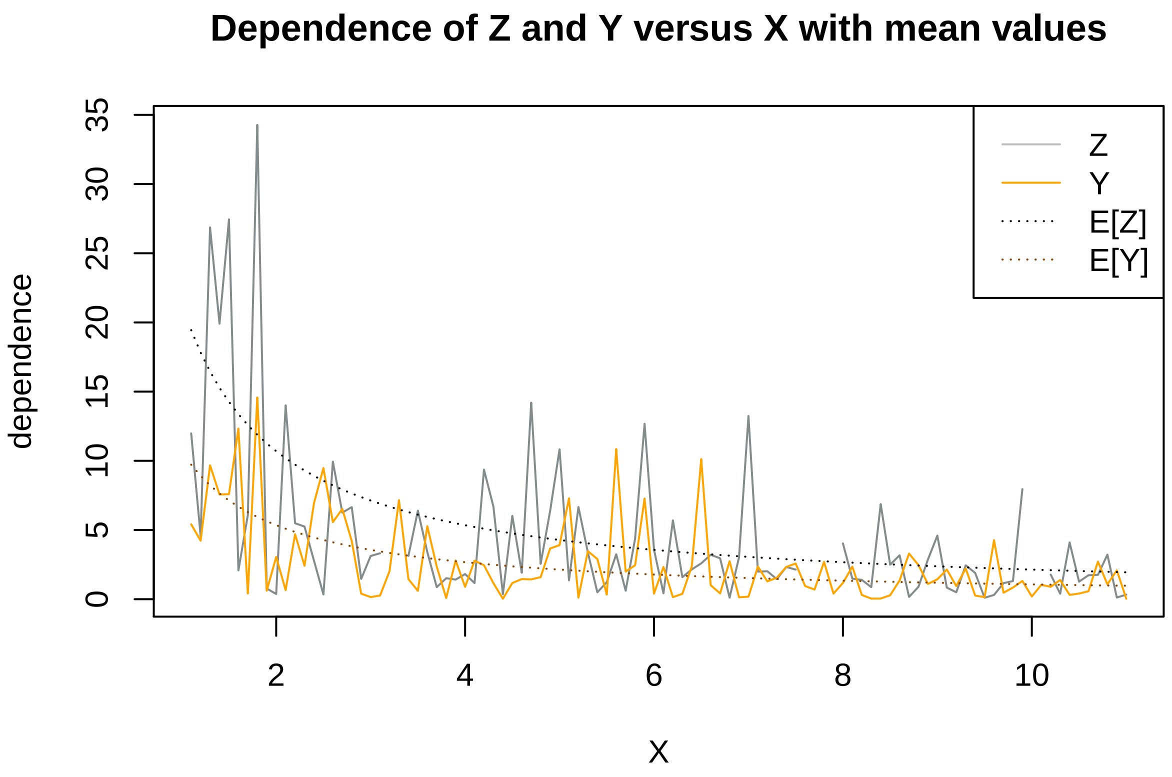
**2-4 Plot E[Y] and E[Z] versus X in the same plot as Y and Z versus X. Comment whether the computed $\lambda$ seems to be reasonable.**

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

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

```
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 $\lambda$ is reasonable enough.