Computational Statistics (732A90) Lab6

Anubhav Dikshit(anudi287) and Thijs Quast(thiqu264)

2 March 2019

Contents

Question 1: Genetic algorithm

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

1. Define the function

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

```
function_f <- function(x){
answer <- x^2/exp(x) - 2*exp(-(9*sin(x))/(x^2+x+1))
return(answer)
}
```

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

```
crossover <- function(x,y){
answer <- (x+y) * 0.5
return(answer)
}
```

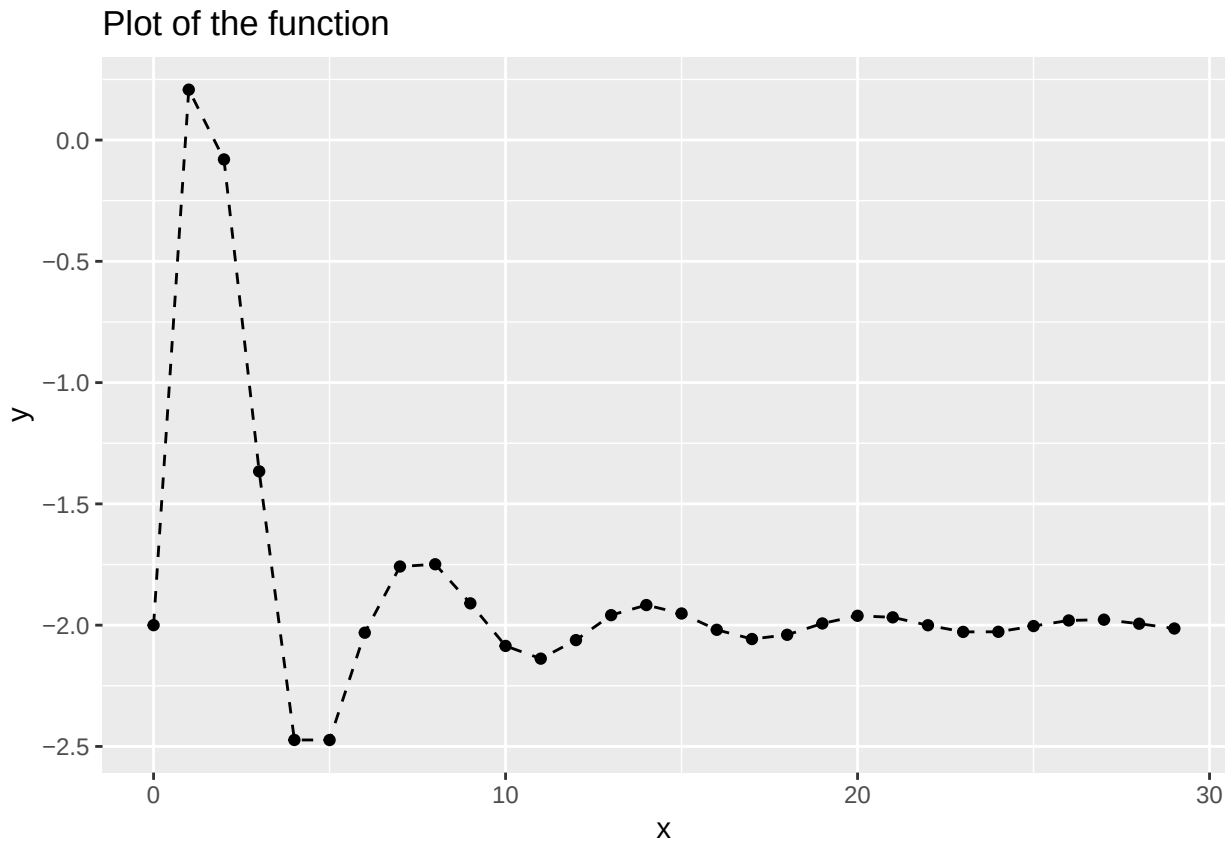
3. Define the function mutate() that for a scalar x returns the result of the integer division x2 mod 30. (Operation mod is denoted in R as %%).

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

4. Write a function that depends on the parameters maxiter and mutprob and:

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

```
y <- vector("double", length = 30)
x <- seq(from=0,to=29,by=1)
for(i in 0:29){y[i+1] <- function_f(x=i)}

data <- data.frame(cbind(x,y))
data$flag <- "original_values"
data$x <- as.numeric(as.character(data$x))
data$y <- as.numeric(as.character(data$y))

ggplot(data, aes(x=x, y=y, group=1)) +
geom_point() +
geom_line(linetype = "dashed") +
ggtitle("Plot of the function")
```




```
cat("The maximum values of the function is at x =",which.max(data$y), "and the value is = ",max(data$y))
```

```
## The maximum values of the function is at x = 2 and the value is =  0.2076688
```

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

```
inital_population <- seq(from=0, to=30, by=5)
inital_population
```

```
## [1]  0  5 10 15 20 25 30
```

(c) Computes vector Values that contains the function values for each population point.

```
# generating new values
Values <- vector("double", length = 7)
for(i in seq_along(inital_population)){Values[i] <- function_f(x=inital_population[i])}
original_max <- max(Values)
original_values <- Values
original_populations <- inital_population
original_values
```

```
## [1] -2.000000 -2.473573 -2.085654 -1.951947 -1.961344 -2.003663 -2.019194
```

(d) Performs maxiter iterations where at each iteration

- Two indexes are randomly sampled from the current population, they are further used as parents (use sample()).
- One index with the smallest objective function is selected from the current population, the point is referred to as victim (use order()).
- Parents are used to produce a new kid by crossover. Mutate this kid with probability mutprob (use crossover(), mutate()).
- 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.

```
myfunction <- function(maxiter, mutprob){

# maxiter = 10
# mutprob = 0.5


for(i in seq_along(1:maxiter)){
# sampling and getting parent and victim position
parent_index <- sample(x = seq(from = 1, to=7, by=1), size = 2, replace = TRUE)
parent_1 <- Values[parent_index[[1]]]
parent_2 <- Values[parent_index[[2]]]
victim <- which.min(order(Values))

# performing crossover and mutate based on prob
kid <- crossover(x = parent_1, y = parent_2)

if(as.numeric(rbinom(1,1,mutprob))==1){kid <- mutate(x=kid)}
inital_population[victim] <- kid

# generating new values
for(j in seq_along(inital_population)){Values[j] <- function_f(x=inital_population[j])}

}
new_max <- max(Values)
index_max_val <- inital_population[which.max(Values)]

data2 <- data.frame(cbind(x=inital_population,y=Values))
data2$flag <- "new_values"
data2$x <- as.numeric(as.character(data2$x))
data2$y <- as.numeric(as.character(data2$y))
data3 <- rbind(data2,data)

answer <- ggplot(data3, aes(x=x, y=y, group=1)) +
geom_point() +
geom_point(aes(colour=flag)) +
geom_line(linetype = "dashed") +
ggtitle(paste0("Plot of the function with maxiter = ",maxiter," and mutprob = ",mutprob))
```

```
return(answer)
}
```

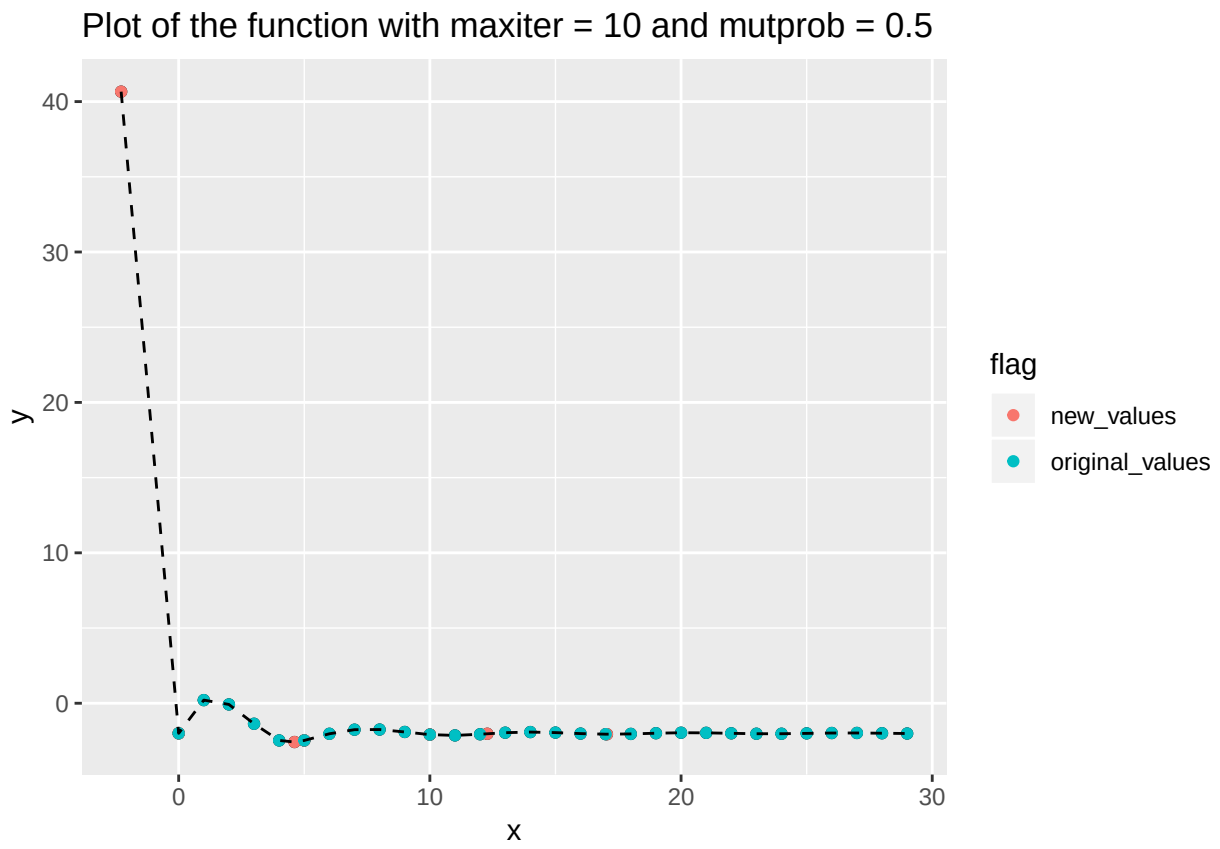
5. Run your code with different combinations of maxiter= 10,100 and mutprob= 0.1,0.5,0.9. Observe the initial population and final population. Conclusions?

```
myfunction(maxiter=10, mutprob=0.1)
```

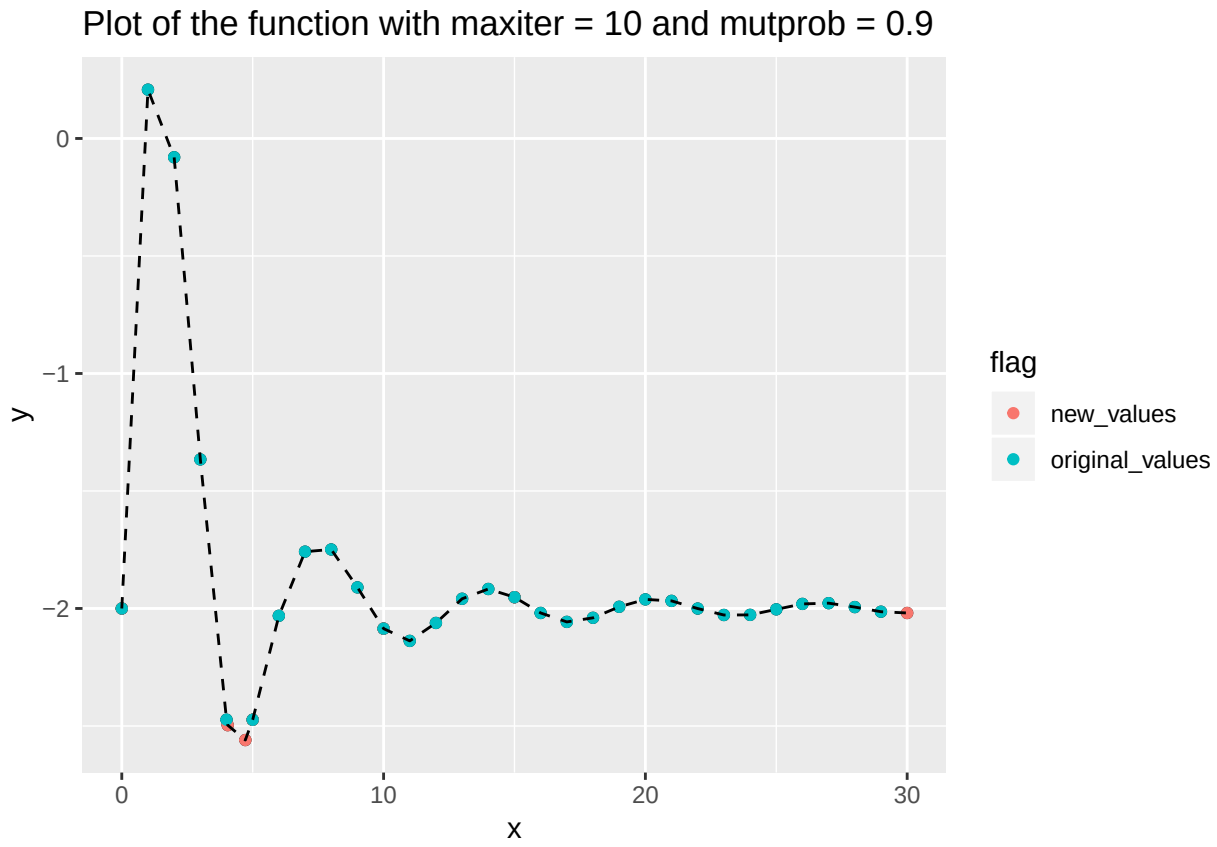
```
406866464688480640648806008040226202486862826400404280600884420086420824226282260662684484066(
```

```
208488282844240820824408004020668606248486468200202640800442260048260462668646680886842242088(
```

```
myfunction(maxiter=10, mutprob=0.5)
```



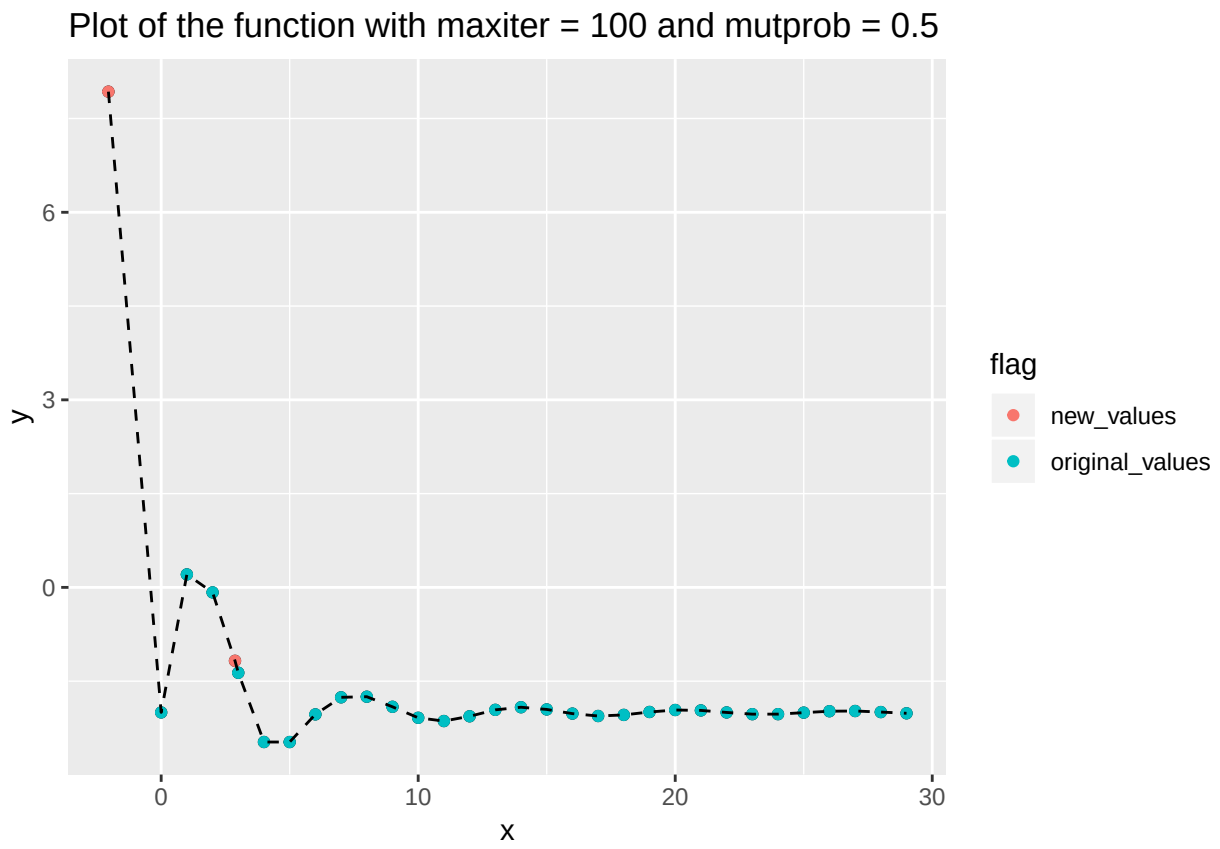

```
myfunction(maxiter=10, mutprob=0.9)
```



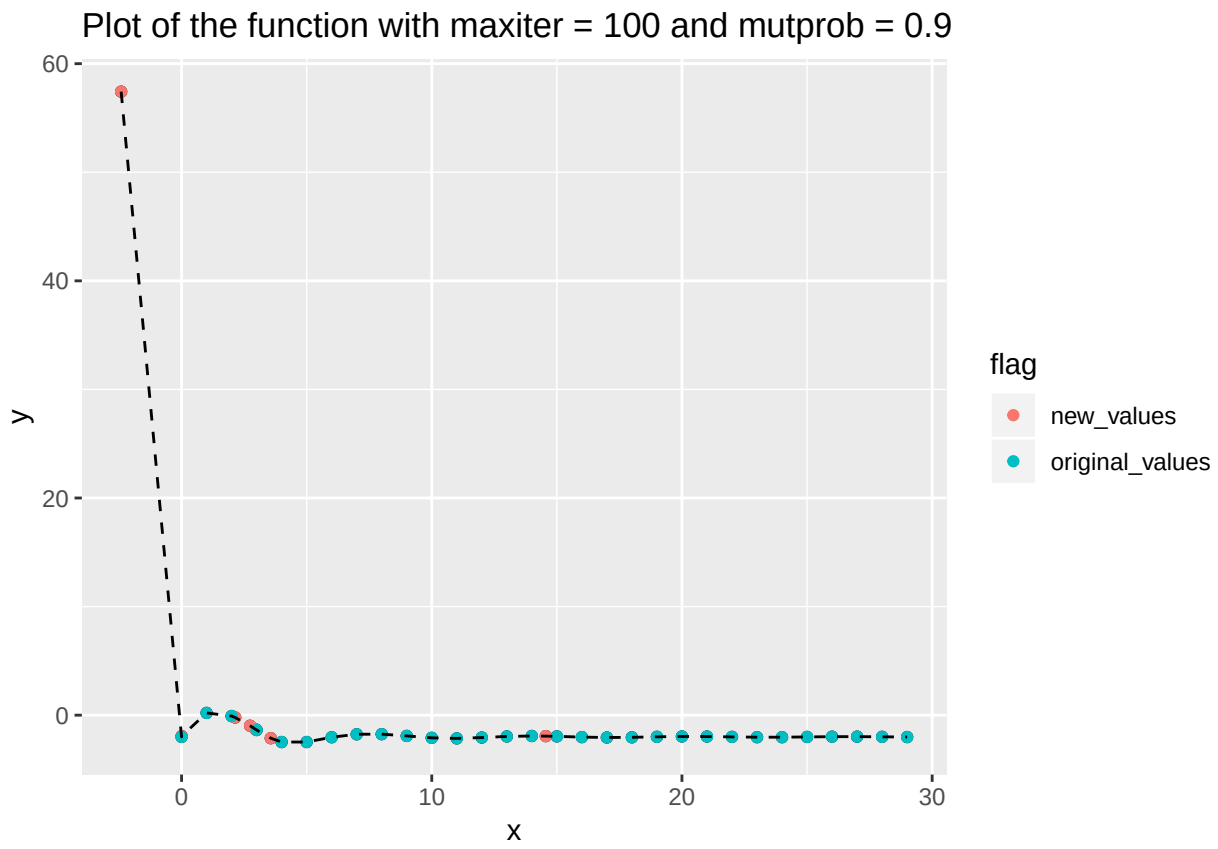

```
myfunction(maxiter=100, mutprob=0.1)
```


```
myfunction(maxiter=100, mutprob=0.5)
```

```
## Warning in mutate(x = kid): probable complete loss of accuracy in modulus

## Warning in mutate(x = kid): probable complete loss of accuracy in modulus

## Warning in mutate(x = kid): probable complete loss of accuracy in modulus
## Warning: Removed 5 rows containing missing values (geom_point).

## Warning: Removed 5 rows containing missing values (geom_point).
## Warning: Removed 5 rows containing missing values (geom_path).
```




```
myfunction(maxiter=100, mutprob=0.9)
```

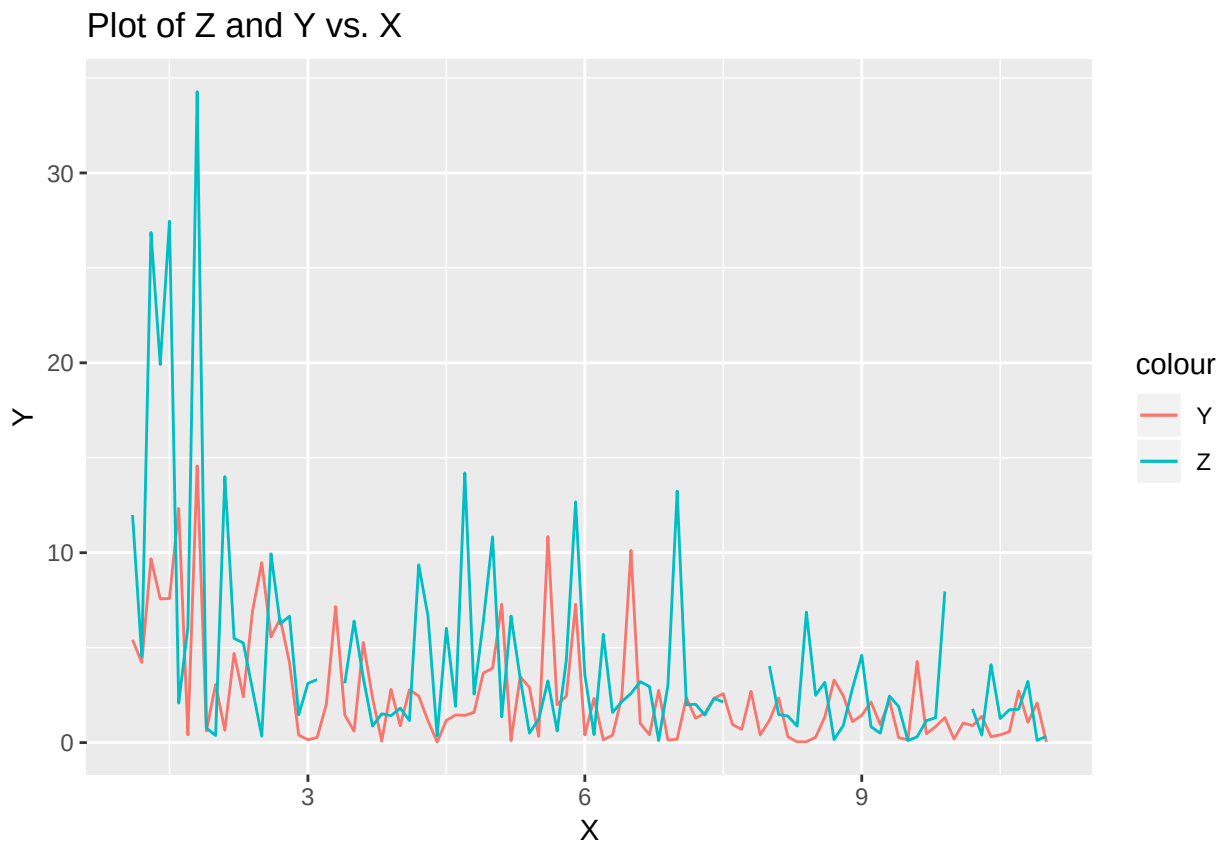


2. EM algorithm

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

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(file="physical1.csv")

ggplot(data=data,aes(x=X, group=1)) +
  geom_line(aes(y = Y, colour = "Y")) +
  geom_line(aes(y = Z, colour = "Z")) +
  ggtitle("Plot of Z and Y vs. X")
```



Analysis: There is a difference in the amplitude of 'Y' and 'Z' with respect to 'X.' Both of the series vary similarily and show similar patterns of decayed oscillation, however 'Z' has faster decay than 'Y'.

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

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

Where λ is an unknown parameters. The goal is to derive the EM algorithm that estimates λ.

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

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}. Here we are assuming there are 'm' missing Z values. λ_k is the lambda value from the previous iteration.

M-step

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

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

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

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

```
my_EM <- 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))
}
```

```
my_EM(data,0.001,50,100)
```

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

Analysis: The result indicates that the optimal lambda is 10.69566 and after 5 iteration are needed for convergence.

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.

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

```
lambda <- 10.69566

new_data <- data
new_data$E_Y <- lambda/data$X
new_data$E_Z <- 2*lambda/data$X

ggplot(data=new_data,aes(x=X, group=1)) +
  geom_line(aes(y = Y, colour = "Y")) +
  geom_line(aes(y = Z, colour = "Z")) +
  geom_line(aes(y = E_Y, colour = "E_Y")) +
  geom_line(aes(y = E_Z, colour = "E_Z")) +
  ggtitle("Plot of Y,Z and their expected value vs. X")
```

Analysis: 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.

Appendix

```
knitr::opts_chunk$set(echo = TRUE)
options(scipen=999)
options(stringsAsFactors = FALSE)
library(dplyr)
library(ggplot2)
library(knitr)
set.seed(12345)

function_f <- function(x){
answer <- x^2/exp(x) - 2*exp(-(9*sin(x))/(x^2+x+1))
return(answer)
}

crossover <- function(x,y){
answer <- (x+y) * 0.5
return(answer)
}

mutate <- function(x){
```

```
answer <- (x^2)%%30
return(answer)
}

y <- vector("double", length = 30)
x <- seq(from=0,to=29,by=1)
for(i in 0:29){y[i+1] <- function_f(x=i)}

data <- data.frame(cbind(x,y))
data$flag <- "original_values"
data$x <- as.numeric(as.character(data$x))
data$y <- as.numeric(as.character(data$y))

ggplot(data, aes(x=x, y=y, group=1)) +
geom_point() +
geom_line(linetype = "dashed") +
ggtitle("Plot of the function")

cat("The maximum values of the function is at x =",which.max(data$y), "and the value is = ",max(data$y))

inital_population <- seq(from=0, to=30, by=5)
inital_population

# generating new values
Values <- vector("double", length = 7)
for(i in seq_along(inital_population)){Values[i] <- function_f(x=inital_population[i])}
original_max <- max(Values)
original_values <- Values
original_populations <- inital_population
original_values
myfunction <- function(maxiter, mutprob){

# maxiter = 10
# mutprob = 0.5


for(i in seq_along(1:maxiter)){
# sampling and getting parent and victim position
parent_index <- sample(x = seq(from = 1, to=7, by=1), size = 2, replace = TRUE)
parent_1 <- Values[parent_index[[1]]]
parent_2 <- Values[parent_index[[2]]]
victim <- which.min(order(Values))

# performing crossover and mutate based on prob
kid <- crossover(x = parent_1, y = parent_2)

if(as.numeric(rbinom(1,1,mutprob))==1){kid <- mutate(x=kid)}
inital_population[victim] <- kid

# generating new values
for(j in seq_along(inital_population)){Values[j] <- function_f(x=inital_population[j])}

}
```

```
new_max <- max(Values)
index_max_val <- inital_population[which.max(Values)]

data2 <- data.frame(cbind(x=inital_population,y=Values))
data2$flag <- "new_values"
data2$x <- as.numeric(as.character(data2$x))
data2$y <- as.numeric(as.character(data2$y))
data3 <- rbind(data2,data)

answer <- ggplot(data3, aes(x=x, y=y, group=1)) +
geom_point() +
geom_point(aes(colour=flag)) +
geom_line(linetype = "dashed") +
ggtitle(paste0("Plot of the function with maxiter = ",maxiter," and mutprob = ",mutprob))
return(answer)
}

myfunction(maxiter=10, mutprob=0.1)
myfunction(maxiter=10, mutprob=0.5)
myfunction(maxiter=10, mutprob=0.9)
myfunction(maxiter=100, mutprob=0.1)
myfunction(maxiter=100, mutprob=0.5)
myfunction(maxiter=100, mutprob=0.9)

data <- read.csv(file="physical1.csv")

ggplot(data=data,aes(x=X, group=1)) +
  geom_line(aes(y = Y, colour = "Y")) +
  geom_line(aes(y = Z, colour = "Z")) +
  ggtitle("Plot of Z and Y vs. X")

my_EM <- 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))
}

my_EM(data,0.001,50,100)
lambda <- 10.69566

new_data <- data
new_data$E_Y <- lambda/data$X
new_data$E_Z <- 2*lambda/data$X

ggplot(data=new_data,aes(x=X, group=1)) +
  geom_line(aes(y = Y, colour = "Y")) +
  geom_line(aes(y = Z, colour = "Z")) +
  geom_line(aes(y = E_Y, colour = "E_Y")) +
  geom_line(aes(y = E_Z, colour = "E_Z")) +
  ggtitle("Plot of Y,Z and their expected value vs. X")
```