Computer Lab 6 Computational Statistics

Phillip Hölscher (phiho267) & Zijie Feng (zijfe244)

1 3 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

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

```
# define the function
func = function(x){
  return((x^2/exp(x)) - 2 * exp(-(9 * sin(x))/ (x^2 +x +1)))
}
```

2. Define the function `crossover()`

for two scalars x and y it returns their "kid as $(x+y)/2$.

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

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

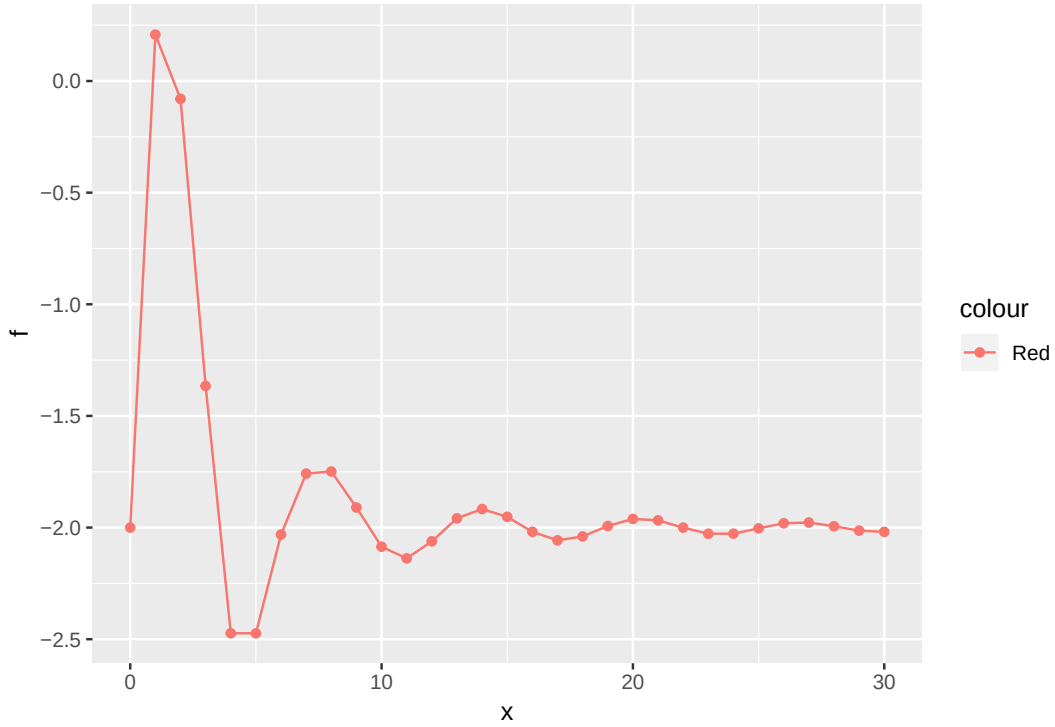
4. Write a function that depends on the parameters `maxiter` and `mutprob` and:

- (a) Plots function f in the range from 0 to 30. Do you see any maximum value?
- (b) Defines an initial population for the genetic algorithm as $X = (0, 5, 10, 15, ..., 30)$.
- (c) Computes vector `Values` that contains the function values for each population point.
- (d) Performs `maxiter` iterations where at each iteration
  - i. Two indexes are randomly sampled from the current population, they are further used as parents (use *sample()*).
  - ii. One index with the smallest objective function is selected from the current population, the point is referred to as victim (use *order()*).
  - iii. Parents are used to produce a new kid by crossover. Mutate this kid with probability *mutprob* (use *crossover()*, *mutate()*).
  - iv. The victim is replaced by the kid in the population and the vector *Values* is updated.

  - v. The current maximal value of the objective function is saved.

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

```
#1.4a#################################################################
dataa <- data.frame(x=0:30,f=func(0:30))
plot1.4=ggplot(dataa,aes(x=x,y=f,color="Red"))+
  geom_line()+
  geom_point()    # max x=1
plot1.4
```



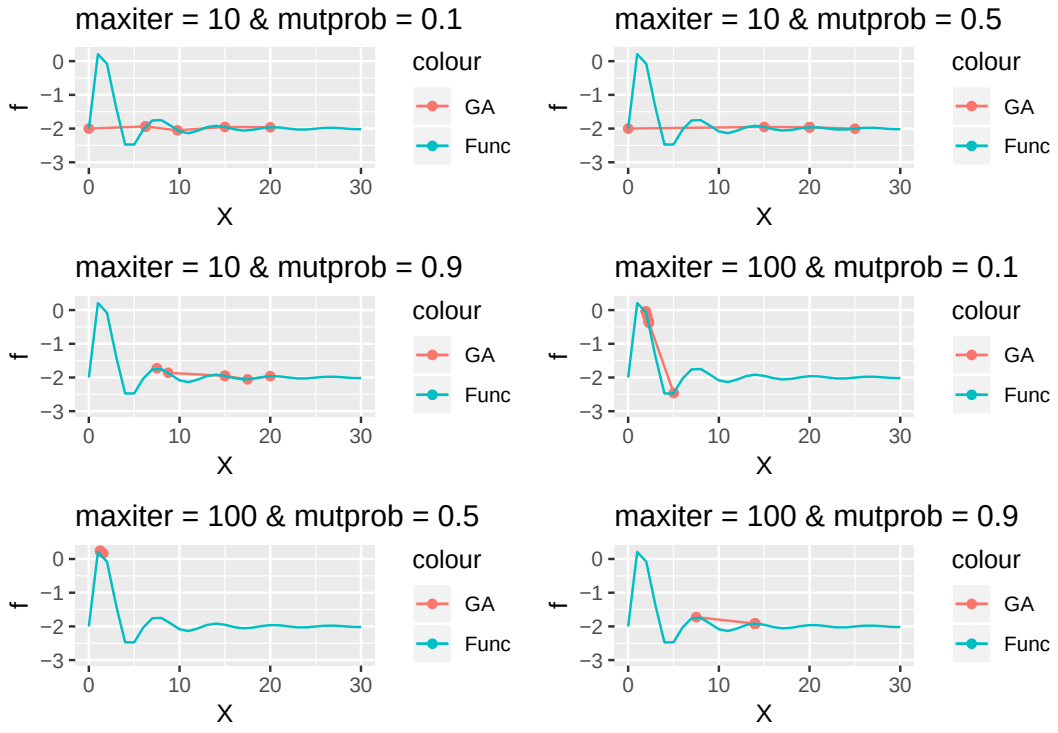

```
#1.4b#################################################################
# initial population
X <- seq(0,30,5)

#1.4c#################################################################
# the function values for each population points
Values <- func(X)
```

```
#1.4d#################################################################
set.seed(1234567)
func4 <- function(pars, animation = F){
  maxiter = pars$maxiter
  mutprob = pars$mutprob
  name = pars$name
  tX <- X
  for(i in 1:maxiter) {
    samples <- sample(tX, 2, replace = F)
    id <- which.min(func(tX))
    kid <- crossover(samples[1],samples[2])
    if(runif(1)>mutprob){
      kid <- mutate(kid)
```

```
    }
    tX[id] <- kid
    tX <- sort(tX)
    if(animation){
      plot(tX,func(tX),type = "b",xlim=c(0,30), ylim = c(-3,0.25),col="Blue")
      lines(x=seq(0,30),y=f(seq(0,30)))
      Sys.sleep(0.2)
    }
  }
  dt <- data.frame(X=tX,f=func(tX))
  pl = ggplot(dt,aes(x=X,y=f,color="Blue"))+
    geom_point()+
    geom_line()+
    geom_line(data=dataa,aes(x=x,y=f,color="Red"))+
    ylim(-3,0.25)+
    xlim(0,30)+
    ggtitle(name)+
    scale_color_discrete(labels=c("GA","Func"))
  return(pl)
}
```

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?

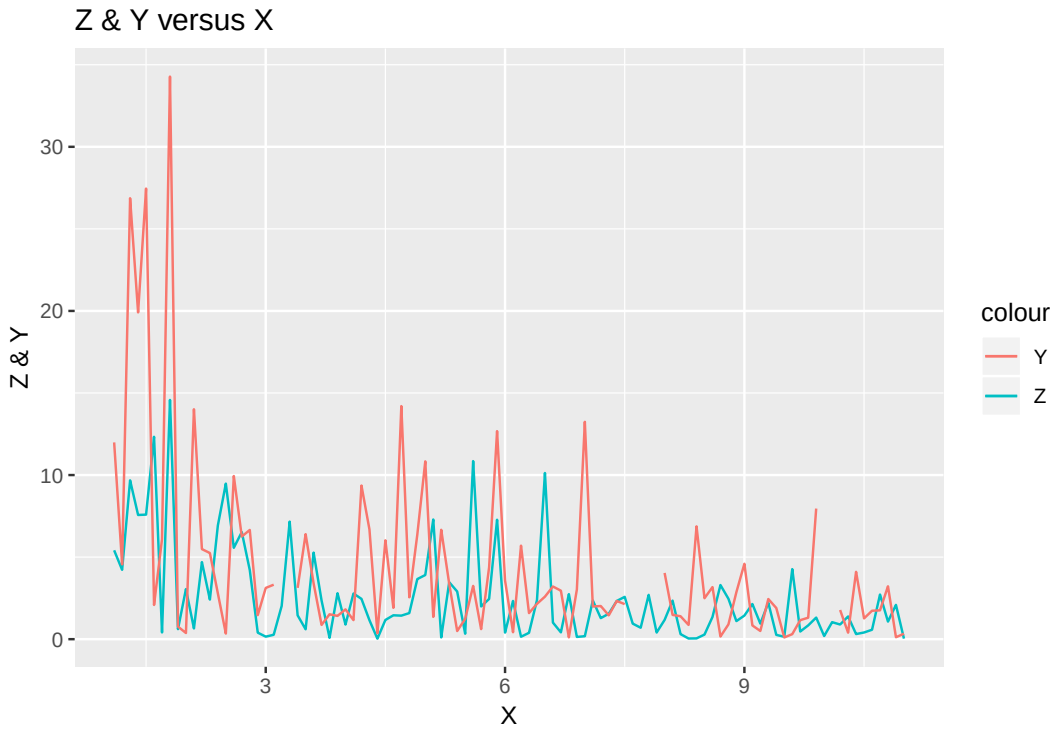


Question 2: EM algorithm

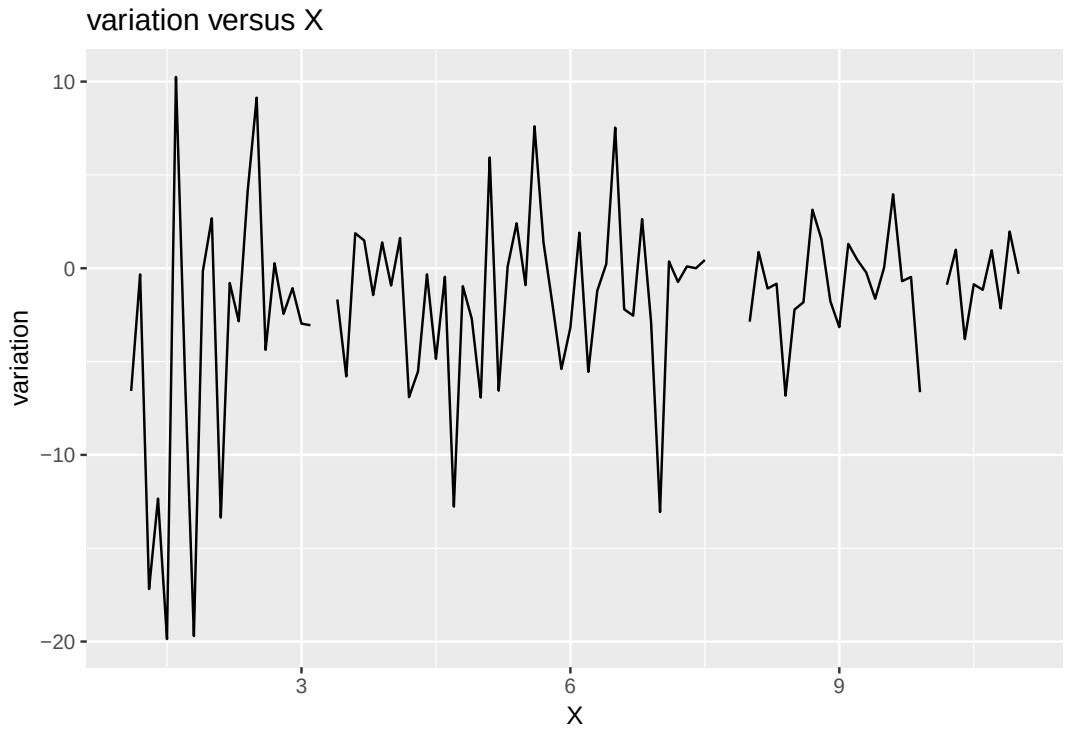
The data file *physical.csv* describes a behavior of two related physical processes $Y = Y(X) and Z = Z(X)$.

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?




```
ggplot(data = data.frame(X=data2$X,variation=data2$Y-data2$Z), aes(x = X)) +
  geom_line(aes(y = variation)) +
  ggtitle("variation versus X")
```



It does not seem the two processes are related to each other. In the beginning, the Z value does sleep off more than double of Y. Also, the movements of the processes rarely lie on top of each other. We can also recognize the Z values are incomplete in some parts, the curve has gaps.

- variation of the response values with respect to X?

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

$$Y_i \sim \exp(X_i/\lambda), \quad Z_i \sim \exp(X_i/2\lambda)$$

where λ is some unknown parameter. *The goal is to derive an EM algorithm that estimates* λ.

We know the probability density functions (PDF) of Z are

$$f(Y_i) = \frac{X_i}{\lambda} e^{-\frac{X_i}{\lambda} Y_i}, f(Z_i) = \frac{X_i}{2\lambda} e^{-\frac{X_i}{2\lambda} Z_i}.$$

Since we consider that all the X_i are independent and we can thereby get the PDF of a joint distribution, which is

$$\mathcal{L}(\lambda|Y,Z) = \prod (\frac{X_i}{\lambda} e^{-\frac{X_i}{\lambda} Y_i}) \cdot \prod (\frac{X_i}{2\lambda} e^{-\frac{X_i}{2\lambda} Z_i})$$

$$= \frac{\prod X_i}{\lambda^n} e^{-\frac{\sum X_i Y_i}{\lambda}} \cdot \frac{\prod X_i}{2^n \lambda^n} e^{-\frac{\sum X_i Z_i}{2\lambda}} .$$

Thus, the log-likelihood should be

$$\updownarrow(\lambda|Y,Z) = \prod (\ln X_i) - n \ln \lambda - \frac{\sum X_i Y_i}{\lambda} + \prod (\ln X_i) - n \ln(2\lambda) - \frac{1}{2\lambda} \sum X_i Z_i.$$

Expectation

Estimating the expected value for the miss data by the other data provided in last iterations. Additionally, we replace the miss data with the expected value of Z.

$$Q(\lambda, \lambda_k) = E[\updownarrow(\lambda|Y, Z)|\lambda_k, Y, Z]$$

$$= \prod(\ln X_i) - n\ln\lambda_k - \frac{\sum X_i Y_i}{\lambda_k} + \prod(\ln X_i) - n\ln(2\lambda_k) - \frac{1}{2\lambda_k}[\sum_{obs} X_i Z_i + \sum_{miss} X_i \frac{2\lambda_{k-1}}{X_i}]$$

$$= 2\prod(\ln X_i) - n\ln\lambda_k - \frac{\sum X_i Y_i}{\lambda_k} - n\ln(2\lambda_k) - \frac{1}{2\lambda_k}\sum_{obs} X_i Z_i - \frac{1}{\lambda_k}\sum_{miss}\lambda_{k-1}.$$

maximization

To obtain the maximum likelihood estimate of λ, we need to solve that the partial erivative equals 0.

$$\nabla\lambda = -\frac{2n}{\lambda_k} + \frac{\sum X_i Y_i}{\lambda_k^2} + \frac{1}{2\lambda_k^2}\sum_{obs} X_i Z_i + \frac{1}{\lambda_k^2}\sum_{miss}\lambda_{k-1} = 0$$

$$\lambda = \frac{1}{4n}(2\sum X_i Y_i + \sum_{obs} X_i Z_i + 2\sum_{miss}\lambda_{k-1})$$

3.Implement this algorithm in R

use $\lambda_0 = 100$ 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?

```
EM <- function(data2, eps, kmax=500, lda){
  X <- data2$X
  Y <- data2$Y
  Z <- data2$Z
  obs <- !is.na(Z)
  n <- length(X)
  m <- sum(is.na(Z))
  # browser()
  loglik <- function(lda,ldap){
    return(2*sum(log(X))-n*log(lda)-(X%*%Y)/lda-n*log(2*lda)
           -(X[obs]%*%Z[obs])/(2*lda)-m*ldap/lda)
  }
  # initial state
  k <- 0
  llvalprev <- lda*10+10  # make some difference
  llvalcurr <- lda
  llk <- loglik(llvalcurr,llvalprev)
  print(c(k, llvalcurr,llk))

  while ((abs(llvalprev-llvalcurr)>eps) && (k<(kmax+1))){
    llvalprev <- llvalcurr
    # since we already know how to get argmax lda
```

```
    llvalcurr <- 1/(4*n)*(2*X%*%Y+X[obs]%*%Z[obs]+2*m*llvalprev)

    k<-k+1
    llk <- loglik(llvalcurr,llvalprev)
    print(c(k, llvalcurr,llk))
  }
  return(llvalcurr)
}

lda_opt <- EM(data2, eps=0.001, lda=100)
```

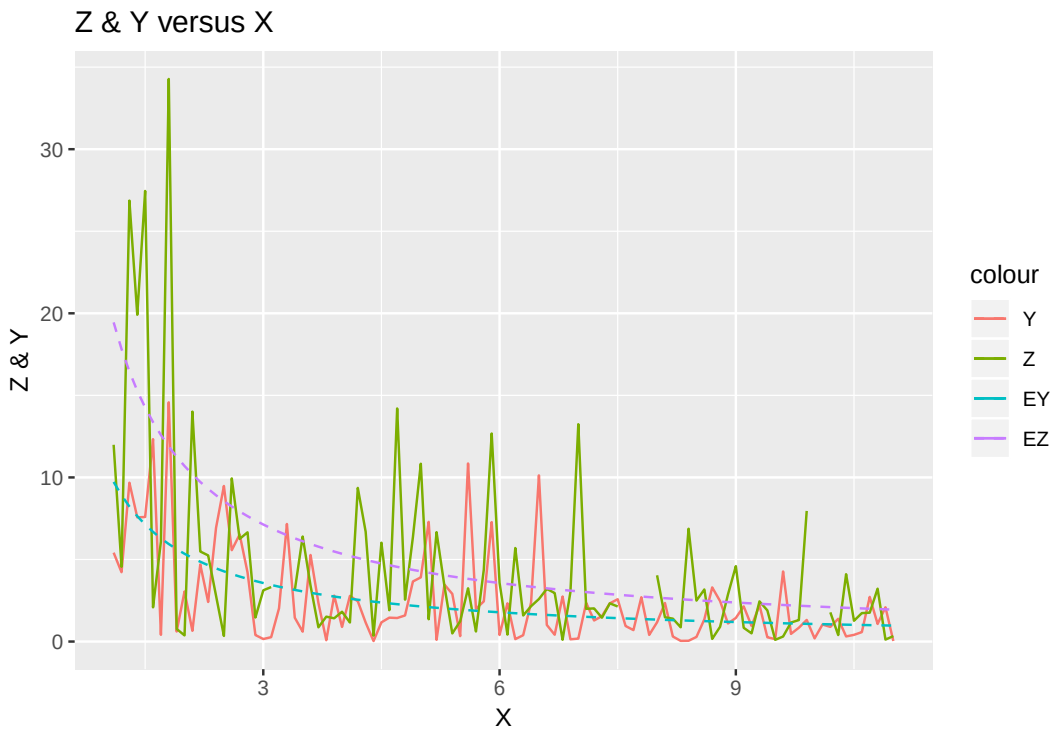
```
## [1]    0.0000  100.0000 -761.7647
## [1]    1.00000   14.26782 -470.99635
## [1]    2.00000   10.83853 -416.01655
## [1]    3.00000   10.70136 -413.46921
## [1]    4.00000   10.69587 -413.36664
## [1]    5.00000   10.69566 -413.36253
```

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.

```
X <- data2$X
EY <- c(lda_opt)/X
EZ <- 2*EY

data24 <- cbind(data2,data.frame(EY=EY,EZ=EZ))
ggplot(data = data24) +
  geom_line(aes(x = X, y=Y, color="1")) +
  geom_line(aes(x = X, y=Z, color="2")) +
  geom_line(aes(x = X, y=EY, color="3"),linetype="dashed") +
  geom_line(aes(x = X, y=EZ, color="4"),linetype="dashed") +
  ggtitle("Z & Y versus X") +
  ylab("Z & Y")+
  scale_color_discrete(labels=c("Y","Z","EY","EZ"))
```


Z & Y versus X
30
20
10
0
Z & Y
3
6
9
X
colour
Y
Z
EY
EZ

Appendix

```
knitr::opts_chunk$set(echo = TRUE, out.height = "280px")
# library used in this lab
library(ggplot2) # ex 2.1 - time series plot
library(gridExtra)
# clean the environment
rm(list=ls())
# define the function
func = function(x){
  return((x^2/exp(x)) - 2 * exp(-(9 * sin(x))/ (x^2 +x +1)))
}
# crossover function
crossover = function(x,y){
  kid = (x+y)/2
  return(kid)
}
# mutate function
mutate = function(x){
  return(x^2 %%30)
}
#1.4a#################################################################
dataa <- data.frame(x=0:30,f=func(0:30))
plot1.4=ggplot(dataa,aes(x=x,y=f,color="Red"))+
  geom_line()+
  geom_point()    # max x=1
plot1.4
#1.4b#################################################################
# initial population
X <- seq(0,30,5)

#1.4c#################################################################
# the function values for each population points
Values <- func(X)
#1.4d#################################################################
set.seed(1234567)
func4 <- function(pars, animation = F){
  maxiter = pars$maxiter
  mutprob = pars$mutprob
  name = pars$name
  tX <- X
  for(i in 1:maxiter) {
    samples <- sample(tX, 2, replace = F)
    id <- which.min(func(tX))
    kid <- crossover(samples[1],samples[2])
    if(runif(1)>mutprob){
      kid <- mutate(kid)
    }
    tX[id] <- kid
    tX <- sort(tX)
    if(animation){
      plot(tX,func(tX),type = "b",xlim=c(0,30), ylim = c(-3,0.25),col="Blue")
      lines(x=seq(0,30),y=f(seq(0,30)))
```

```
      Sys.sleep(0.2)
    }
  }
  dt <- data.frame(X=tX,f=func(tX))
  pl = ggplot(dt,aes(x=X,y=f,color="Blue"))+
    geom_point()+
    geom_line()+
    geom_line(data=dataa,aes(x=x,y=f,color="Red"))+
    ylim(-3,0.25)+
    xlim(0,30)+
    ggtitle(name)+
    scale_color_discrete(labels=c("GA","Func"))
  return(pl)
}
maxiter = c(10,100)
mutprob = c(0.1,0.5,0.9)
names = c("maxiter = 10 & mutprob = 0.1",
          "maxiter = 100 & mutprob = 0.5",
          "maxiter = 10 & mutprob = 0.9",
          "maxiter = 100 & mutprob = 0.1",
          "maxiter = 10 & mutprob = 0.5",
          "maxiter = 100 & mutprob = 0.9")
pairs = data.frame(maxiter=rep(maxiter,3),mutprob=rep(mutprob,2),name=names)
pairs = split(pairs,pairs[,3])

plot(arrangeGrob(grobs=lapply(t(pairs), func4)))
# clean the environment
rm(list=ls())
# load the data
data2 = read.csv("physical1.csv")
# Z & Y versus X
col = c("Y" = "#F8766D", "Z" = "#619CFF")

plot21 <- ggplot(data = data2) +
  geom_line(aes(x = X, y=Y,color = "Red")) +
  geom_line(aes(x = X, y=Z,color = "Blue")) +
  ggtitle("Z & Y versus X") +
  ylab("Z & Y")+
  scale_color_discrete(labels=c("Y","Z"))
plot21
ggplot(data = data.frame(X=data2$X,variation=data2$Y-data2$Z), aes(x = X)) +
  geom_line(aes(y = variation)) +
  ggtitle("variation versus X")
EM <- function(data2, eps, kmax=500, lda){
  X <- data2$X
  Y <- data2$Y
  Z <- data2$Z
  obs <- !is.na(Z)
  n <- length(X)
  m <- sum(is.na(Z))
  # browser()
  loglik <- function(lda,ldap){
    return(2*sum(log(X))-n*log(lda)-(X%*%Y)/lda-n*log(2*lda)
```

```
          -(X[obs]%*%Z[obs])/(2*lda)-m*ldap/lda)
  }
  # initial state
  k <- 0
  llvalprev <- lda*10+10  # make some difference
  llvalcurr <- lda
  llk <- loglik(llvalcurr,llvalprev)
  print(c(k, llvalcurr,llk))

  while ((abs(llvalprev-llvalcurr)>eps) && (k<(kmax+1))){
    llvalprev <- llvalcurr
    # since we already know how to get argmax lda
    llvalcurr <- 1/(4*n)*(2*X%*%Y+X[obs]%*%Z[obs]+2*m*llvalprev)

    k<-k+1
    llk <- loglik(llvalcurr,llvalprev)
    print(c(k, llvalcurr,llk))
  }
  return(llvalcurr)
}

lda_opt <- EM(data2, eps=0.001, lda=100)
X <- data2$X
EY <- c(lda_opt)/X
EZ <- 2*EY

data24 <- cbind(data2,data.frame(EY=EY,EZ=EZ))
ggplot(data = data24) +
  geom_line(aes(x = X, y=Y, color="1")) +
  geom_line(aes(x = X, y=Z, color="2")) +
  geom_line(aes(x = X, y=EY, color="3"),linetype="dashed") +
  geom_line(aes(x = X, y=EZ, color="4"),linetype="dashed") +
  ggtitle("Z & Y versus X") +
  ylab("Z & Y")+
  scale_color_discrete(labels=c("Y","Z","EY","EZ"))
```