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)
}
```

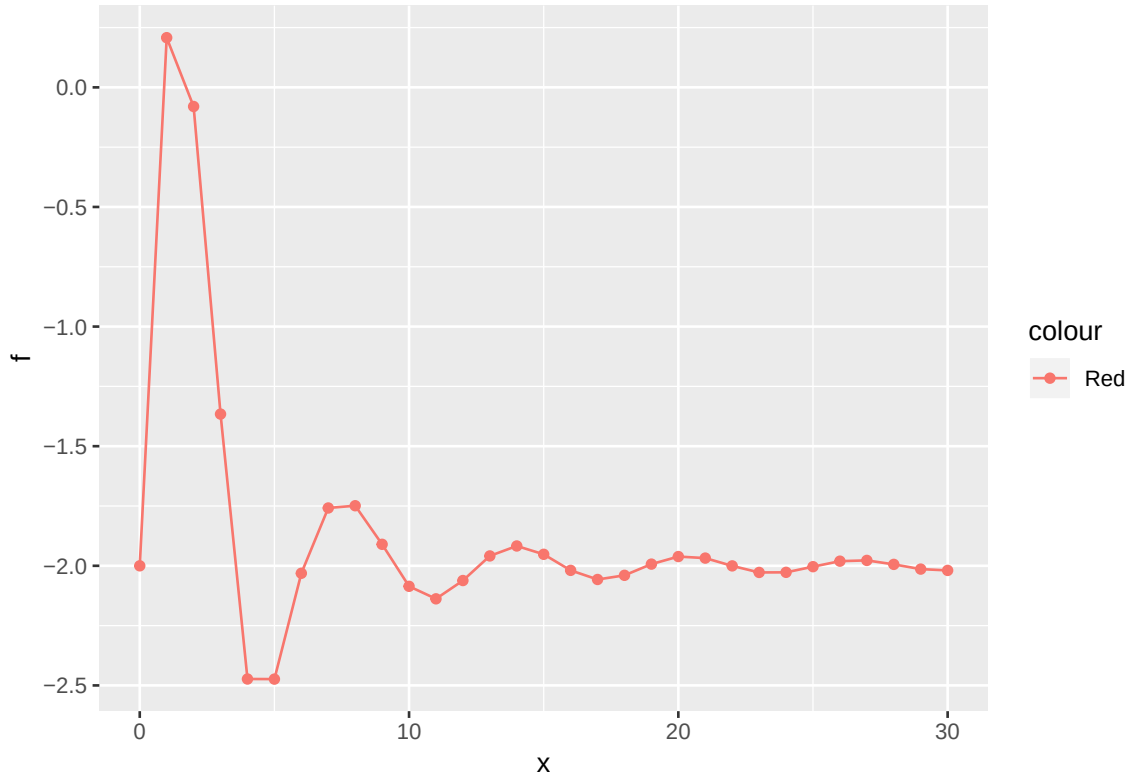
4. Write a function that depends on the parameters `maxiter` and `mutprob` and:

- (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
```



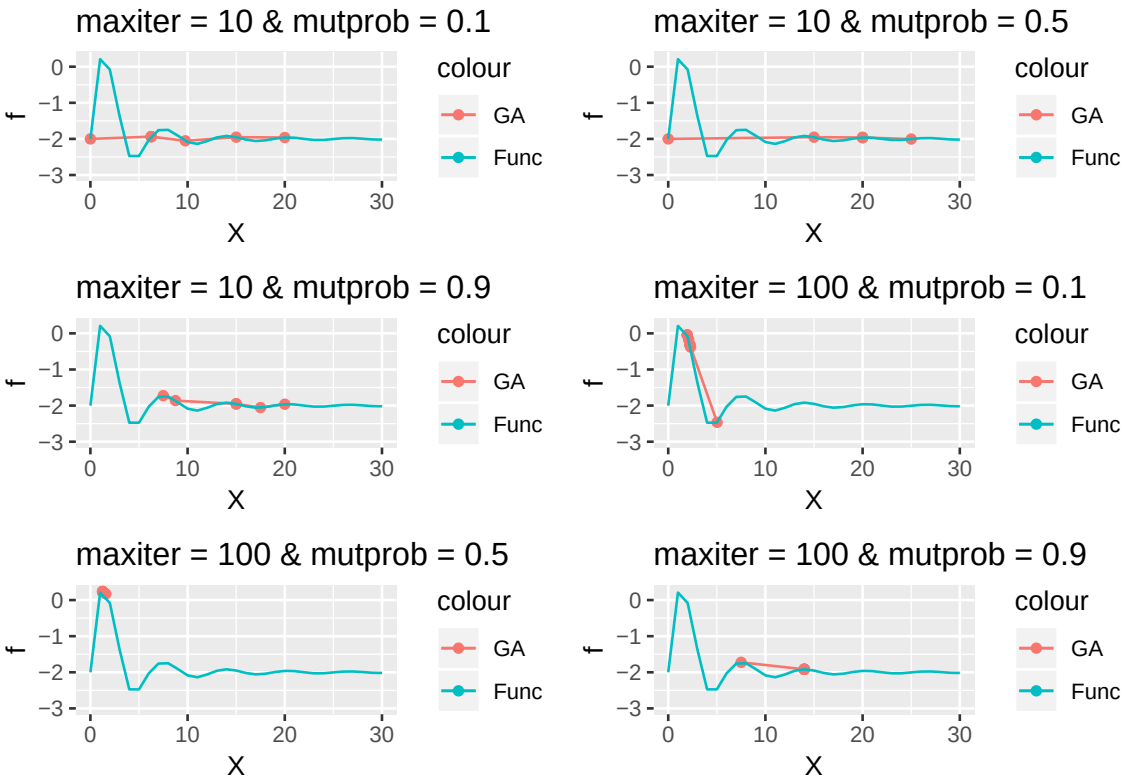

```
#1.4b#################################################################
# initial population
X <- seq(0,30,5)

#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?

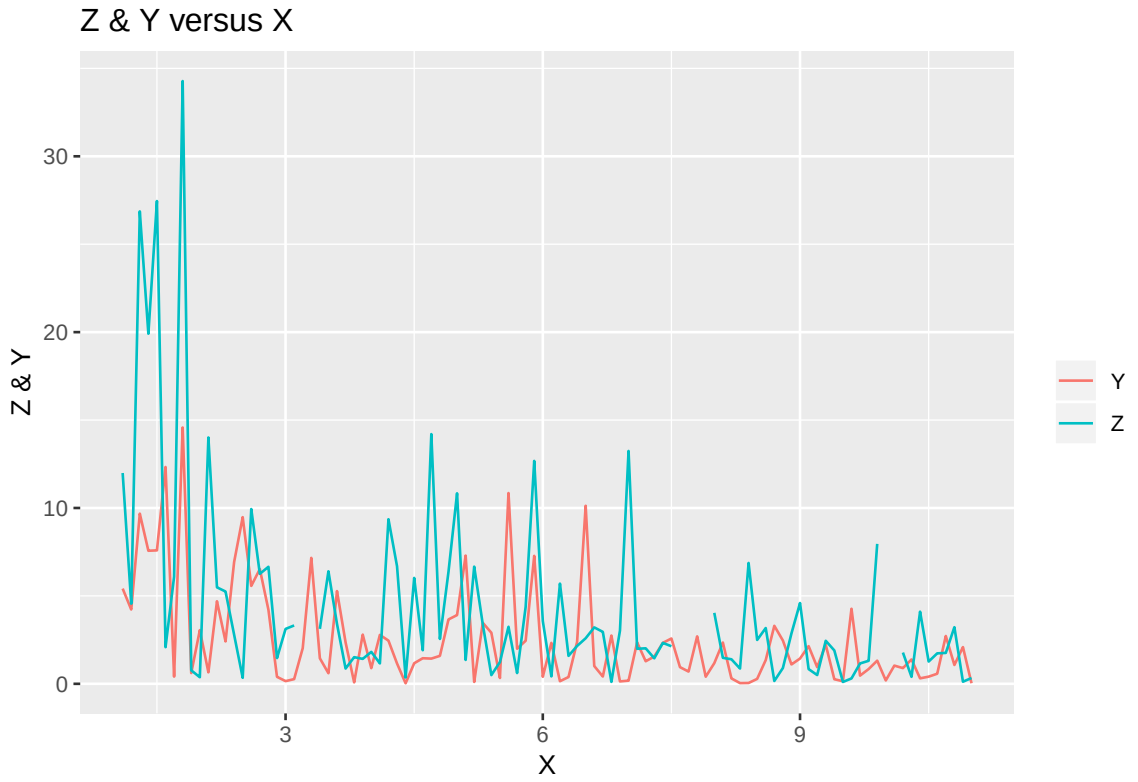


#Question 2: EM algorithm

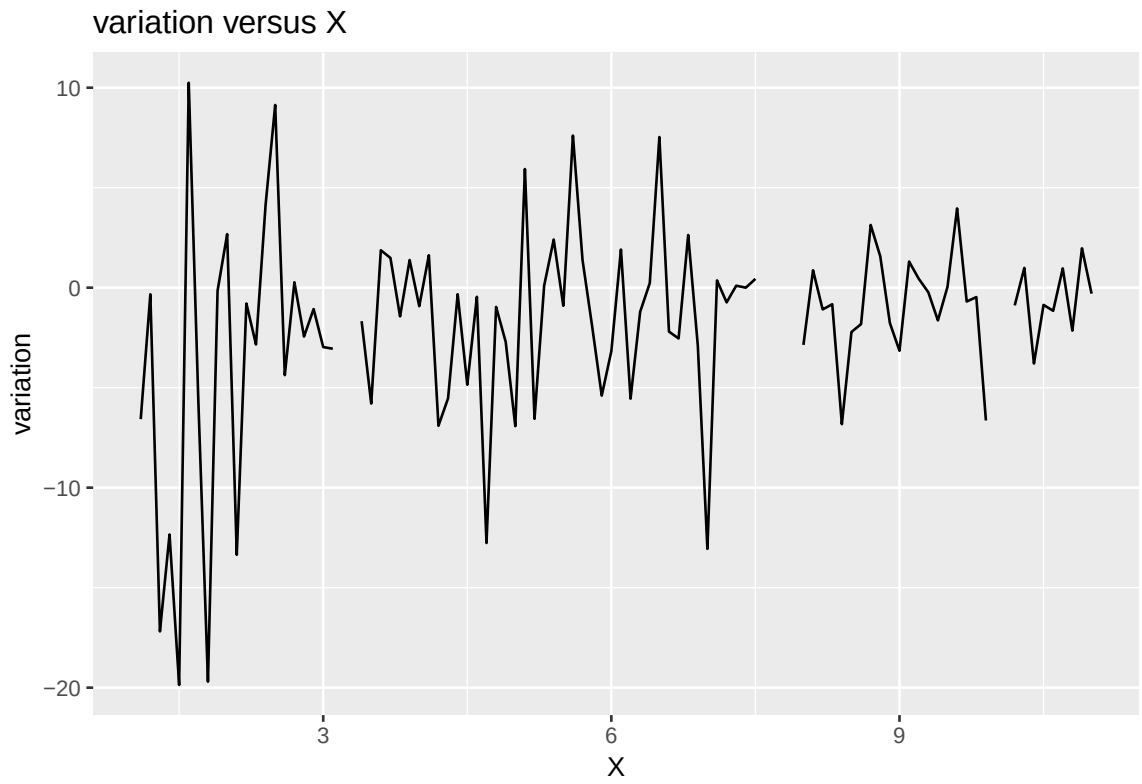
The data file *physical.csv* describes a behavior of two related physical processes $Y = Y(X) and Z = Z(X)$.

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

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

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

Appendix

```
knitr::opts_chunk$set(echo = TRUE, out.height = "300px")
# 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" = "#FFC312", "Z" = "#0652DD")

data21 = reshape2::melt(data2, id.vars="X")
ggplot(data = data21, aes(x = X,y=value,color=variable)) +
  geom_line() +
  ggtitle("Z & Y versus X") +
  ylab("Z & Y")+
  scale_color_discrete("")

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