Assigment 1: Genetic algorithm

1

Define the function $f(x)$:

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

2

Define the `crossover()` function:

```
crossover<-function(x,y) sum(x,y)/2
```

3

Define the `mutate()` function:

```
mutate<-function(x) x^2%%30
```

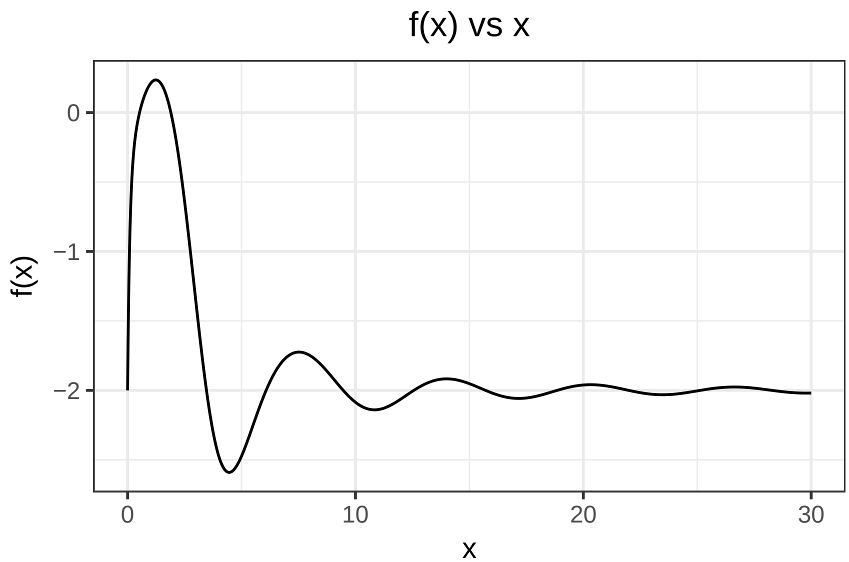
4

Write a function that use $f(x)$, `crossover()` and `mutate()` to make a genetic algorithm. The function takes the arguments `maxiter` and `mutprob` that determines how many interaction and how likely that is that the new observation will mutate.

a

But we start to plot the function that we want to maximize.

```
x<-seq(0,30,0.01)
values<-f_x(x)
  ggplot(mapping=aes(x=x,y=values))+
    geom_line()+
    theme_bw()+
    labs(y="f(x)",title="f(x) vs x")+
    theme(plot.title = element_text(hjust = 0.5))
```




```
values[which.max(values)] #max f(x)
```

```
## [1] 0.2349
```

```
x[which.max(values)]      #the x that maximize f(x)
```

```
## [1] 1.24
```

We can see that the x that maximize f(x) is 1.24 which gives us f(1.24)=0.235.

b:e

```
my_fun<-function(maxiter=100,mutprob=0.6){
  x<-seq(0,30,5)             #Set the start pop
  x_start<-x

  count_mutate<-0            #Just for count the numer of mutate
  n<-length(x)               #The length of the population


  for (i in 1:maxiter){      #For over the length of maxiter

    index<-sample(1:n,2)     #select th index of the parents
    parents<-x[index]        #Extract x from parents

    values<-f_x(x)           #calc f_x on the pop
    x<-x[-order(values)[1]] #remove the x that have samllest f_x


    kid<-crossover(parents[1],parents[2]) #Produce a kid with the selected parents
                                          #We assume that the samllest f_x still
                                          #can produce a kid even if we have remove
                                          #it from the pop


    if(runif(1)<mutprob){             #Mutate or not?
```

```r
      kid<-mutate(kid)               #Mutate the kid
      count_mutate<-count_mutate+1   #Count how many times we have mutate a kid
    }


    x<-c(x,kid)                    #Get our new pop
    values<-f_x(x)                 #Whats the f_x on our new pop
  }


  #Make a list that the function return
  svar<-list(x=x,values=values,maxiter=maxiter,mutprob=mutprob,
             count_mutate=count_mutate,x_start=x_start)
  return(svar)
}
```

`plota_fun()` is a function that takes the output from the function `my_fun()` and make a nice plot.

```r
plota_fun<-function(svar){

  text_nr1<-paste("Maxiter =",svar$maxiter,"\n Mutprob =",svar$mutprob,
                  "\n Number of mutates =",svar$count_mutate,
                  "\n max(f(x)) =",max(round(svar$values,2)))

  x<-seq(0,30,0.01)
  values<-f_x(x)

  ggplot(mapping=aes(x=x,y=values))+
    geom_line()+
    theme_bw()+
    labs(y="f(x)",title="f(x) vs x")+
    theme(plot.title = element_text(hjust = 0.5))+
    geom_point(aes(x=svar$x_start,y=f_x(svar$x_start)),alpha=0.7,col="dodgerblue3",size=2)+
    geom_point(aes(x=svar$x[-length(svar$x)],y=svar$values[-length(svar$x)]),alpha=0.3,
               col="black",size=1.5)+
    geom_point(aes(x=svar$x[length(svar$x)],y=svar$values[length(svar$x)]),alpha=0.7,
               col="red",size=2)+
    annotate("text",svar$x[length(svar$x)],svar$values[length(svar$x)]-0.1,
             label=paste("Last one"))+
    annotate("text",20,-0.3,label=text_nr1)
}
```

5

Now when we have the functions can we use them. We will set maxiter$=10,\ 100$ and mutprob$=0.1,\ 0.5,\ 0.9$

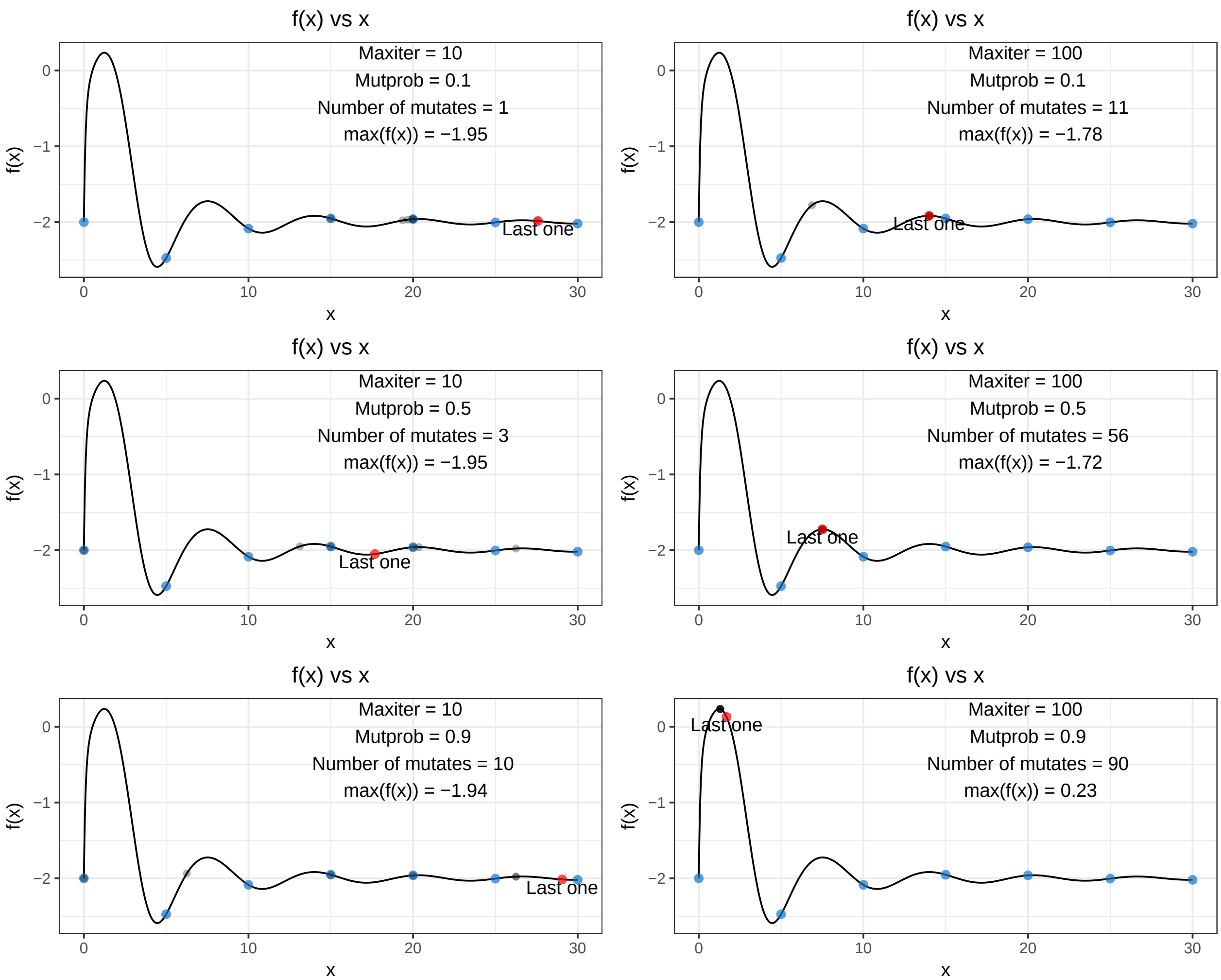
```r
maxiter<-c(10,100)      #All the maxiter
mutprob<-c(0.1,0.5,0.9) #All the mutprob
```

```r
plot_list<-list() #List to save the plots

set.seed(3456789)                    #Set a seed
for(i in maxiter){                  #For over all maxiter
  for(j in mutprob){                #For over all mutprob
    k<-length(plot_list)+1          #Index for save the plot
    plot_list[[k]]<-plota_fun(my_fun(i,j))  #The return is a plot
  }
}

lay <- rbind(c(1,4),   #The layot of the plots
             c(2,5),
             c(3,6))

#Plot all the graphs
grid.arrange(
  plot_list[[1]],
  plot_list[[2]],
  plot_list[[3]],
  plot_list[[4]],
  plot_list[[5]],
  plot_list[[6]],
  layout_matrix = lay)
```



On the left side we use maxiter=10 and mutprob=0.1, 0.5 and 0.9. While one the right side we use maxiter=100 and mutprob=0.1, 0.5 and 0.9.

We can see that when we use maxiter=10 the algorithm dont converge and dont find the a good x that maximize f(x) in any of the cases. The conclusion is that maxiter=10 is to small.

When we use maxiter=100 insted the algorithm find a local or global maximum in all cases. When we use a higher value of mutprob we get a higher chance to find the global maximum.

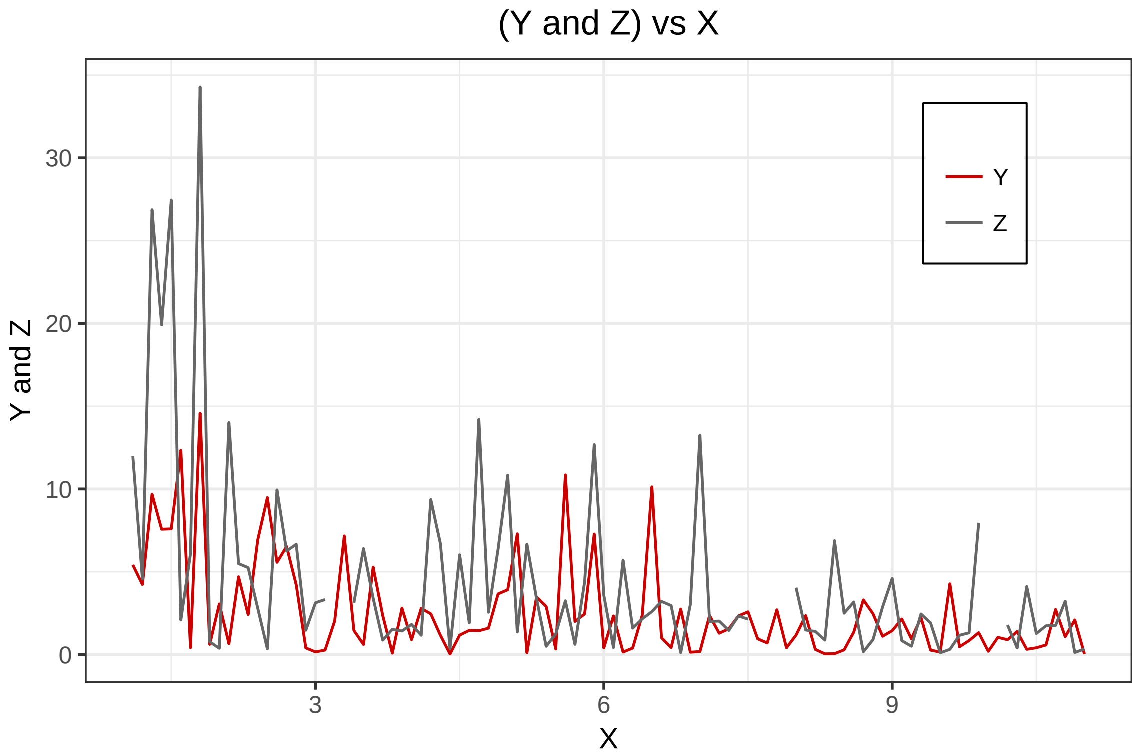
Assigment 2: EM algorithm

1

For this task we will make a time series plot describing dependence of Z and Y versus X.

```
ggplot(physical1,aes(x=X))+
  geom_line(aes(y=Y,col="Y"))+
  geom_line(aes(y=Z,col="Z"))+
```

```r
  theme_bw()+
  labs(y="Y and Z",title="(Y and Z) vs X")+
  theme(legend.position = c(0.85, 0.80),
        legend.box.background = element_rect(),
        legend.box.margin = margin(1, 1, 1, 1))+
  theme(plot.title = element_text(hjust = 0.5))+
  scale_colour_manual(name="",
                      breaks=c("Y", "Z"),
                      labels=c("Y", "Z"),values=c("red3","grey40"))
```



We see that both Y and Z have higher values when X is low then they have when X is high.

2

For this task we will derive the EM algorithm that esimates λ. We have the distubutions:

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

These two distubutions come from the exponential distribution:

$$\lambda e^{-\lambda x_i}$$

From this we need to cacluate the joint probability density between Y_i and Z_i:

$$L(\lambda|X,Y,Z) = \left(\frac{1}{2\lambda^2}\right)^n \prod x_i^2 \cdot exp\left(\frac{1}{\lambda}\sum x_i y_i + \frac{1}{2\lambda}\sum x_i z_i\right)$$

Performing the logarithm on this will yield:

$$log(L(\lambda|X,Y,Z)) = l(\lambda|X,Y,Z) = -n\ log(2) - 2n\ log(\lambda) + 2\sum log(x_i) + \frac{1}{\lambda}\sum x_i y_i + \frac{1}{2\lambda}\sum x_i z_i$$

For the next steps we need to consider that we have n observations in data and we have some amount of observed data $1, ..., r$ and unobserved values $(r+1), ..., n$. The unobserved data are the missing values in vaiable Z.

We will devide the values in Z in two groups, one group that is observed (not missing) and one that is unobserved (missing values). We give the observed values the name w_i and the unobserved values the name v_i so that $Z_i \ni \{w_i, v_i\}$. We then take the expected value of $log(L)$ given all observed values:

$$E[l(\lambda|X,Y,\underbrace{W,V}_{Z}) \mid X,Y,W] = -n\ log(2) - 2n\ log(\lambda) + 2\sum log(x_i) + \frac{1}{\lambda}\sum x_i y_i + \frac{1}{2\lambda}\sum E[x_i z_i]$$

We know that the expected value of a exponential distribution is $1/\lambda$ and therefore can compute the expected value of Z:

$$E[v_i] = \frac{2\lambda_k}{x_i} \quad \rightarrow \quad E\Big[\sum x_i z_i\Big] = E\Big[\sum x_i w_i + \sum x_i v_i\Big] = \sum_{i=1}^{r} x_i w_i + \underbrace{\sum_{i=r+1}^{n} x_i \frac{2\lambda_k}{x_i}}_{(n-r)2\lambda_k}$$

Putting this in to the log-likelihood will give us:

$$E[l(\lambda|X,Y,W,V) \mid X,Y,W] = -n\ log(2) - 2n\ log(\lambda) + 2\sum log(x_i) + \frac{1}{\lambda}\sum x_i y_i + \frac{1}{2\lambda}\left(\sum_{i=1}^{r} x_i w_i + (n-r)2\lambda_k\right)$$

To find lambda we will derive the $E[l(\lambda|X,Y,W,V) \mid X,Y,W]$ and set it to 0:

$$\frac{\partial\ log(L)}{\partial\lambda} = -\frac{2n}{\lambda} + \frac{1}{\lambda^2}\sum x_i y_i + \frac{1}{2\lambda^2}\left(\sum_{i=1}^{r} x_i w_i + (n-r)2\lambda_k\right) = 0$$

Solving λ will give:

$$\lambda = \frac{\sum_1^n x_i y_i}{2n} + \frac{\sum_1^r x_i w_i}{4n} + \frac{(n-r)\cdot 2\lambda_k}{4n}$$

We are also little curious about what we can expect, so we will plot the $l(\lambda|X,Y,W)$ when we have removed the missing values(V) from the log-likelihood. The log-liklihood will then look like this:

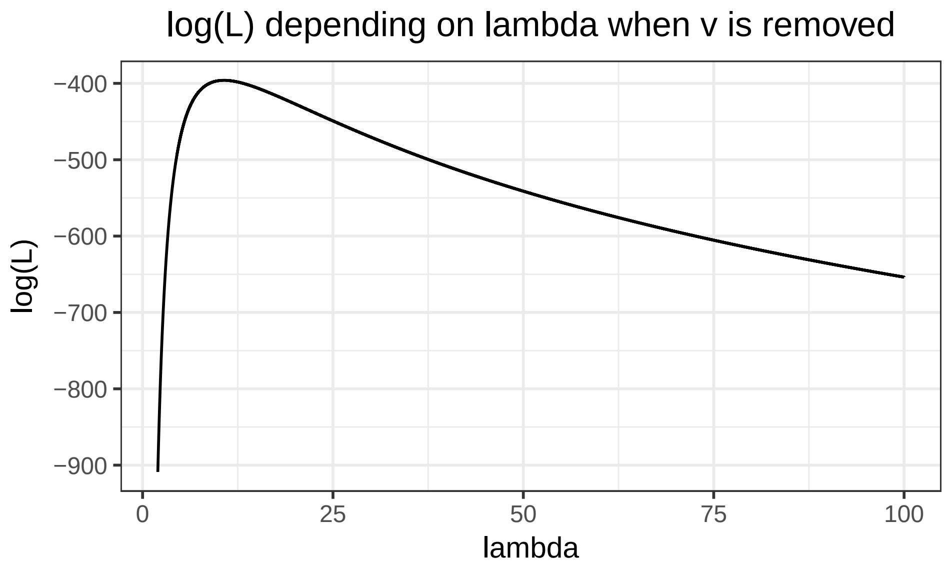
$$l(\lambda|X,Y,W) = \sum_1^n log(x_i) + \sum_1^r log(x_i) - n\ log(\lambda) - r\ log(2\lambda) - \frac{1}{\lambda}\sum_1^n x_i y_i - \frac{1}{2\lambda}\sum_1^r x_i w_i$$

```r
#Make a function that calc the log-likelihood when v is removed
log_like_fun<-function(x,y,z,lambda){

  i<-!is.na(z)                        #Get the index where z is not NA
                                      #We can then get the w values

  n<-length(x)                        #get the n
  r<-sum(i)                           #get the r

  sum(log(x))+                        #Here is the log-likelihood.
    sum(log(x[i]))-                   #Look at the formula just above
    n*log(lambda)-
    r*log(2*lambda)-
    (1/lambda)*sum(x*y)-
    1/(2*lambda)*sum(x[i]*z[i])
}
```



Here is a plot of $l(\lambda|X,Y,W)$. V contains just 8 values, so we would expect us something like this. The maximum value for $l(\lambda|X,Y,W)$ is when $\lambda = 10.696$.

3

Now we will implement this algorithm in R by seting the $\lambda_0 = 100$ and the stoping criterion to that if the λ dont change more then 0.001 from one interaction to another we will stop.

```r
EM_func<-function(my_data,start_lambda,eps=0.001,kmax=1000){

  #set x,y and z
  x<-my_data$X
  y<-my_data$Y
  z<-my_data$Z
```

```r
  #n and r
  n<-length(z)
  r<-sum(!is.na(z))

  #Extract the w values and x values for w
  j<-!is.na(z)
  x_for_w<-x[j]
  w<-z[j]

  #Set start lambda_k
  lambda_k<-start_lambda
  prev_lambda_k<-start_lambda*2 #Just to make the loop to start
  k<-1

  # Calc log like
  log_like_value<-log_like_fun(x,y,z,lambda_k)

  # Print the obtaiend values
  print(paste("K=",k," lambda=",round(lambda_k,2)," log-like=",round(log_like_value,4),sep=""))


  while(abs(lambda_k-prev_lambda_k)>eps && k<=kmax){ #Do until lambda not changing

    prev_lambda_k<-lambda_k                 #Save old lambda
    prev_log_like_value<-log_like_value     #Save old log-like

    #Set the new lambda
    lambda_k<-sum(x*y)/(2*n)+           #Use all y and x
      sum(x_for_w*w)/(4*n)+             #Use observed z and x that match
      (n-r)*2*prev_lambda_k/(4*n)       #Expected value for missing z




    log_like_value<-log_like_fun(x,y,z,lambda_k)   #Calc log-like of new lambda
    k<-k+1

    # Print the obtaiend values
    print(paste("K=",k," lambda=",round(lambda_k,4)," log-like=",round(log_like_value,4),sep=""))

  }
  best_lambda<<-lambda_k #Save the obtaind lamda
  number_of_k<<-k        #Svae the k
}
```

Now when we have implement the algorithm we can use it on our data.

```r
EM_func(physical1,start_lambda=100,eps=0.001)
```

```
## [1] "K=1 lambda=100 log-like=-653.7407"
## [1] "K=2 lambda=14.2678 log-like=-403.2794"
## [1] "K=3 lambda=10.8385 log-like=-396.0379"
```

```
## [1] "K=4 lambda=10.7014 log-like=-396.0211"
## [1] "K=5 lambda=10.6959 log-like=-396.021"
## [1] "K=6 lambda=10.6957 log-like=-396.021"
```

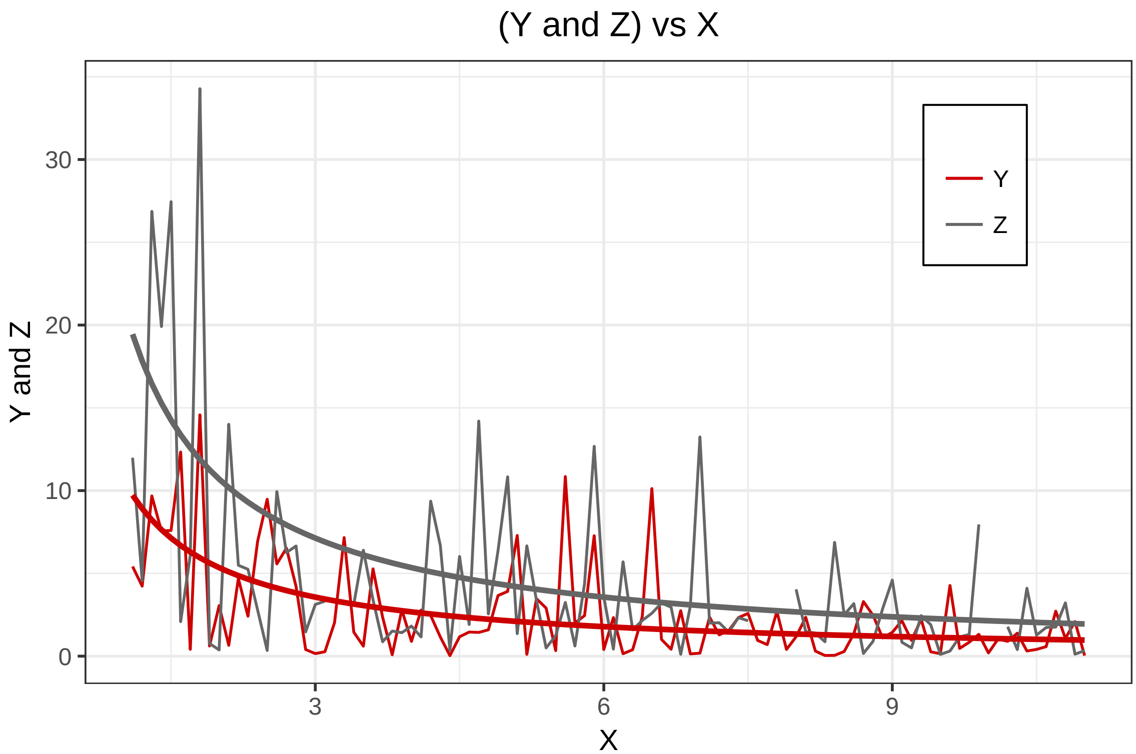
We can see that the optimal λ is 10.6957. We needed 6 interaction get reach this λ.

4

Here we will plot Y and Z vs X. We will also include the expected value of Y and Z in the plot.

```
E_y<-best_lambda/physical1$X      #E(Y)
E_z<-(best_lambda*2)/physical1$X  #E(Z)

#Just make a plot
ggplot(physical1,aes(x=X))+
  geom_line(aes(y=Y,col="Y"))+
  geom_line(aes(y=Z,col="Z"))+
  geom_line(aes(y=E_z),col="grey40",size=1)+
  geom_line(aes(y=E_y),col="red3",size=1)+
  theme_bw()+
  labs(y="Y and Z",title="(Y and Z) vs X")+
  theme(legend.position = c(0.85, 0.80),
        legend.box.background = element_rect(),
        legend.box.margin = margin(1, 1, 1, 1))+
  theme(plot.title = element_text(hjust = 0.5))+
  scale_colour_manual(name="",
                      breaks=c("Y", "Z"),
                      labels=c("Y", "Z"),values=c("red3","grey40"))
```



The expected values of Y and Z seems to follow the observed values of Y and Z in the best way they can when we appley a exponential distribution. Which means that we have selected the best λ as possible.