Genetic Algorithm

Define the function

Below is the objective function :

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

```
#objective function . This is used to check the fitness of the population
funcs <- function(x){
    frst_trm = x^2/exp(x)
    scnd_trm = 2*exp(-(9*sin(x))/((x^2)+x+1))

    y <- frst_trm - scnd_trm
    return(y)
}
```

Define the function crossover

```
#Produce new kid
crossover <- function(x,y){
    return((x+y)/2)
}
```

Define function mutate

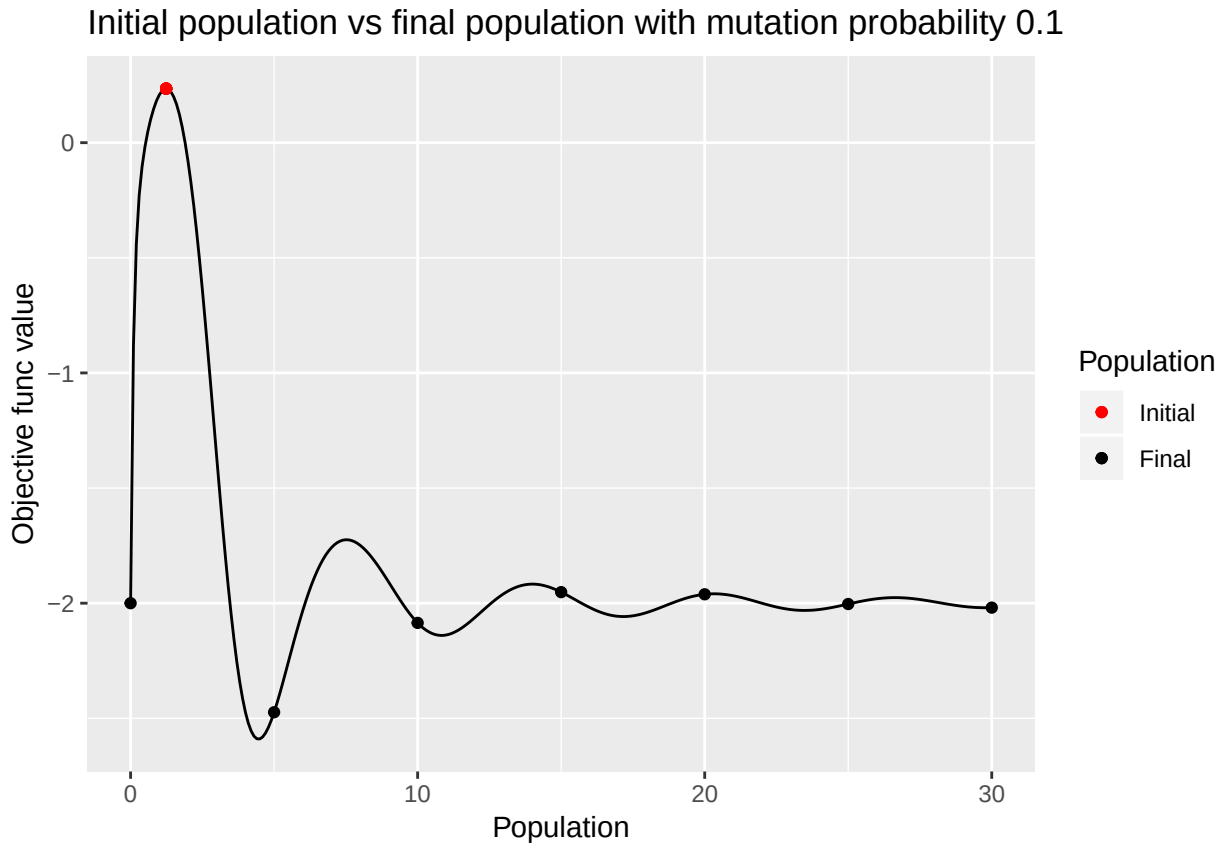
```
#Mutate the kid
mutate <- function(x) {
    return((x^2)%%30)
}
```

Function for genetic algorithm

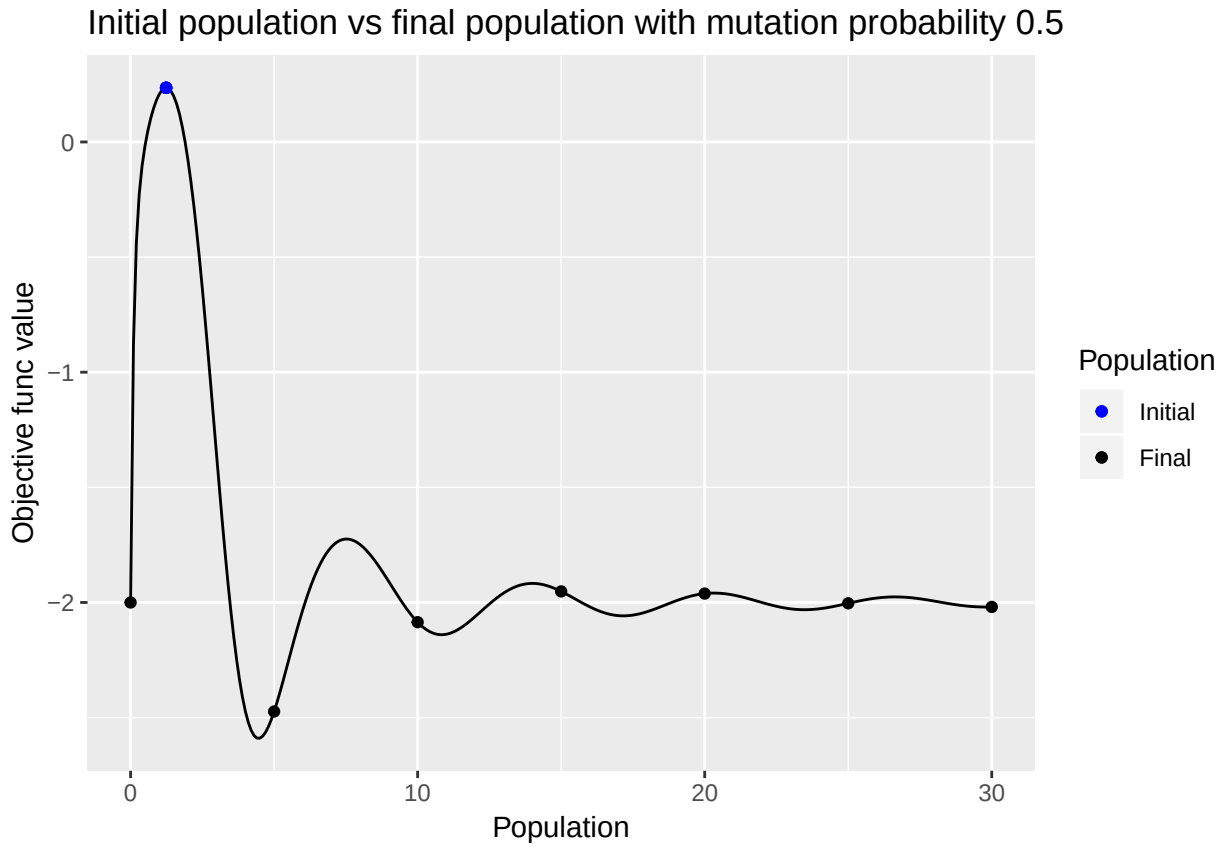
```
genetic_algorithm <- function(maxiter,mutprob) {
    x = seq(0,30,by=5)

    Values <- funcs(x)
    max_obj_func <- 0

    for(i in 1: maxiter) {
        s <- sample(1:7,2)         #Two index is randomly sampled from population

#Find the victim which has the least objectiv func
        victim <- order(Values)[1]
        kid <- crossover(x[s[1]],x[s[2]])  #Send the kids for the crossover

        u <- runif(1,0,1)           #Using random find check if we can mutate the kid
        if(mutprob>u)
            kid <- mutate(kid)
```

```
        x <- replace(x,victim,kid)

        new_val <- funcs(kid)  #calculate objective function with respect to the new kid
        Values <-replace(Values,victim,new_val) #Update the resultant objective function
    }


    return(list(x,Values))
}

gene_algo1 <- genetic_algorithm(10100,0.1)
gene_algo2 <- genetic_algorithm(10100,0.5)
gene_algo3 <- genetic_algorithm(10100,0.9)

 x <- seq(0,30,by=0.1)
 val <- funcs(x)

 data_pt <- cbind(x=x,val=val)
 data_pt <- as.data.frame(data_pt)

 x_ini=seq(0,30,by=5)
 y_ini=funcs(x_ini)
```

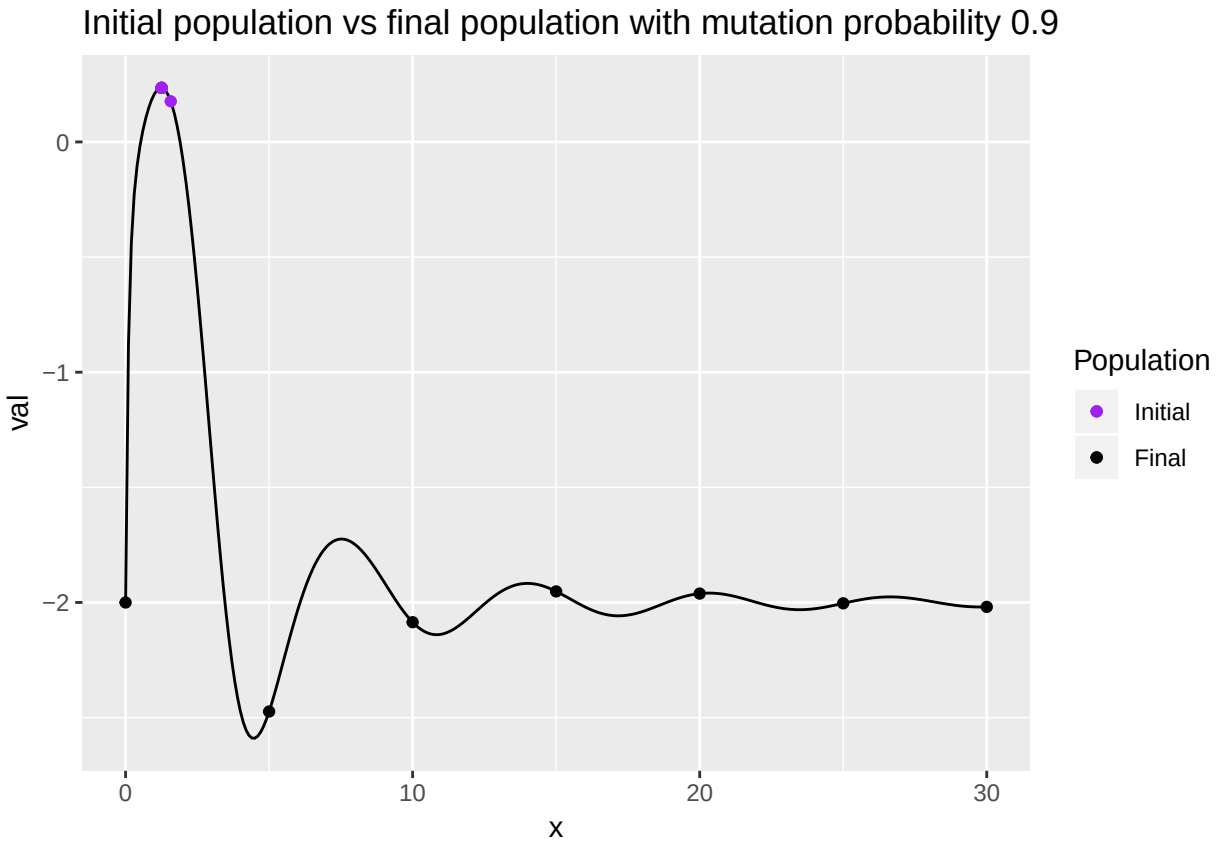
With mutation probability 0.1, not all the 5 points might reach maximum because the probability is very low. But it might happen due to random chance that the points might reach maxima since we are doing many iterations.



With mutation probability 0.5, the probability for the initial population to mutate is more than the previous one. Here we can see all the 5 points has converged to one.



With mutation probability 0.9, the probability to mutate further increases than the previous one. So most of the points lies on the peak.



Analysis

Here we choose victim based on the objective function. Basically we can cosnider the victim as the weakest in the population. So we tend to replace the weakest with the strong one with each iteration. With many iteration, all the the points in the population converges to maximum. With mutation probability 0.1, the tendency to mutate is very less. But with many iterations, the mutation keeps happens although at slow rate. The mutation further increases with probability 0.5 and 0.9