```r
par(family = 'serif')
setwd("/Users/mikhailgaerlan/Box Sync/Education/UC Davis/2016-2017
Spring/STA 243 Computational Statistics/Assignments/Assignment 2")
rm(list=ls())

#=========
#    2
#=========
#======================
# Test Data and Function
#======================
truefunction = function(x){
  t = c(0.1,0.13,0.15,0.23,0.25,0.4,0.44,0.65,0.76,0.78,0.81)
  h = c(4,-5,3,-4,5,-4.2,2.1,4.3,-3.1,2.1,-4.2)
  temp = 0
  for (i in 1:11){
    temp = temp+h[i]/2*(1+sign(x-t[i]))
  }
  return(temp)
}
n = 512
x = (0:(n-1))/n
f = truefunction(x)
set.seed(0401)
y = f+rnorm(f)/3
plot(x,y)
lines(x,f)

#======================
# Program functions
#======================
getparams = function(chromo){
  pieces = sum(chromo)+1
  breaks = 0*1:(pieces-1)
  breakindex = 0*1:(pieces-1)
  heights = 0*1:pieces
  j = 1
  for (n in 1:length(chromo)){
    if (chromo[n] == 1){
      breaks[j] = x[n+1]
      breakindex[j] = n+1
      if (j == 1){
        heights[j] = mean(y[1:n])
      } else{
        heights[j] = mean(y[(breakindex[j-1]+1):breakindex[j]])
      }
      sum = y[n]
      j = j + 1
    }
  }
  heights[pieces] = mean(y[(breakindex[pieces-1]):length(x)])
  return(c(pieces,breaks,heights,breakindex))
}

modelfunction = function(x,t,h){
  temp = 0
  for (j in 1:length(h)){
    if (j == 1){
      if ((x >= 0)&&(x < t[j])){
        temp = h[j]
        break
      }
    } else if (j == length(h)){
      if ((x >= t[j-1])&&(x <= 1)){
        temp = h[j]
        break
      }
    } else {
      if ((x >= t[j-1])&&(x < t[j])){
        temp = h[j]
        break
      }
    }
  }
  return(temp)
}

mdl = function(chromo){
  params = getparams(chromo)
  limits = params[2:(params[1])]
  heights = params[(params[1]+1):(2*params[1])]
  indices = params[(2*params[1]+1):length(params)]
  nj = 1:params[1]
  nj[1] = indices[1]-1
  for (i in 2:(params[1]-1)){
    nj[i] = indices[i]-indices[i-1]
  }
  nj[params[1]] = n+1-indices[params[1]-1]
  ymodel = getmodel(chromo,limits,heights)
  return(params[1]*log(n)+(1/2)*sum(log(nj))+(n/2)*log((1/n)*sum((y-
ymodel)^2)))
}

aic = function(chromo){
  params = getparams(chromo)
  limits = params[2:(params[1])]
  heights = params[(params[1]+1):(2*params[1])]
  ymodel = getmodel(chromo,limits,heights)
  return(n*log((1/n)*sum(y-ymodel)^2)+2*params[1]*log(n))
}
```

```
getmodel = function(chromo,limits,heights){
  ymodel = 0*x
  for (i in 1:length(x)){
    ymodel[i] = modelfunction(x[i],limits,heights)
  }
  return(ymodel)
}

rank = function(population,fitness){
  rankings = 0*fitness
  sortedfitness = sort(fitness,decreasing=TRUE)
  for (i in 1:dim(population)[1]){
    for (j in 1:dim(population)[1]){
      if (sortedfitness[i]==fitness[j]){
        rankings[j] = i
        break
      }
    }
  }
  return(rankings)
}

#========================
# Main Program
#========================
pop_size = 300
stop_gen = 20
cross_prob = 0.9
mut_prob = 0.05
printl = TRUE
criteria = FALSE
#TRUE for MDL
#FALSE for AIC
if (criteria){
  file_name = sprintf("mdl_gen_data_%d_%d.csv",pop_size,stop_gen)
} else{
  file_name = sprintf("aic_gen_data_%d_%d.csv",pop_size,stop_gen)
}

#----------------------------
# Population Initialization
#----------------------------
if (printl) print("Generation 0")
population = array(0,dim=c(pop_size,n-1))
fitness = array(0,dim=c(pop_size))
for (i in 1:pop_size){
  population[i,1:(n-1)] = sample(c(0,1),n-1,replace=TRUE)
  if (criteria){
    fitness[i] = mdl(population[i,1:(n-1)])
  } else{
    fitness[i] = aic(population[i,1:(n-1)])
  }
}
rankings = rank(population,fitness)

#Best Model
for (i in 1:pop_size){
  if (rankings[i] == pop_size){
    print(sprintf("Best Fitness: %f",fitness[i]))
    write(sprintf("%d %f",0,fitness[i]),file=file_name,append = FALSE)
    chromo = population[i,1:(n-1)]
    params = getparams(chromo)
    limits = params[2:(params[1])]
    heights = params[(params[1]+1):(2*params[1])]
    ymodel = getmodel(chromo,limits,heights)
    plot(x,y)
    title(main = sprintf("Generation 0"))
    lines(x,f)
    lines(x,ymodel,col='red')
    break
  }
}


#----------------------------
# Begin Genetic Algorithm
#----------------------------
stopping = 0
ngen = 1
while (stopping < stop_gen){
  if (printl) print(sprintf("Generation %d",ngen))
  offspring = array(0,dim=c(pop_size,n-1))
  off_fitness = array(0,dim=c(pop_size))

  #----------------------------
  # Creating new generation
  #----------------------------
  if (printl) print("Making offspring")
  for (i in 1:pop_size){
    if (runif(1) < cross_prob){
      parents =
sample(1:pop_size,2,replace=FALSE,prob=rankings/sum(rankings))
      for (j in 1:(n-1)){
        if (runif(1) < 1/2) {
          offspring[i,j] = population[parents[1],j]
        } else{
          offspring[i,j] = population[parents[2],j]
        }
      }
    } else {
      parent =
sample(1:pop_size,1,replace=FALSE,prob=rankings/sum(rankings))
```

```
      offspring[i] = population[parent[1]]
      for (j in 1:(n-1)){
        if (runif(1) < mut_prob){
          if (offspring[i,j] == 0){
            offspring[i,j] = 1
          } else{
            offspring[i,j] = 0
          }
        }
      }
    }
  }
  for (i in 1:pop_size){
    if (criteria){
      off_fitness[i] = mdl(offspring[i,1:(n-1)])
    } else{
      off_fitness[i] = aic(offspring[i,1:(n-1)])
    }
  }

  #---------------------------
  # Elitist
  #---------------------------
  #Combine parents and offspring
  total_pop = rbind(population,offspring)
  total_fit = rbind(fitness,off_fitness)
  total_rank = rank(total_pop,total_fit)
  #Keep only n best individuals
  off_rankings = array(0,dim=c(pop_size))
  for (i in 1:pop_size){
    for (j in 1:(2*pop_size)){
      if (total_rank[j] == (2*pop_size+1-i)){
        offspring[i] = total_pop[j]
        off_fitness[i] = total_fit[j]
        off_rankings[i] = pop_size+1-i
      }
    }
  }

  for (i in 1:pop_size){
    if (off_rankings[i] == pop_size){
      print(sprintf("Best Fitness: %f",off_fitness[i]))
      write(sprintf("%d %f",ngen,off_fitness[i]),file=file_name,append =
TRUE)
      chromo = offspring[i,1:(n-1)]
      params = getparams(chromo)
      limits = params[2:(params[1])]
      heights = params[(params[1]+1):(2*params[1])]
      ymodel = getmodel(chromo,limits,heights)
      plot(x,y)
      title(main = sprintf("Generation %d",ngen))
      lines(x,f)
      lines(x,ymodel,col='red')
      break
    }
  }

  i = 1
  while (i <= pop_size){
    if (off_rankings[i]==pop_size){
      j = 1
      while (j <= pop_size){
        if (rankings[j]==pop_size){
          if (off_fitness[i] == fitness[j]){
            stopping = stopping + 1
            if (printl) print(sprintf("Same for %d
generations",stopping))
            i = pop_size
            j = pop_size
          } else{
            stopping = 0
            if (printl) print(sprintf("Not the same..."))
            i = pop_size
            j = pop_size
          }
        }
        j = j + 1
      }
    }
    i = i + 1
  }

  population = offspring
  fitness = off_fitness
  rankings = off_rankings
  ngen = ngen + 1
}

#Best Model
for (i in 1:pop_size){
  if (rankings[i] == pop_size){
    chromo = population[i,1:(n-1)]
    params = getparams(chromo)
    limits = params[2:(params[1])]
    heights = params[(params[1]+1):(2*params[1])]
    ymodel = getmodel(chromo,limits,heights)
    plot(x,y)
    title(main = sprintf("Generation %d",ngen-1))
    lines(x,f)
    lines(x,ymodel,col='blue')
    break
  }
```

```
}
```