STA 243 Spring 2018 **Assignment 2** Due date: April 20 (5pm), 2018

For this assignment you can do it in a group of maximum three people. Objectives of this assignment:

- To make yourself more familiar with R programming (or any other languages).
- To gain practical experience with simulated annealing and genetic algorithms.

1 Simulated Annealing

In this question we want to solve the traveling salesman problem discussed in class using simulated annealing. The matrix below summarizes the distances between any two of 15 cities; e.g., City B and City E are 7 units apart, while City C and City O are 4 units apart.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
A	0	1	2	4	9	8	3	2	1	5	7	1	2	9	3
B	1	0	5	3	7	2	5	1	3	4	6	6	6	1	9
C	2	5	0	6	1	4	7	7	1	6	5	9	1	3	4
D	4	3	6	0	5	2	1	6	5	4	2	1	2	1	3
E	9	7	1	5	0	9	1	1	2	1	3	6	8	2	5
F	8	2	4	2	9	0	3	5	4	7	8	3	1	2	5
G	3	5	7	1	1	3	0	2	6	1	7	9	5	1	4
H	2	1	7	6	1	5	2	0	9	4	2	1	1	7	8
I	1	3	1	5	2	4	6	9	0	3	3	5	1	6	4
J	5	4	6	4	1	7	1	4	3	0	9	1	8	5	2
K	7	6	5	2	3	8	7	2	3	9	0	2	1	8	1
L	1	6	9	1	6	3	9	1	5	1	2	0	5	4	3
M	2	6	1	2	8	1	5	1	1	8	1	5	0	9	6
N	9	1	3	1	2	2	1	7	6	5	8	4	9	0	7
O	3	9	4	3	5	5	4	8	4	2	1	3	6	7	0

Implement a simulated annealing algorithm to find the optimal traveling path. To begin with, set $\alpha(\tau) = p\tau$ with $p = 0.999$, $\beta(m) = 100$ (i.e., $m_j = 100$ for all j), and use a uniform distribution for the proposal density $g_t(\boldsymbol{\theta}|\boldsymbol{\theta}_t)$. Also, set the initial temperature $\tau_1 = 400$. Once your algorithm is working, play around with different values of τ_1, p etc to see how the algorithm behaves. (The shortest path is of length 17.)

2 Genetic Algorithms

For this question you will develop automatic procedures for fitting piecewise constant regression. Loosely speaking, your task is to use the circles in Figure 1 to estimate the true function which is also displayed in the same figure. You are also welcome to try any extensions, generalizations and/or modifications. We begin with the problem statement.

2.1 Problem Statement

Suppose n pairs of noisy measurements (x_i, y_i) are observed, with

$$y_i = f(x_i) + e_i, \quad x_1 < \ldots < x_n, \quad e_i \sim \text{ iid } N(0, \sigma^2), \quad i = 1, \ldots, n.$$

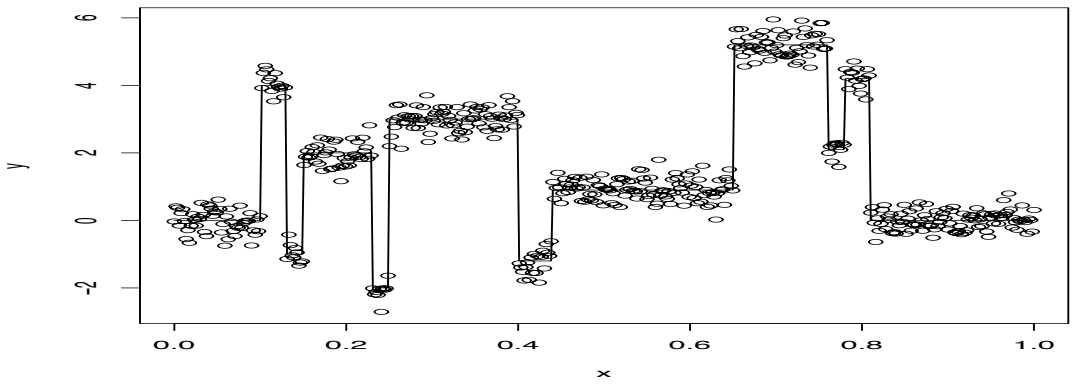

Figure 1: *The circles are the observations* $\{(x_i, y_i)\}_{i=1}^n$ *while the solid line is the true regression function* $f(x)$*. Your task is to estimate* $f(x)$ *given* $\{(x_i, y_i)\}_{i=1}^n$*. This figure is generated by the R-codes listed in Section 2.6.*

The aim is to estimate f. It is known that f is a piecewise constant function, but other details, such as the number of pieces, are unknown.

Let the (unknown) number of pieces be B, and the different pieces are joined at breakpoints $b_1, b_2, \ldots, b_{B-1}$. Without loss of generality, let $b_0 = 0 = x_1$ and $b_B = x_n + \delta = 1$ for a small $\delta > 0$, and assume $b_0 < b_1 < \ldots < b_B$. Let I_E be the indicator function for the event E; that is, $I_E = 1$ if E is true and $I_E = 0$ otherwise. Then our regression model for f is

$$f(x) = f_1 I_{\{b_0 \le x < b_1\}} + f_2 I_{\{b_1 \le x < b_2\}} + \ldots + f_B I_{\{b_{B-1} \le x < b_B\}}, \tag{1}$$

where f_j is the function value (or the "height") of the j-th piece of $f(x)$. To estimate $f(x)$ with the regression model (1), we need to estimate B, $b_1, \ldots, b_{B-1}$, and $f_1, \ldots, f_B$. For clarity, we collect all these parameters in a vector $\boldsymbol{\theta} = (B, b_1, \ldots, b_{B-1}, f_1, \ldots, f_B)$ and denote the corresponding estimates as $\hat{\boldsymbol{\theta}} = (\hat{B}, \hat{b}_1, \ldots, \hat{b}_{\hat{B}-1}, \hat{f}_1, \ldots, \hat{f}_{\hat{B}})$. Unfortunately, for the estimation of $\boldsymbol{\theta}$, the least-squares principle does not work here, nor maximum likelihood (why?). Thus we need to switch to some other methods.

Before proceeding further, we remark that once B and $b_1, \ldots, b_{B-1}$ are estimated, $f_1, \ldots, f_B$ can be uniquely estimated by

$$\hat{f}_j = \frac{1}{\hat{n}_j} \sum_{\hat{b}_{j-1} \le x_i < \hat{b}_j} y_i,$$

where $\hat{n}_j$ is the number of x_i that are inside the interval $[\hat{b}_{j-1}, \hat{b}_j)$. In other words, f_j is estimated by the average of all the y_i's that are in the estimated j-th piece $[\hat{b}_{j-1}, \hat{b}_j)$.

2.2 Model Selection Methods

Now we presents two methods for estimating a "best" fitting model $\hat{\boldsymbol{\theta}}$: the minimum description length (MDL) principle and the Akaike information criterion (AIC).

Minimum Description Length Principle: The MDL principle *defines* the best fitting model as the one that produces the shortest code length of the data; see [5] and references given therein. We will skip the details and state the result that, for our problem, the best $\hat{f}$ (equivalently $\hat{\boldsymbol{\theta}}$) is estimated

as the minimizer of

$$\mathrm{MDL}(\hat{f}) = \hat{B}\log n + \frac{1}{2}\sum_{j=1}^{\hat{B}} \log \hat{n}_j + \frac{n}{2}\log\left[\frac{1}{n}\sum_{i=1}^{n}\left\{y_i - \hat{f}(x_i)\right\}^2\right].$$

Akaike Information Criterion: With AIC the best fitting model is chosen as the one that minimizes an estimator of the Kullback–Leibler (KL) distance measure between a fitted model and the "true" model (e.g., see [1]). If p is the number of parameters that need to be estimated in a fitted model, then under mild regularity conditions one can show that such a KL distance estimator is $-2 \times$ "maximized log likelihood" $+ 2p$. For our piecewise constant function fitting problem this distance estimator amounts to

$$\mathrm{AIC}(\hat{f}) = n\log\left[\frac{1}{n}\sum_{i=1}^{n}\left\{y_i - \hat{f}(x_i)\right\}^2\right] + \gamma p\Big|_{\gamma=2}.$$

However, it is known that for similar problems $\gamma = \log n$ is a better choice than $\gamma = 2$. Therefore in here we shall select the $\hat{f}$ that minimizes $\mathrm{AIC}(\hat{f})$ with $\gamma = \log n$ and $p = 2\hat{B}$.

2.3 Minimization by Genetic Algorithms

When the number of data points is large, finding the best estimate according to any of the above criteria involves solving a hard, large scale minimization problem. We shall use genetic algorithms for this; for a general introduction, see [2] for example. Genetic algorithms have been applied successfully to solve different statistical estimation problems; e.g., see [3, 4] and references given therein.

General Description: Genetic algorithms solve minimization problems by mimicking the natural evolution of living structures on earth, which can be briefly described as follows. An initial set, or population, of possible solutions to a minimization problem is obtained and represented in vector form. These vectors are often called *chromosomes* and are free to "evolve" in the following way. Parent chromosomes are randomly chosen from the initial population and chromosomes having lower values of the objective criterion to be minimized would have a higher chance of being chosen. Offspring are then produced by applying a *crossover* or a *mutation* operation to the chosen parents. Once a sufficient number of such second generation offspring are produced, third generation offspring are further produced from these second generation offspring. This process continues for a number of generations. If one believes in Darwin's Natural Selection, the expectation is that objective criterion values of the offspring will gradually improve over generations and approach the optimal value.

Crossover: In a crossover operation, one child chromosome is produced from "mixing" two parent chromosomes. The aim is to allow the possibility that the child receives different best parts from its parents. A typical "mixing" strategy is that every child gene location has an equal chance of receiving either the corresponding father gene or the corresponding mother gene. This crossover operation is the distinct feature that makes genetic algorithms different from other optimization methods.

Mutation: In a mutation operation one child chromosome is produced from one parent chromosome. The child is essentially the same as its parent except for a small number of genes where randomness is introduced to alter the types of these genes. Such a mutation operation prevents the algorithm being trapped in local minima.

Chromosome Representation: First recall that for our problem, a possible solution $\hat{\boldsymbol{\theta}}$ can be uniquely specified by $\hat{B}$, $\hat{b}_1, \ldots, \hat{b}_{\hat{B}-1}$. Thus for our problem a chromosome only needs to carry information about $\hat{B}$, $\hat{b}_1, \ldots, \hat{b}_{\hat{B}-1}$. A simple example will be used to illustrate this representation scheme. Suppose $n = 20$, $\hat{B} = 3$, $\hat{b}_1 = 5$ and $\hat{b}_2 = 12$. That is, the estimate $\hat{f}(x)$ is composed of three pieces separated at x_5 and x_{12}. If we use "1" to denote a breakpoint gene and "0" to denote a normal gene, then the chromosome for this example is composed of $n = 20$ genes arranged as: 00001000000100000000.

2.4 What is Your Task?

Your task in this project is to implement a genetic algorithm for fitting the piecewise constant regression model (1). You will need to implement both MDL($\hat{f}$) and AIC($\hat{f}$). If you are using R, you should write an R function that takes two input arguments, the noisy data and an indicator specifying if MDL or AIC should be used. As outputs, your R function should plot the noisy data set as well as the fitting piecewise constant function on the screen. Write a brief report (maximum two pages) summarizing your experience gained from this question. Additional implementation details are given in Section 2.5.

For those who want more: You are of course welcome to try other extensions, generalizations and/or modifications. Here are two simple suggestions: perform a simulation study to formally compare MDL($\hat{f}$) and AIC($\hat{f}$), and, instead of piecewise constant, do piecewise polynomial (see [4]).

2.5 Implementation of Genetic Algorithms

This subsection lists the major steps in the implementation of the genetic algorithm for the minimization of MDL($\hat{f}$). A genetic algorithm for minimizing AIC($\hat{f}$) is similar and hence omitted. The steps are as follows.

1. Randomly generate an initial population of chromosomes of size S.
2. Compute the MDL($\hat{f}$) value for each of the S chromosomes.
3. Sort the MDL($\hat{f}$) values in descending order. Assign rank 1 to the chromosome with the largest MDL($\hat{f}$) value, rank 2 to the one with the second largest MDL($\hat{f}$) value, and so on. Denote the rank of chromosome i as r_i.
4. With probability P_{cross}, perform a crossover operation. Otherwise, perform a mutation operation.
   - *Crossover* — one child is produced from two parents: select two parent chromosomes from the initial population. The probability that chromosome i will be selected is $r_i/(r_1+\ldots+r_S)$. Then for each gene location of the child chromosome, with equal probability, assign it either the corresponding father gene or the corresponding mother gene.
   - *Mutation* — one child is produced from one parent: select one parent using the same probability law as in crossover. First copy the parent's gene to the child (i.e., the child is initially identical to the parent). Then, for each child gene location, change its gene value with probability P_c.
5. Repeat Step 4 until S child chromosomes are produced. That is, until a whole new generation is obtained.

6. Repeat Steps 2 to 5 using the new generation as the initial population. Then repeat the whole process until the MDL($\hat{f}$) value of the best chromosome does not change for N_{same} generations.

7. The best chromosome of the youngest generation is taken as the minimizer of MDL($\hat{f}$).

You can start with $S = 300$, $P_{cross} = 0.9$, $P_c = 0.05$ and $N_{same} = 20$. You may also want to experiment with different values.

2.6 *R*-Codes for Generating Figure 1

```
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)
```

References

[1] K. P. Burnham and D. R. Anderson. *Model Selection and Inference: A Practical Information-Theoretic Approach*. Springer-Verlag New York Inc., 1998.

[2] L. Davis. *Handbook of Genetic Algorithms*. Van Nostrand Reinhold, New York, 1991.

[3] R. A. Davis, T. C. M. Lee, and G. A. Rodriguez-Yam. Structural break estimation for non-stationary time series models. *Journal of the American Statistical Association*, 101:223–239, 2006.

[4] T. C. M. Lee. Automatic smoothing for discontinuous regression functions. *Statistica Sinica*, 12:823–842, 2002.

[5] J. Rissanen. *Stochastic Complexity in Statistical Inquiry*. World Scientific, Singapore, 1989.