Computational Statistics HW2

Code

Example 3.3, 3.4, 3.5 / Question 3-1, 3-3, 3-4 (a)(b)(c-i,ii)

Hide

```
current_path = rstudioapi::getActiveDocumentContext()$path
setwd(dirname(current_path))
baseball <- read.table("baseball.dat", header=TRUE)

# Example 3.3
baseball$freeagent = factor(baseball$freeagent)
baseball$arbitration = factor(baseball$arbitration)
baseball.sub = baseball[, -1]
salary.log = log(baseball$salary)
n = length(salary.log)
m = length(baseball.sub[1,])
num = 5
runs = matrix(0, num, m)
iter = 15
runs.aic = matrix(0, num, iter)

set.seed(123)
for(i in 1:num){runs[i,] = rbinom(m,1,.5)}

for(k in 1:num){
  run.current = runs[k,]
  # iterates each random start
  for(j in 1:iter){
    run.vars = baseball.sub[,run.current==1]
    g = lm(salary.log~.,run.vars)
    run.aic = extractAIC(g)[2]
    run.next = run.current
    # tests all models in 1-neighborhood and selects the model with lowest AIC
    for(i in 1:m){
      run.step = run.current
      run.step[i] = !run.current[i]
      run.vars = baseball.sub[,run.step==1]
      g = lm(salary.log~.,run.vars)
      run.step.aic = extractAIC(g)[2]
      if(run.step.aic < run.aic){
        run.next = run.step
        run.aic = run.step.aic
      }
    }
    run.current = run.next
    runs.aic[k,j]=run.aic
  }
  runs[k,] = run.current
}

## Output: lists of predictors and AIC values
runs
```

```
     [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10] [,11] [,12] [,13] [,14] [,
15] [,16] [,17]
[1,]    0    0    1    0    0    0    0    1    0     1     0     0     1     1
1     0     0
[2,]    0    0    0    1    0    1    0    1    0     1     0     0     1     1
0     0     0
[3,]    1    0    1    0    0    0    0    1    0     1     1     0     1     1
1     1     0
[4,]    1    0    1    0    0    1    0    1    0     1     0     0     1     1
1     1     0
[5,]    0    1    1    0    0    1    0    1    0     1     0     0     1     1
1     1     0
     [,18] [,19] [,20] [,21] [,22] [,23] [,24] [,25] [,26] [,27]
[1,]     0     0     1     1     1     0     0     1     1     0
[2,]     0     0     0     0     0     0     1     1     0     0
[3,]     0     0     0     0     0     0     1     0     0     1
[4,]     0     0     0     0     0     0     1     1     1     0
[5,]     0     0     0     0     0     0     1     1     1     0
```

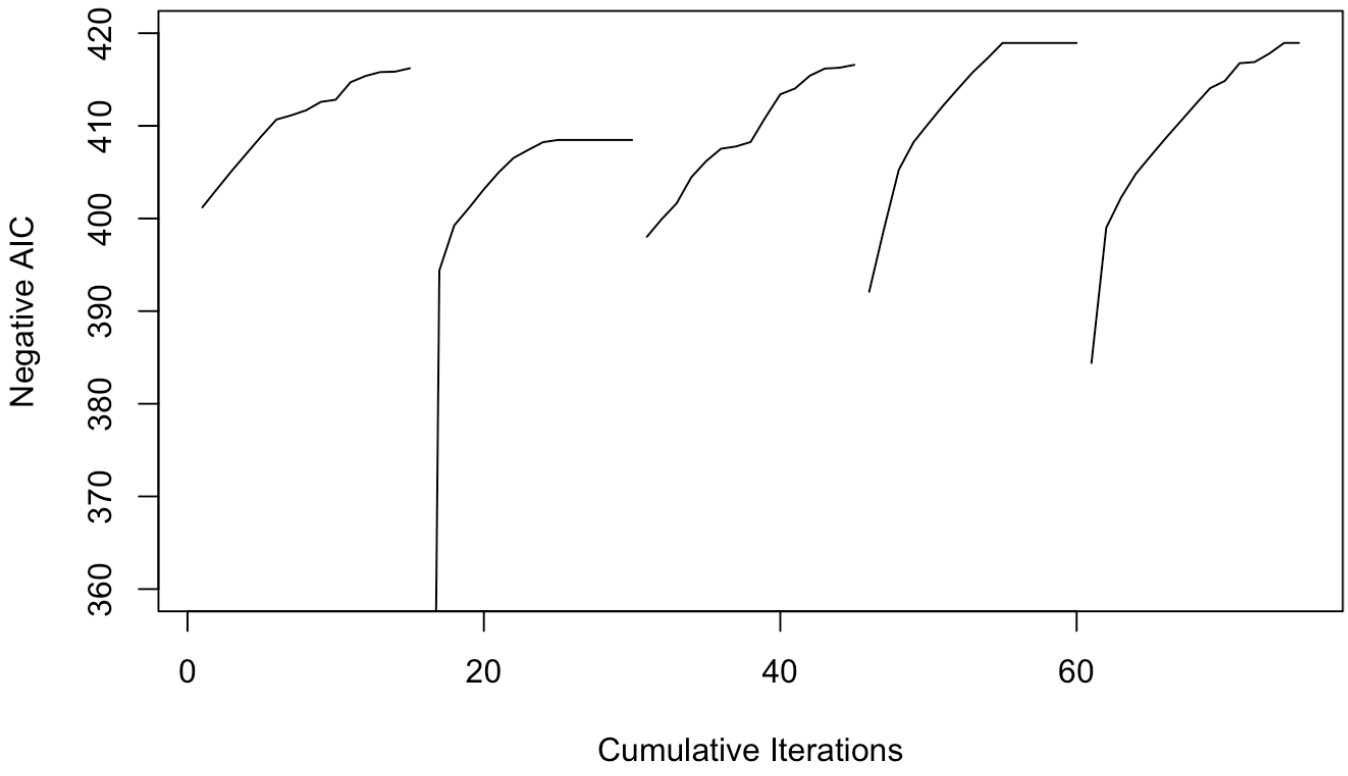
Hide

```
runs.aic
```

```
          [,1]      [,2]      [,3]      [,4]      [,5]      [,6]      [,7]      [,
8]      [,9]
[1,] -401.2039 -403.1991 -405.1694 -407.0423 -408.9099 -410.6776 -411.1377 -411.67
64 -412.5852
[2,] -230.3428 -394.4185 -399.2532 -401.1517 -403.1507 -404.9759 -406.5342 -407.40
77 -408.2400
[3,] -398.0244 -399.9323 -401.6238 -404.4554 -406.1898 -407.5306 -407.7678 -408.25
48 -410.9158
[4,] -392.0918 -398.8837 -405.2592 -408.2478 -410.2474 -412.1952 -414.0142 -415.79
85 -417.3195
[5,] -384.4041 -399.0112 -402.2639 -404.8337 -406.7615 -408.6514 -410.4633 -412.28
16 -414.0570
         [,10]     [,11]     [,12]     [,13]     [,14]     [,15]
[1,] -412.8168 -414.7061 -415.3752 -415.8019 -415.8462 -416.2141
[2,] -408.4745 -408.4745 -408.4745 -408.4745 -408.4745 -408.4745
[3,] -413.4002 -414.0330 -415.4141 -416.1781 -416.2747 -416.5833
[4,] -418.9421 -418.9421 -418.9421 -418.9421 -418.9421 -418.9421
[5,] -414.8493 -416.7620 -416.8813 -417.7992 -418.9472 -418.9472
```

Hide

```
## Plotting
plot(1:(iter*num), -c(t(runs.aic)), xlab="Cumulative Iterations",
     ylab="Negative AIC", ylim=c(360,420), type="n")
for(i in 1:num) {
  lines((i-1)*iter+(1:iter), -runs.aic[i,]) }
```



Hide

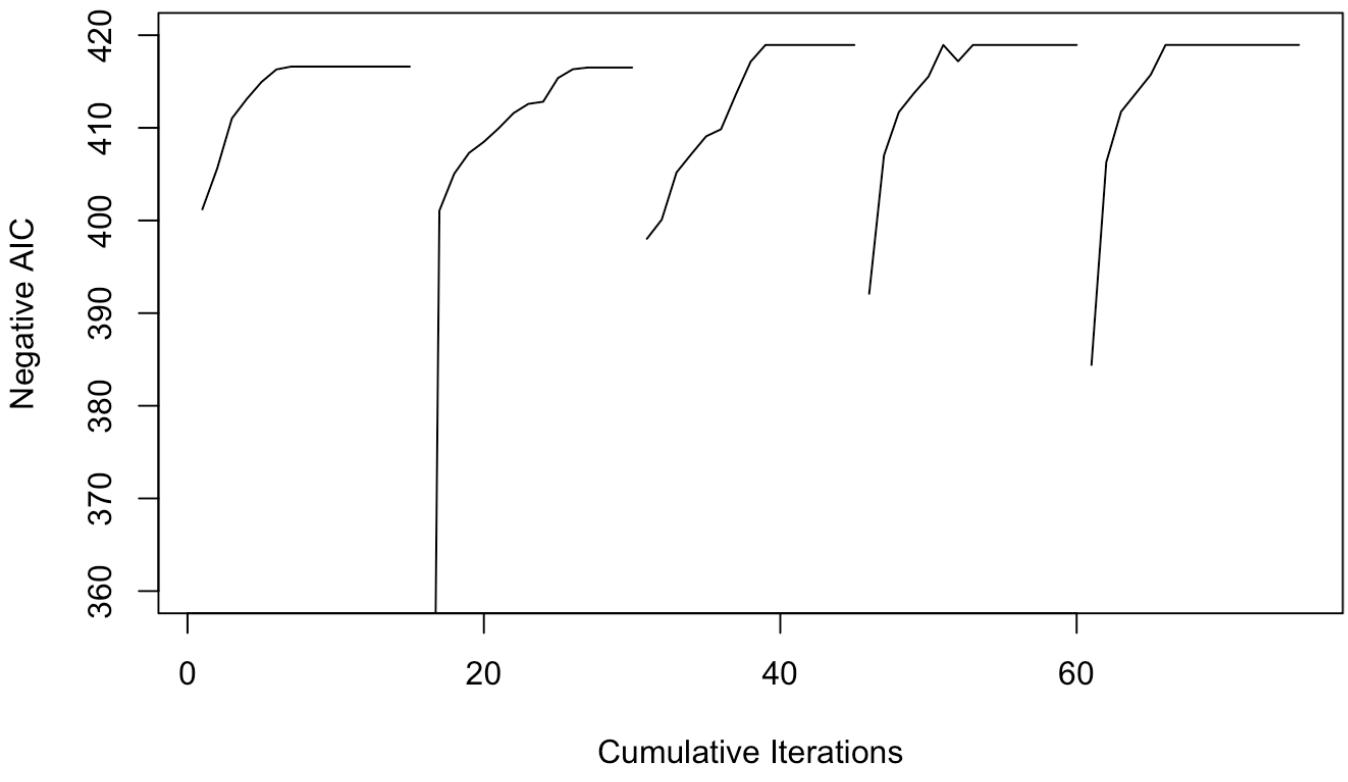
```
# Question 3-1
m = ncol(baseball)-1
num = 5
runs = matrix(0, num, m)
iter = 15
runs.aic = matrix(0, num, iter)

set.seed(123)
for(i in 1:num){runs[i,] = rbinom(m, 1, 0.5)}

# Local search, 1-neighbor
for(k in 1:num){
  run.current = runs[k,]  # mod == run.current
  # iterates each random start
  for(l in 1:iter){
    run.vars = baseball.sub[, run.current==1]  # base1 == run.vars
```

```
    g = lm(salary.log~., run.vars)  # fit == g
    run.aic = extractAIC(g)[2]  # run.aic == AIC_opt == AIC_seq
    run.next = run.current
    numcal = 1
    more = TRUE
    # immediate adoption of first randomly selected downhill neighbor
    while(more) {
      ind = sample(1:m, m)
      i = 0
      more2 = TRUE
      while(more2){
        i = i+1
        j = ind[i]
        run.step = run.current
        run.step[j] = 1 - run.step[j]
        run.vars = baseball.sub[, run.step==1]
        g = lm(salary.log~., data=run.vars)
        run.step.aic = extractAIC(g)[2]
        numcal = numcal+1
        if((run.step.aic < run.aic) | (i==m))
          more2 = FALSE
      }
      more = FALSE
      if(run.step.aic < run.aic) {
        more = TRUE
        run.next[j] = 1-run.next[j]
        run.aic = run.step.aic
      }
    }
    run.current = run.next
    runs.aic[k,l] = run.aic
  }
  runs[k,] = run.current
}

## Result & Plot
runs.aic
```

```
          [,1]      [,2]      [,3]      [,4]      [,5]      [,6]      [,7]      [,
8]      [,9]
[1,] -401.2039 -405.6021 -411.0291 -413.1184 -414.9606 -416.3035 -416.6119 -416.61
19 -416.6119
[2,] -230.3428 -401.0692 -405.0550 -407.2947 -408.4898 -409.9563 -411.5865 -412.58
52 -412.8168
[3,] -398.0244 -400.0871 -405.1864 -407.1695 -409.0854 -409.8435 -413.5947 -417.13
73 -418.9421
[4,] -392.0918 -407.0360 -411.6925 -413.6925 -415.5272 -418.9421 -417.1816 -418.94
21 -418.9421
[5,] -384.4041 -406.2934 -411.7459 -413.7357 -415.7336 -418.9472 -418.9472 -418.94
72 -418.9472
         [,10]     [,11]     [,12]     [,13]     [,14]     [,15]
[1,] -416.6119 -416.6119 -416.6119 -416.6119 -416.6119 -416.6119
[2,] -415.3752 -416.3228 -416.5069 -416.5069 -416.5069 -416.5069
[3,] -418.9421 -418.9421 -418.9421 -418.9421 -418.9421 -418.9421
[4,] -418.9421 -418.9421 -418.9421 -418.9421 -418.9421 -418.9421
[5,] -418.9472 -418.9472 -418.9472 -418.9472 -418.9472 -418.9472
```

Hide

```
plot(1:(iter*num), -c(t(runs.aic)), xlab="Cumulative Iterations",
     ylab="Negative AIC", ylim=c(360,420), type="n")
for(i in 1:num) {
  lines((i-1)*iter+(1:iter), -runs.aic[i,]) }
```



Hide

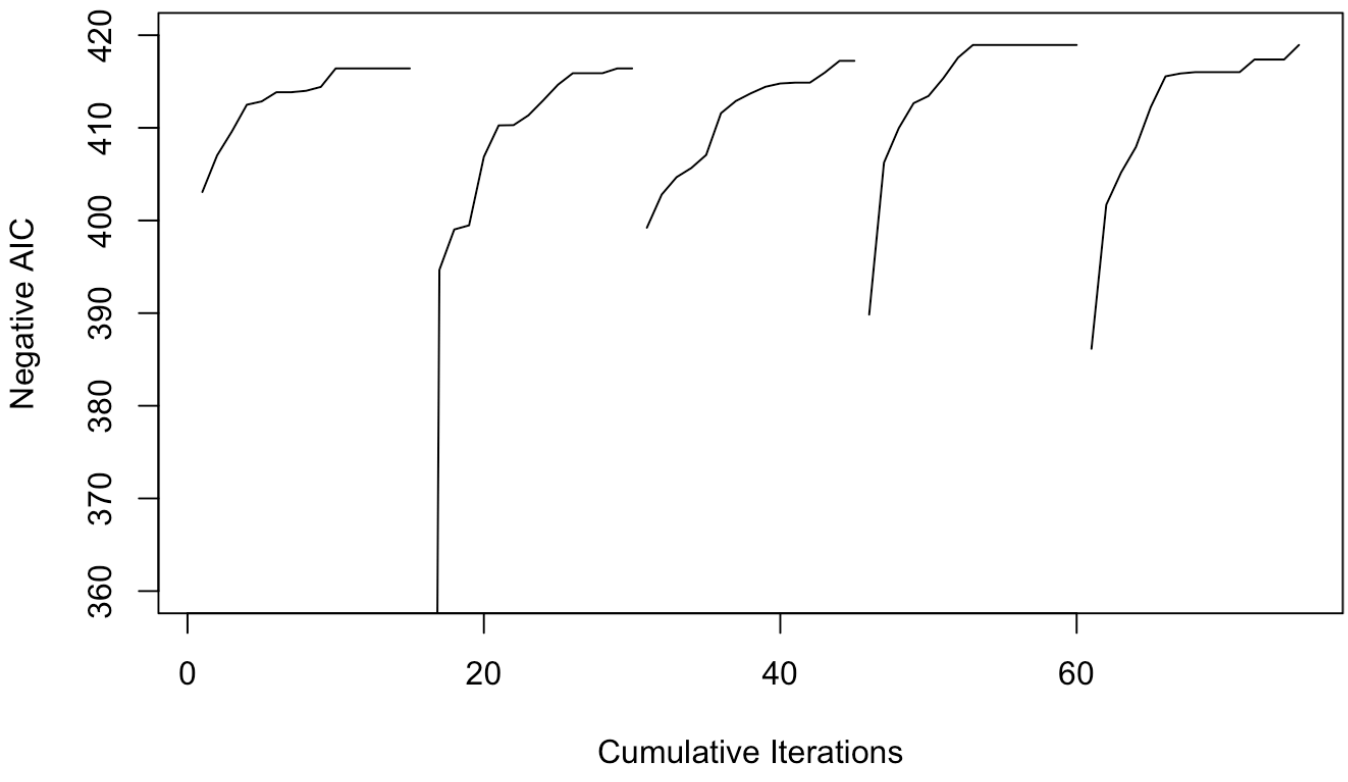
Hide

```
# b) Local search, 2-neighbor
n = length(salary.log)
m = length(baseball.sub[1,])
num = 5
runs = matrix(0, num, m)
iter = 15
runs.aic = matrix(0, num, iter)

set.seed(123)
for(i in 1:num){runs[i,] = rbinom(m,1,.5)}

for(k in 1:num){
  run.current = runs[k,]
  # iterates each random start
  for(j in 1:iter){
    run.vars = baseball.sub[,run.current==1]
    g = lm(salary.log~.,run.vars)
    run.aic = extractAIC(g)[2]
    run.next = run.current
    # tests all models in 2-neighborhood and selects the model with lowest AIC
    for(i in 1:m){
      run.step = run.current
      pos = sample(1:m, 2)
      run.step[pos[1]] = 1-run.step[pos[1]]
      run.step[pos[2]] = 1-run.step[pos[2]]
      run.vars = baseball.sub[, run.step==1]
      g = lm(salary.log~.,run.vars)
      run.step.aic = extractAIC(g)[2]
      if(run.step.aic < run.aic){
        run.next = run.step
        run.aic = run.step.aic
      }
    }
    run.current = run.next
    runs.aic[k,j]=run.aic
  }
  runs[k,] = run.current
}

## Result & Plot
runs.aic
```

```
          [,1]      [,2]      [,3]      [,4]      [,5]      [,6]      [,7]      [,
8]      [,9]
[1,] -403.0730 -407.0310 -409.6213 -412.4918 -412.8494 -413.8435 -413.8435 -413.99
58 -414.4195
[2,] -126.3887 -394.6757 -399.0293 -399.4752 -406.8727 -410.2633 -410.2910 -411.34
37 -412.9588
[3,] -399.2215 -402.8008 -404.6743 -405.6663 -407.0740 -411.5823 -412.8930 -413.71
51 -414.4314
[4,] -389.8475 -406.2514 -409.9747 -412.6602 -413.4349 -415.3516 -417.5822 -418.94
21 -418.9421
[5,] -386.1525 -401.6981 -405.1761 -407.9296 -412.2406 -415.5614 -415.8665 -416.01
10 -416.0110
         [,10]     [,11]     [,12]     [,13]     [,14]     [,15]
[1,] -416.4101 -416.4101 -416.4101 -416.4101 -416.4101 -416.4101
[2,] -414.6568 -415.8958 -415.8958 -415.8958 -416.4101 -416.4101
[3,] -414.7888 -414.8776 -414.8776 -415.9589 -417.2310 -417.2310
[4,] -418.9421 -418.9421 -418.9421 -418.9421 -418.9421 -418.9421
[5,] -416.0110 -416.0110 -417.3726 -417.3726 -417.3726 -418.9421
```

Hide

```
plot(1:(iter*num), -c(t(runs.aic)), xlab="Cumulative Iterations",
     ylab="Negative AIC", ylim=c(360,420), type="n")
for(i in 1:num) {
  lines((i-1)*iter+(1:iter), -runs.aic[i,]) }
```



Hide

Hide

```
# Example 3.4
baseball$freeagent = factor(baseball$freeagent)
baseball$arbitration = factor(baseball$arbitration)
baseball.sub = baseball[, -1]
salary.log = log(baseball$salary)
n = length(salary.log)
m = length(baseball.sub[1,])
cooling = c(rep(60,5), rep(120,5), rep(220,5))
tau_start = 10
tau = rep(tau_start, 15)
aics = NULL

set.seed(123)
run = rbinom(m, 1, 0.5)
run.current = run
run.vars = baseball.sub[, run.current==1]
g = lm(salary.log~., run.vars)
run.aic = extractAIC(g)[2]
best.aic = run.aic
aics = run.aic
for(j in 2:15) {tau[j] = 0.9*tau[j-1]}

for(j in 1:15) {
  for(i in 1:cooling[j]) {
    pos = sample(1:m,1)
    run.step = run.current
    run.step[pos] = !run.current[pos]
    run.vars = baseball.sub[,run.step==1]
    g = lm(salary.log~.,run.vars)
    run.step.aic = extractAIC(g)[2]
    p = min(1,exp((run.aic-extractAIC(g)[2])/tau[j]))
    if(run.step.aic < run.aic){
      run.current = run.step
      run.aic = run.step.aic}
    if(rbinom(1,1,p)){
      run.current = run.step
      run.aic = run.step.aic}
    if(run.step.aic < best.aic){
      run = run.step
      best.aic = run.step.aic}
    aics = c(aics,run.aic)
  }
}

## Output
run         # Best list of predictors found
```

```
 [1] 1 1 1 0 0 1 0 1 0 1 0 0 1 1 1 1 1 0 0 0 0 0 0 1 1 1 0
```

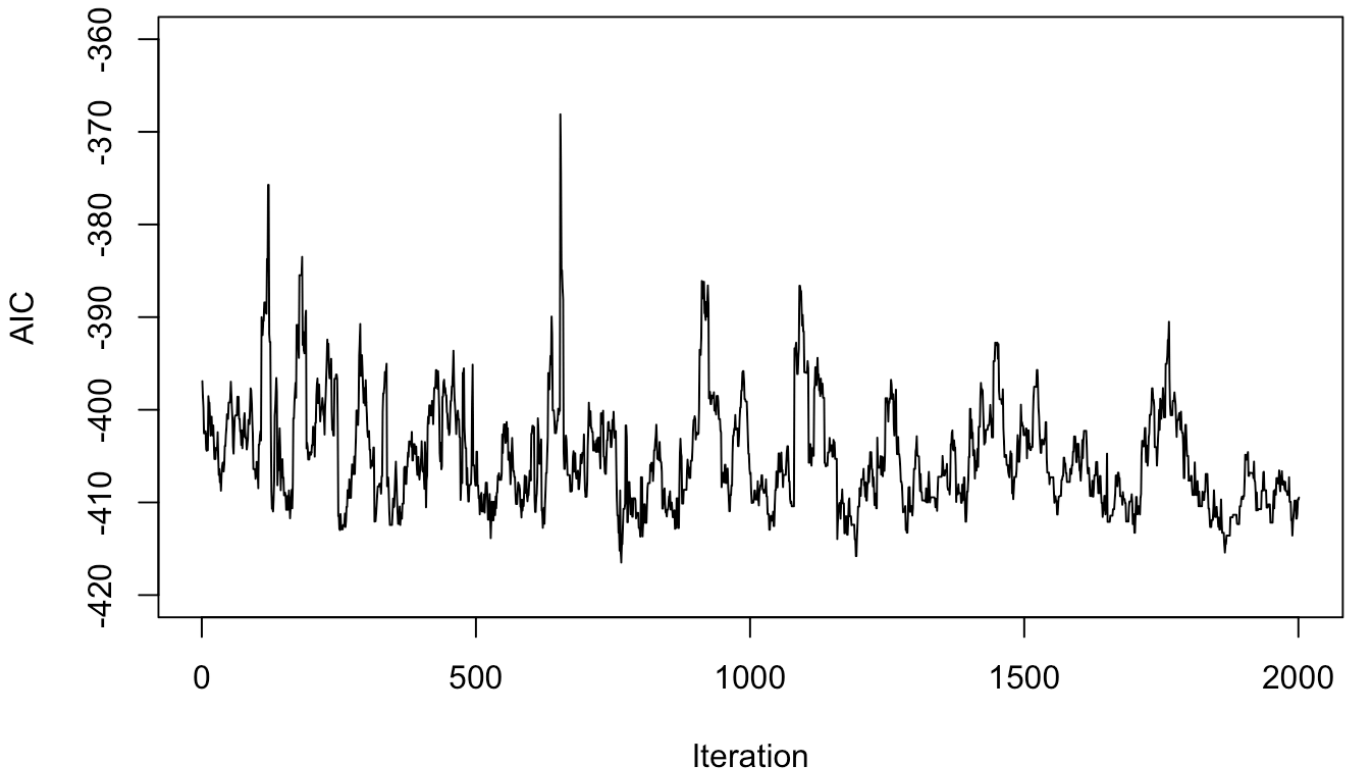
Hide

```
best.aic    # Best AIC value
```

```
[1] -416.4881
```

Hide

```
## Plot of AIC values
plot(aics, ylim=c(-420,-360), type="n", ylab="AIC", xlab="Iteration")
lines(aics)
```



Hide

```
(1:2001)[aics==min(aics)]
```

```
[1] 765
```

Hide

```
# Question 3-3
salary.log = log(baseball$salary)
m = length(baseball.sub[1,])
cooling = c(rep(60,4), rep(120,5), rep(220,6))  # compared to the previous example
3.4,
# giving different durations at each temperature gave similar results
tau_start = 10
tau = rep(tau_start, 15)
aics = NULL

set.seed(123)
run = sample(c(0,1), m, replace=TRUE)  # run == mod
run.current = run
run.vars = baseball.sub[, run.current==1]  # run.vars == base1
idx1 = c(1:m)[run==1]
fit = lm(salary.log~., run.vars)  # g == fit
run.aic = extractAIC(fit)[2]
best.aic = run.aic
aics = run.aic
for(j in 2:15) {tau[j] = 0.9*tau[j-1]}

# b. 2-neighborhoods
for(j in 1:15) {
  for(i in 1:cooling[j]) {
    run.step = run.current
    pos1 = sample(1:m, 1)
    pos2 = sample(1:m, 1)
    run.step[pos1] = 1-run.step[pos1]
    run.step[pos2] = 1-run.step[pos2]
    idx2 = c(1:m)[run.step==1]
    run.vars = baseball.sub[,run.step==1]
    fit2 = lm(salary.log~., data=run.vars)
    run.step.aic = extractAIC(fit2)[2]  #run.step.aic == AIC2
    p = min(1, exp((run.aic-extractAIC(fit2)[2])/tau[j]))
    if(run.step.aic < run.aic){
      run.current = run.step
      run.aic = run.step.aic}
    if(run.step.aic < best.aic){
      run = run.step
      best.aic = run.step.aic}
    aics = c(aics, run.aic)
  }
}

## Output
run         # Best list of predictors found
```

```
 [1] 0 0 1 0 0 1 0 1 0 1 0 0 1 1 0 0 0 0 0 1 1 1 0 1 1 0 0
```

Hide

```
best.aic    # Best AIC value
```

```
[1] -416.6119
```

Hide

```
# b. 3-neighborhoods
for(j in 1:15) {
  for(i in 1:cooling[j]) {
    run.step = run.current
    pos1 = sample(1:m, 1)
    pos2 = sample(1:m, 1)
    pos3 = sample(1:m, 1)
    run.step[pos1] = 1-run.step[pos1]
    run.step[pos2] = 1-run.step[pos2]
    run.step[pos3] = 1-run.step[pos3]
    idx2 = c(1:m)[run.step==1]
    run.vars = baseball.sub[,run.step==1]
    fit2 = lm(salary.log~., data=run.vars)
    run.step.aic = extractAIC(fit2)[2]  #run.step.aic == AIC2
    p = min(1, exp((run.aic-extractAIC(fit2)[2])/tau[j]))
    if(run.step.aic < run.aic){
      run.current = run.step
      run.aic = run.step.aic}
    if(run.step.aic < best.aic){
      run = run.step
      best.aic = run.step.aic}
    aics = c(aics, run.aic)
    }
}

## Output
run         # Best list of predictors found
```

```
 [1] 0 0 1 0 0 1 0 1 0 1 0 0 1 1 0 0 0 0 0 1 1 1 0 1 1 0 0
```

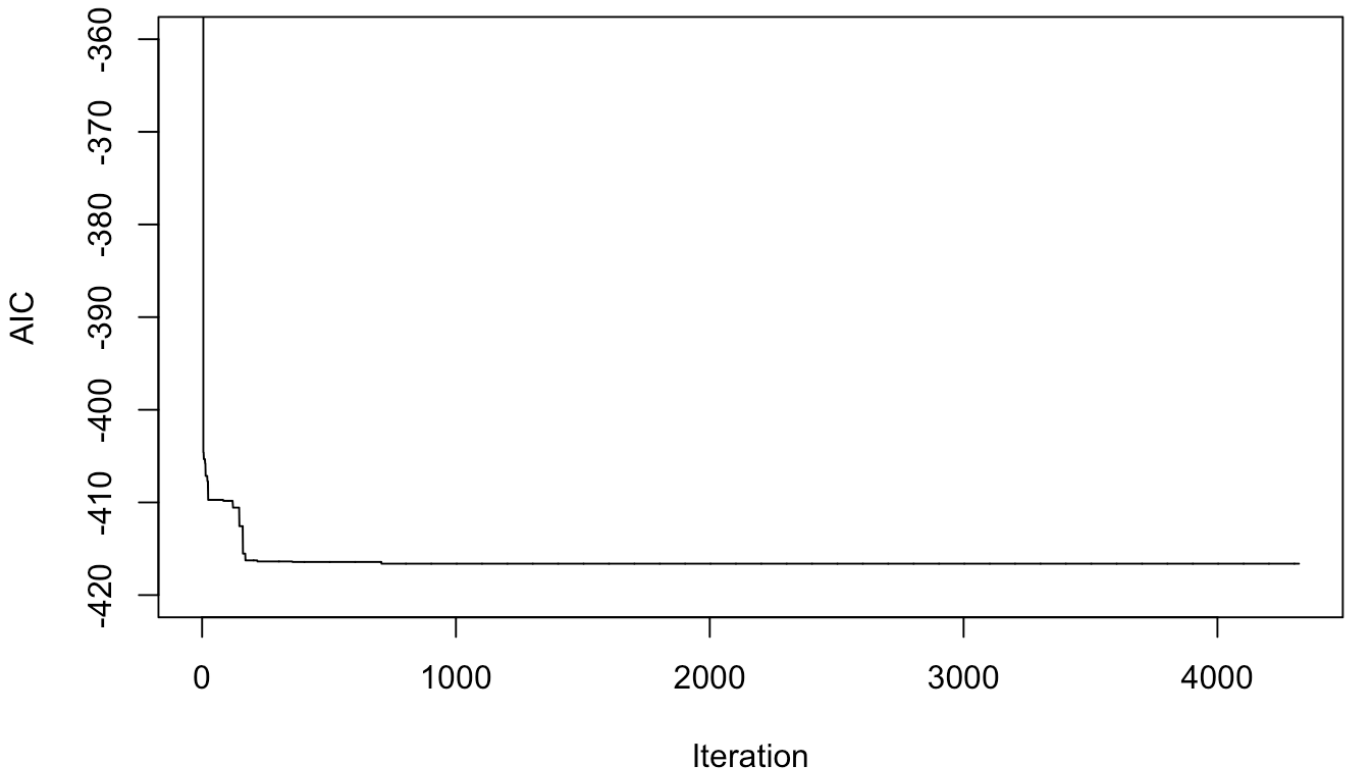
Hide

```
best.aic    # Best AIC value
```

```
[1] -416.6119
```

Hide

```
## Plot of AIC values
plot(aics, ylim=c(-420,-360), type="n", ylab="AIC", xlab="Iteration")
lines(aics)
```



Hide

```
# Example 3.5
baseball <- read.table("baseball.dat", header=TRUE)
baseball$freeagent = factor(baseball$freeagent)
baseball$arbitration = factor(baseball$arbitration)
baseball.sub = baseball[, -1]
salary.log = log(baseball$salary)
P = 20
m = ncol(baseball)-1
iter = 100
mu = 0.01
r = matrix(0, P, 1)
phi = matrix(0, P, 1)
runs = matrix(0, P, m)
runs.next = matrix(0, P, m)
runs.aic = matrix(0, P, 1)
aics = matrix(0, P, iter)
run = NULL
best.aic = 0
```

```
best.aic.gen = rep(0, iter)

## Initialization
set.seed(123)
for(i in 1:P) {
  runs[i, ] = rbinom(m, 1, 0.5)
  run.vars = baseball.sub[, runs[i,]==1]
  g = lm(salary.log~., run.vars)
  runs.aic[i] = extractAIC(g)[2]
  aics[i, 1] = runs.aic[i]
  if(runs.aic[i] < best.aic){
    run = runs[i, ]
    best.aic = runs.aic[i]
  }
}
r = rank(-runs.aic)
phi = 2*r/(P*(P+1))
best.aic.gen[1] = best.aic

# Process: one parent with probability proportional to fitness, other random
for(j in 1:iter-1) {
  for(i in 1:10){
    rownum = sample(1:P, 1, prob=phi)
    parent.1 = runs[rownum, ]
    runs1 = runs[-c(rownum), ]
    rownames(runs1) <- 1:nrow(runs1)
    parent.2 = runs1[sample(1:(P-1), 1), ]
    pos = sample(1:(m-1), 1)
    mutate = rbinom(m, 1, mu)
    runs.next[i,] = c(parent.1[1:pos], parent.2[(pos+1):m])
    runs.next[i,] = (runs.next[i,] + mutate)%%2
    mutate = rbinom(m, 1, mu)
    runs.next[P+1-i,] = c(parent.2[1:pos],parent.1[(pos+1):m])
    runs.next[P+1-i,] = (runs.next[P+1-i,] + mutate)%%2
  }
  runs = runs.next

  for(i in 1:P) {
    run.vars = baseball.sub[, runs[i,]==1]
    g = lm(salary.log~., run.vars)
    runs.aic[i] = extractAIC(g)[2]
    aics[i, j+1] = runs.aic[i]
    if(runs.aic[i] < best.aic) {
      run = runs[i,]
      best.aic = runs.aic[i]
    }
  }
  best.aic.gen[j+1] = best.aic
  r = rank(-runs.aic)
  phi = 2*r/(P*(P+1))
```

```
}

## Output
run          # Best list of predictors found
```

```
 [1] 0 1 1 0 0 1 0 1 0 1 0 0 1 1 1 1 0 0 0 0 0 0 0 1 1 1 0
```

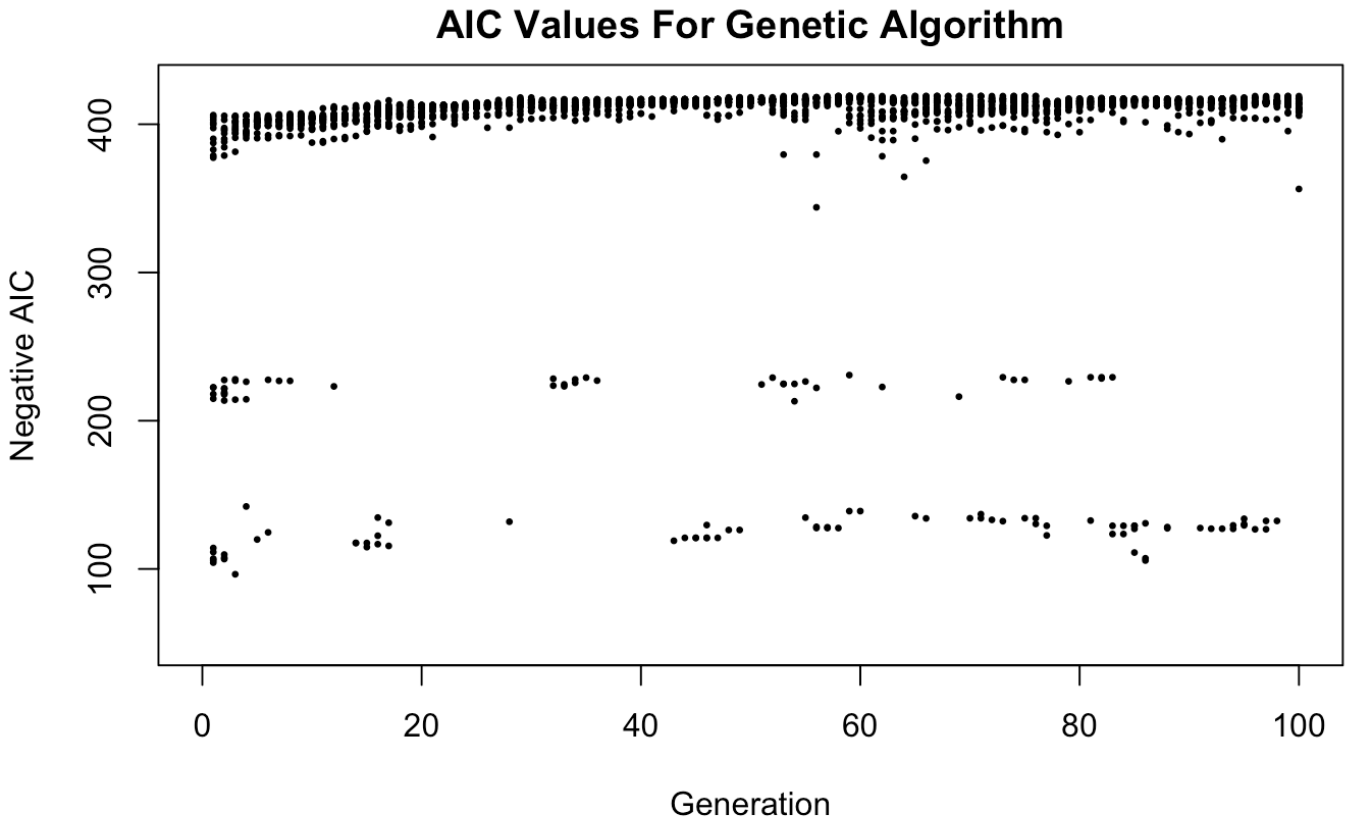
Hide

```
best.aic    # AIC value
```

```
[1] -418.9472
```

Hide

```
## Plotting AIC values
plot(-aics, xlim=c(0, iter), ylim=c(50,425), type="n", ylab="Negative AIC",
     xlab="Generation", main="AIC Values For Genetic Algorithm")
for(i in 1:iter) {points(rep(i,P), -aics[,i], pch=20, cex=0.5)}
```



Hide

```
# Question 3.4. (a),(b)
```

```
baseball <- read.table("baseball.dat", header=TRUE)
baseball$freeagent = factor(baseball$freeagent)
baseball$arbitration = factor(baseball$arbitration)
baseball.sub = baseball[, -1]
salary.log = log(baseball$salary)
m = ncol(baseball)-1
iter = 100
mu = 0.1
P = 10  # AIC plot is more spread when generation size is smaller
r = matrix(0, P, 1)
phi = matrix(0, P, 1)
runs = matrix(0, P, m)
runs.next = matrix(0, P, m)
runs.aic = matrix(0, P, 1)
aics = matrix(0, P, iter)
run = NULL
best.aic = 0
best.aic.gen = rep(0, iter)

## Initialization
set.seed(123)
for(i in 1:P) {
  runs[i, ] = rbinom(m, 1, 0.5)
  run.vars = baseball.sub[, runs[i,]==1]
  g = lm(salary.log~., run.vars)
  runs.aic[i] = extractAIC(g)[2]
  aics[i, 1] = runs.aic[i]
  if(runs.aic[i] < best.aic){
    run = runs[i, ]
    best.aic = runs.aic[i]
  }
}
r = rank(-runs.aic)
phi = 2*r/(P*(P+1))
best.aic.gen[1] = best.aic

# Process: one parent with probability proportional to fitness, other random
for(j in 1:iter-1) {
  for(i in 1:10){
    rownum = sample(1:P, 1, prob=phi)
    parent.1 = runs[rownum, ]
    runs1 = runs[-c(rownum), ]
    rownames(runs1) <- 1:nrow(runs1)
    parent.2 = runs1[sample(1:(P-1), 1), ]
    pos = sample(1:(m-1), 1)
    mutate = rbinom(m, 1, mu)
    runs.next[i,] = c(parent.1[1:pos], parent.2[(pos+1):m])
    runs.next[i,] = (runs.next[i,] + mutate)%%2
    mutate = rbinom(m, 1, mu)
    runs.next[P+1-i,] = c(parent.2[1:pos],parent.1[(pos+1):m])
```

```
    runs.next[P+1-i,] = (runs.next[P+1-i,] + mutate)%%2
  }
  runs = runs.next

  for(i in 1:P) {
    run.vars = baseball.sub[, runs[i,]==1]
    g = lm(salary.log~., run.vars)
    runs.aic[i] = extractAIC(g)[2]
    aics[i, j+1] = runs.aic[i]
    if(runs.aic[i] < best.aic) {
      run = runs[i,]
      best.aic = runs.aic[i]
    }
  }
  best.aic.gen[j+1] = best.aic
  r = rank(-runs.aic)
  phi = 2*r/(P*(P+1))
}

## Output
run         # Best list of predictors found
```

```
 [1] 0 0 1 1 0 1 0 1 0 1 1 0 1 1 0 0 0 0 0 1 1 1 0 0 1 0 0
```

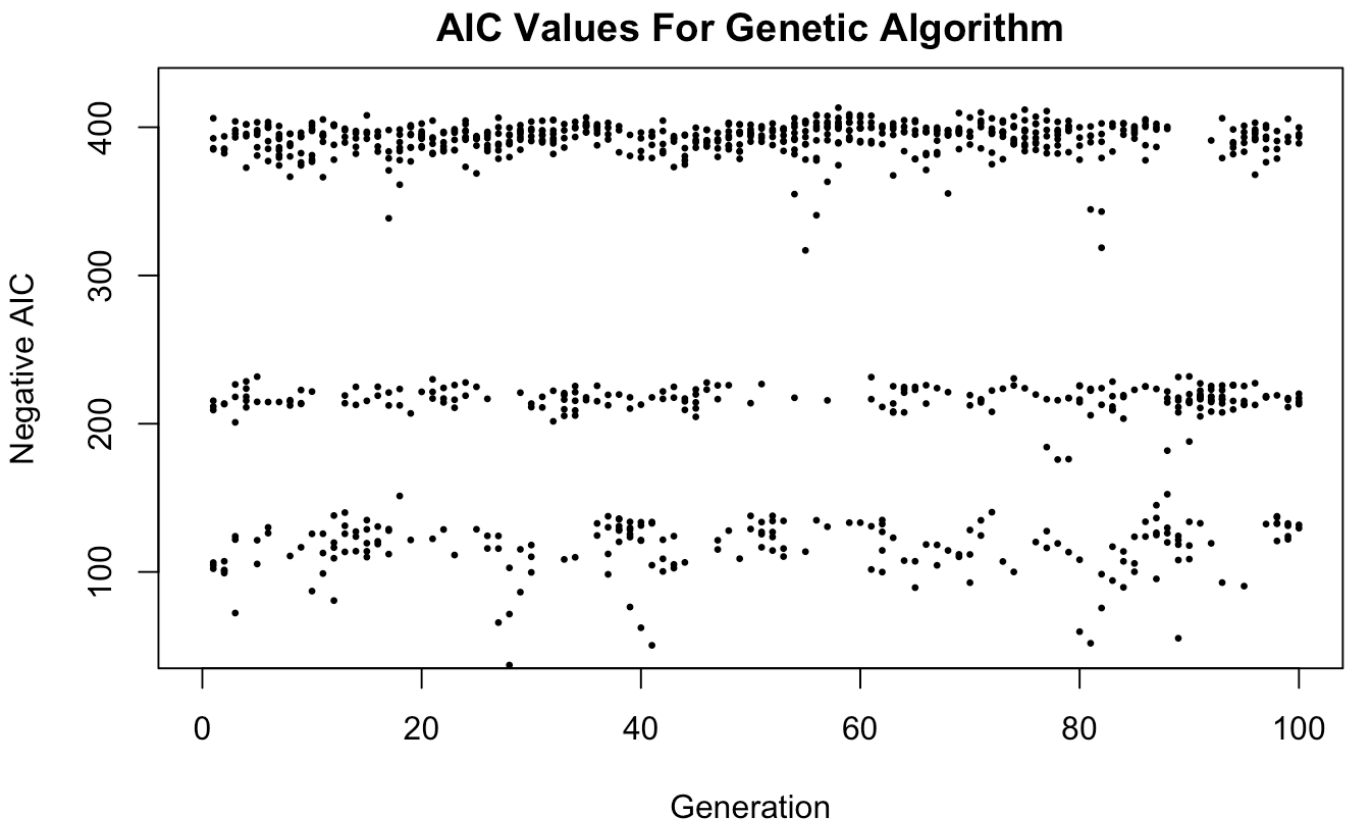
Hide

```
best.aic    # AIC value
```

```
[1] -413.1831
```

Hide

```
## Plotting AIC values
plot(-aics, xlim=c(0, iter), ylim=c(50,425), type="n", ylab="Negative AIC",
     xlab="Generation", main="AIC Values For Genetic Algorithm")
for(i in 1:iter) {points(rep(i,P), -aics[,i], pch=20, cex=0.5)}
```



Hide

```
# Question 3.4. c-i,ii
m = ncol(baseball)-1
P = 20
iter = 100
mu = 0.01   # mutation rate
r = matrix(0, P, 1)
phi = matrix(0, P, 1)
runs = matrix(0, P, m)  # pop == runs
runs.next = matrix(0, P, m)
runs.aic = matrix(0, P, 1)  # runs.aic == AIC_fit
aics = matrix(0, P, iter)
run = NULL
best.aic = 0
best.aic.gen = rep(0, iter)

## Initialization
set.seed(123)
for(i in 1:P) {
  runs[i, ] = rbinom(m, 1, 0.5)
  run.vars = baseball.sub[, runs[i,]==1]  # base1 == run.vars
  g = lm(salary.log~., run.vars)
  runs.aic[i] = extractAIC(g)[2]
  aics[i, 1] = runs.aic[i]
```

```
  if(runs.aic[i] < best.aic){
    run = runs[i, ]
    best.aic = runs.aic[i]
  }
}
r = rank(-runs.aic)
phi = 2*r/(P*(P+1))
best.aic.gen[1] = best.aic

## c-i: same as Example 3.5
## c-ii: each parent selected by probability proportional to fitness
for(j in 1:iter-1) {
  for(i in 1:10){
    rownum = sample(1:P, 1, prob=phi)
    parent.1 = runs[rownum, ]
    runs1 = runs[-c(rownum), ]
    rownames(runs1) <- 1:nrow(runs1)
    phi2 = phi[-rownum]
    parent.2 = runs1[sample(1:(P-1), 1, prob=phi2), ]
    pos = sample(1:(m-1), 1)
    mutate = rbinom(m, 1, mu)
    runs.next[i,] = c(parent.1[1:pos], parent.2[(pos+1):m])
    runs.next[i,] = (runs.next[i,] + mutate)%%2
    mutate = rbinom(m, 1, mu)
    runs.next[P+1-i,] = c(parent.2[1:pos],parent.1[(pos+1):m])
    runs.next[P+1-i,] = (runs.next[P+1-i,] + mutate)%%2
  }
  runs = runs.next

  for(i in 1:P) {
    run.vars = baseball.sub[, runs[i,]==1]
    g = lm(salary.log~., run.vars)
    runs.aic[i] = extractAIC(g)[2]
    aics[i, j+1] = runs.aic[i]
    if(runs.aic[i] < best.aic) {
      run = runs[i,]
      best.aic = runs.aic[i]
    }
  }
  best.aic.gen[j+1] = best.aic
  r = rank(-runs.aic)
  phi = 2*r/(P*(P+1))
}

## Output
run         # Best list of predictors found
```

```
 [1] 1 0 1 0 0 1 0 1 0 1 0 1 1 1 1 1 0 0 0 0 0 0 0 0 1 1 0
```

Hide

```
best.aic    # AIC value
```

```
[1] -418.0818
```

Hide

```
## Plotting AIC values
plot(-aics, xlim=c(0, iter), ylim=c(50,425), type="n", ylab="Negative AIC",
     xlab="Generation", main="AIC Values For Genetic Algorithm")
for(i in 1:iter) {points(rep(i,P), -aics[,i], pch=20, cex=0.5)}
```