Computing\_HW2\_엄상준

Eom SangJun

2020 10 27

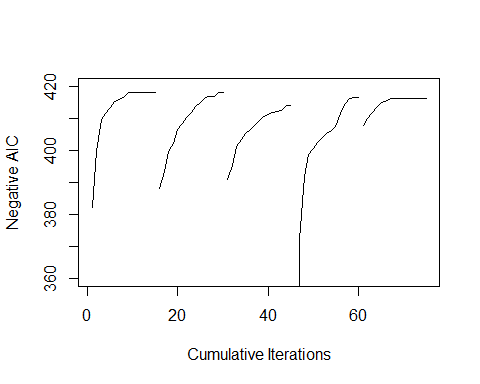
#########################################################################  
### 3.1 RANDOM STARTS LOCAL SEARCH  
#########################################################################  
## INITIAL VALUES  
baseball.dat = read.table('baseball.dat',header=TRUE)  
baseball.dat$freeagent = factor(baseball.dat$freeagent)  
baseball.dat$arbitration = factor(baseball.dat$arbitration)  
baseball.sub = baseball.dat[,-1]  
salary.log = log(baseball.dat$salary)  
n = length(salary.log) #case 개수  
m = length(baseball.sub[1,]) #독립변수 개수  
num.starts = 5 #random start 개수  
runs = matrix(0,num.starts,m)  
itr = 15  
runs.aic = matrix(0,num.starts,itr)  
  
# INITIALIZES STARTING RUNS  
set.seed(1234)   
for(i in 1:num.starts){runs[i,] = rbinom(m,1,.5)}   
#random으로 열을 뽑는 것을 다섯 번 시행  
  
  
## MAIN  
for(k in 1:num.starts){  
 run.current = runs[k,]  
  
 # ITERATES EACH RANDOM START  
 for(j in 1:itr){  
 run.vars = baseball.sub[,run.current==1] #1로 선택된 변수들 뽑아내기.  
 g = lm(salary.log~.,run.vars)  
 run.aic = extractAIC(g)[2] #[1]은 equivalent d.f, [2]가 AIC  
 run.next = run.current  
  
 # TESTS ALL MODELS IN THE 1-NEIGHBORHOOD AND SELECTS THE  
 # MODEL WITH THE LOWEST AIC  
 for(i in 1:m){  
 run.step = run.current  
 run.step[i] = !run.current[i] #0이라면 1로 1이라면 0으로 바꿈.  
 run.vars = baseball.sub[,run.step==1] #바뀐 것으로 variable을 새로 뽑음.  
 g = lm(salary.log~.,run.vars) #model 적용  
 run.step.aic = extractAIC(g)[2] #AIC 구하기  
 if(run.step.aic < run.aic){  
 run.next = run.step  
 run.aic = run.step.aic  
 } #만약 AIC가 더 작다면 run.next로 할당해줌. 그리고 run.aic도 바꿔줌  
 }  
 run.current = run.next #run.next를 current에 적용.  
 runs.aic[k,j]=run.aic   
 }  
 runs[k,] = run.current #최종적으로 제일 작은 aic를 가진 subset이 골라짐.  
}  
  
## OUTPUT  
runs # LISTS OF PREDICTORS

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

runs.aic # AIC VALUES

## [,1] [,2] [,3] [,4] [,5] [,6] [,7]  
## [1,] -382.1396 -399.5996 -409.5874 -411.5453 -413.5192 -415.3651 -415.9898  
## [2,] -388.0134 -393.5020 -399.7536 -402.2507 -406.3813 -408.3073 -410.2499  
## [3,] -391.0411 -395.3438 -401.3549 -403.3081 -405.1843 -406.4663 -407.8866  
## [4,] -199.6572 -372.1001 -391.9549 -398.5628 -400.4917 -402.3174 -404.0648  
## [5,] -407.8876 -409.8851 -411.7404 -413.4253 -414.8167 -415.3757 -416.1567  
## [,8] [,9] [,10] [,11] [,12] [,13] [,14]  
## [1,] -416.4954 -418.0000 -418.0000 -418.0000 -418.0000 -418.0000 -418.0000  
## [2,] -411.9831 -413.6061 -415.1196 -416.6207 -416.7355 -416.9711 -418.0000  
## [3,] -409.2473 -410.4997 -411.3319 -411.9675 -412.2386 -412.5352 -413.8955  
## [4,] -405.2679 -406.2594 -407.3961 -411.5770 -414.6209 -416.2200 -416.3802  
## [5,] -416.1567 -416.1567 -416.1567 -416.1567 -416.1567 -416.1567 -416.1567  
## [,15]  
## [1,] -418.0000  
## [2,] -418.0000  
## [3,] -413.8955  
## [4,] -416.3802  
## [5,] -416.1567

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



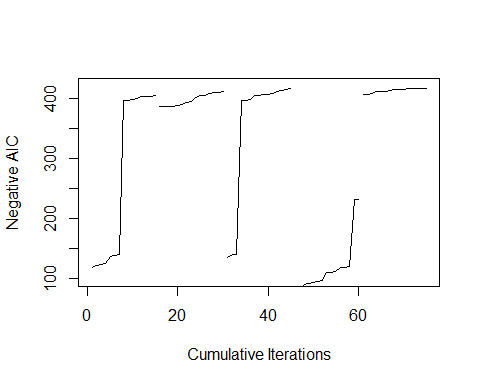
##3-1(a)  
#setting은 위와 동일  
runs2 = matrix(0,num.starts,m)  
runs.aic2 = matrix(0,num.starts,itr)  
  
# INITIALIZES STARTING RUNS  
set.seed(1234)   
for(i in 1:num.starts){runs2[i,] = rbinom(m,1,.5)}   
#random으로 열을 뽑는 것을 다섯 번 시행  
  
for(k in 1:num.starts){  
 run.current = runs2[k,]  
   
 # ITERATES EACH RANDOM START  
 for(j in 1:itr){  
 run.vars = baseball.sub[,run.current==1] #1로 선택된 변수들 뽑아내기.  
 g = lm(salary.log~.,run.vars)  
 run.aic = extractAIC(g)[2] #[1]은 equivalent d.f, [2]가 AIC  
 run.next = run.current  
   
 #aic가 작은게 나오면 바로 채택  
 for(i in 1:m){  
 run.step = run.current  
 run.step[i] = !run.current[i] #0이라면 1로 1이라면 0으로 바꿈.  
 run.vars = baseball.sub[,run.step==1] #바뀐 것으로 variable을 새로 뽑음.  
 g = lm(salary.log~.,run.vars) #model 적용  
 run.step.aic = extractAIC(g)[2] #AIC 구하기  
 if(run.step.aic < run.aic){  
 run.next = run.step  
 run.aic = run.step.aic  
 break  
 } #만약 AIC가 더 작다면 run.next로 할당해줌. 그리고 run.aic도 바꿔줌  
 }  
 run.current = run.next #run.next를 current에 적용.  
 runs.aic2[k,j]=run.aic   
 }  
 runs2[k,] = run.current #최종적으로 제일 작은 aic를 가진 subset이 골라짐.  
}  
  
runs2

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

runs.aic2

## [,1] [,2] [,3] [,4] [,5] [,6]  
## [1,] -118.87903 -120.86363 -122.74945 -124.19637 -137.24499 -137.5626  
## [2,] -385.67324 -386.28517 -386.95993 -387.20933 -388.90768 -390.4413  
## [3,] -135.02096 -139.72876 -140.44932 -396.11099 -396.21333 -398.0474  
## [4,] -79.16357 -80.88432 -90.07468 -92.01331 -93.19905 -95.1881  
## [5,] -406.45354 -406.96372 -408.92228 -411.03872 -411.17277 -412.3083  
## [,7] [,8] [,9] [,10] [,11] [,12] [,13]  
## [1,] -139.26121 -395.8276 -396.1728 -397.7137 -399.6275 -402.6213 -402.7132  
## [2,] -393.34678 -395.3418 -402.2513 -404.6487 -405.4907 -407.6709 -409.3420  
## [3,] -404.31480 -404.7419 -406.6284 -406.7591 -408.6294 -411.0614 -412.7080  
## [4,] -97.30177 -109.3067 -109.7089 -111.6231 -117.9971 -118.0441 -119.8776  
## [5,] -413.94572 -414.3590 -414.6624 -414.8042 -416.8033 -416.8033 -416.8033  
## [,14] [,15]  
## [1,] -402.7818 -404.2657  
## [2,] -409.3859 -410.8496  
## [3,] -414.5081 -415.7830  
## [4,] -230.9679 -232.1195  
## [5,] -416.8033 -416.8033

##PLOT  
plot(1:(itr\*num.starts),-c(t(runs.aic2)),xlab="Cumulative Iterations",  
 ylab="Negative AIC",ylim=c(100,420),type="n")  
for(i in 1:num.starts) {  
 lines((i-1)\*itr+(1:itr),-runs.aic2[i,]) }



##3-1(b)  
#setting은 위와 동일  
#combination function을 위해 gtools package 이용  
runs3 = matrix(0,num.starts,m)  
runs.aic3 = matrix(0,num.starts,itr)  
library(gtools)

## Warning: package 'gtools' was built under R version 3.6.3

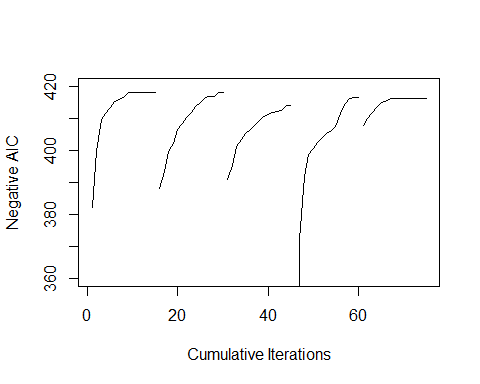
comb <-combinations(m,2) #독립변수에서 2개를 뽑는 경우의 수들  
  
# INITIALIZES STARTING RUNS  
set.seed(1234)   
for(i in 1:num.starts){runs3[i,] = rbinom(m,1,.5)}   
#random으로 열을 뽑는 것을 다섯 번 시행  
  
  
## MAIN  
for(k in 1:num.starts){  
 run.current = runs3[k,]  
   
 # ITERATES EACH RANDOM START  
 for(j in 1:itr){  
 run.vars = baseball.sub[,run.current==1] #1로 선택된 변수들 뽑아내기.  
 g = lm(salary.log~.,run.vars)  
 run.aic = extractAIC(g)[2] #[1]은 equivalent d.f, [2]가 AIC  
 run.next = run.current  
   
 # TESTS ALL MODELS IN THE 2-NEIGHBORHOOD AND SELECTS THE  
 # MODEL WITH THE LOWEST AIC  
 for(i in 1:nrow(comb)){  
 run.step = run.current  
 run.step[comb[i,][1]] = !run.current[comb[i,][1]] #0이라면 1로 1이라면 0으로 바꿈.  
 run.step[comb[i,][2]] = !run.current[comb[i,][2]]  
 run.vars = baseball.sub[,run.step==1] #바뀐 것으로 variable을 새로 뽑음.  
 g = lm(salary.log~.,run.vars) #model 적용  
 run.step.aic = extractAIC(g)[2] #AIC 구하기  
 if(run.step.aic < run.aic){  
 run.next = run.step  
 run.aic = run.step.aic  
 } #만약 AIC가 더 작다면 run.next로 할당해줌. 그리고 run.aic도 바꿔줌  
 }  
 run.current = run.next #run.next를 current에 적용.  
 runs.aic3[k,j]=run.aic   
 }  
 runs3[k,] = run.current #최종적으로 제일 작은 aic를 가진 subset이 골라짐.  
}  
  
## OUTPUT  
runs3 # LISTS OF PREDICTORS

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

runs.aic3 # AIC VALUES

## [,1] [,2] [,3] [,4] [,5] [,6] [,7]  
## [1,] -399.5996 -411.5453 -415.3651 -416.9711 -417.8494 -418.0125 -418.0922  
## [2,] -395.3894 -400.3693 -404.3826 -408.3073 -411.9831 -415.1196 -416.7355  
## [3,] -395.3438 -403.3081 -407.9346 -411.8968 -414.1540 -414.9004 -416.1567  
## [4,] -372.1001 -402.9366 -411.0083 -414.8872 -415.4118 -416.4249 -416.6119  
## [5,] -409.8851 -413.4253 -415.3757 -416.1318 -417.2714 -417.2714 -417.2714  
## [,8] [,9] [,10] [,11] [,12] [,13] [,14]  
## [1,] -418.4511 -418.4511 -418.4511 -418.4511 -418.4511 -418.4511 -418.4511  
## [2,] -418.0000 -418.9472 -418.9472 -418.9472 -418.9472 -418.9472 -418.9472  
## [3,] -416.5907 -416.8036 -418.0818 -418.9421 -418.9472 -418.9472 -418.9472  
## [4,] -416.6119 -416.6119 -416.6119 -416.6119 -416.6119 -416.6119 -416.6119  
## [5,] -417.2714 -417.2714 -417.2714 -417.2714 -417.2714 -417.2714 -417.2714  
## [,15]  
## [1,] -418.4511  
## [2,] -418.9472  
## [3,] -418.9472  
## [4,] -416.6119  
## [5,] -417.2714

##PLOT  
plot(1:(itr\*num.starts),-c(t(runs.aic3)),xlab="Cumulative Iterations",  
 ylab="Negative AIC",ylim=c(360,420),type="n")  
for(i in 1:num.starts) {  
 lines((i-1)\*itr+(1:itr),-runs.aic[i,]) }



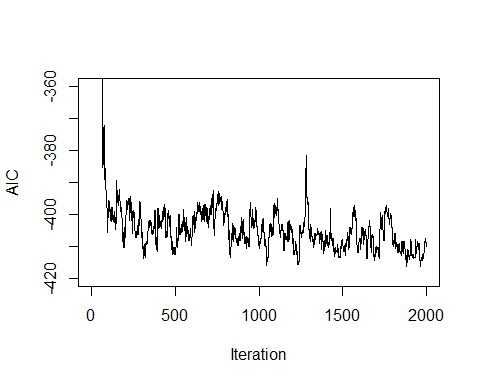
#########################################################################  
### 3.3 SIMULATED ANNEALING  
#########################################################################  
## INITIAL VALUES  
baseball.dat = read.table('baseball.dat',header=TRUE)  
baseball.dat$freeagent = factor(baseball.dat$freeagent)  
baseball.dat$arbitration = factor(baseball.dat$arbitration)  
baseball.sub = baseball.dat[,-1]  
salary.log = log(baseball.dat$salary)  
n = length(salary.log)  
m = length(baseball.sub[1,])  
cooling = c(rep(60,5),rep(120,5),rep(220,5)) #cooling schedule  
tau.start = 10 #시작 온도  
tau = rep(tau.start,15)  
aics = NULL   
  
  
# INITIALIZES STARTING RUN, TEMPERATURE SCHEDULE  
set.seed(1234)  
run = rbinom(m,1,.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 #여기까지는 3.3과 동일  
for(j in 2:15){tau[j] = 0.9\*tau[j-1]} #온도가 이전의 0.9배로 줄어들도록 설정.  
  
  
## MAIN  
for(j in 1:15){  
  
 # Model에서 더하거나 뺄 predictor를 랜덤으로 선택하고  
 # 더 나은지 확인한다. 더 낫다면 선택.  
 for(i in 1:cooling[j]){  
 pos = sample(1:m,1) #독립변수 중 한 개를 랜덤으로 선택  
 run.step = run.current  
 run.step[pos] = !run.current[pos] #선택된 독립변수를 flip  
 run.vars = baseball.sub[,run.step==1] #flip된 변수 적용  
 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)){ #만약 run.step.aic가 run.aic보다 크다면 p의 확률로 run.step을 채택  
 run.current = run.step  
 run.aic = run.step.aic}  
 if(run.step.aic < best.aic){   
 #지금까지 나온 aic중 제일 작다면 best aic로 선택하고 best predictor subset으로 설정  
 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 1 1 1 1 0 0 0 0 0 1 1 1 0 0 0 0 0

best.aic # AIC VALUE

## [1] -416.214

## PLOT OF AIC VALUES  
plot(aics,ylim=c(-420,-360),type="n",ylab="AIC", xlab="Iteration")  
lines(aics)



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

## [1] 1965

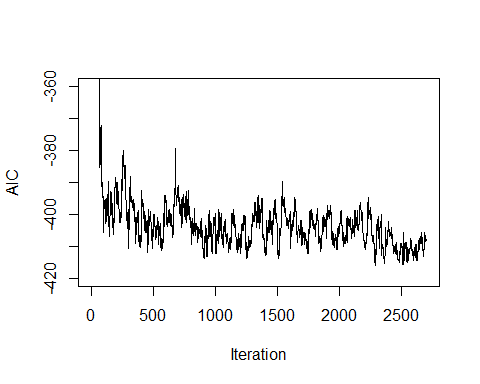
##3-3(a)  
#우선 cooling schedule에서 횟수들을 올려보자.  
baseball.dat = read.table('baseball.dat',header=TRUE)  
baseball.dat$freeagent = factor(baseball.dat$freeagent)  
baseball.dat$arbitration = factor(baseball.dat$arbitration)  
baseball.sub = baseball.dat[,-1]  
salary.log = log(baseball.dat$salary)  
n = length(salary.log)  
m = length(baseball.sub[1,])  
cooling2 = c(rep(100,5),rep(160,5),rep(280,5)) #cooling schedule  
tau.start = 10 #시작 온도  
tau = rep(tau.start,15)  
aics2 = NULL   
  
set.seed(1234)  
run2 = rbinom(m,1,.5)  
run.current = run2  
run.vars = baseball.sub[,run.current==1]  
g = lm(salary.log~.,run.vars)  
run.aic = extractAIC(g)[2]  
best.aic2 = run.aic  
aics2 = run.aic  
for(j in 2:15){tau[j] = 0.9\*tau[j-1]}   
  
for(j in 1:15){  
   
 for(i in 1:cooling2[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.aic2){   
 run2 = run.step  
 best.aic2 = run.step.aic}  
 aics2 = c(aics2,run.aic)  
 }  
}  
  
## OUTPUT  
run2 # BEST LIST OF PREDICTORS FOUND

## [1] 0 0 1 0 0 1 0 1 0 1 0 0 1 1 1 1 1 0 0 1 0 1 0 1 1 1 0

best.aic2 # AIC VALUE

## [1] -416.0132

## PLOT OF AIC VALUES  
plot(aics2,ylim=c(-420,-360),type="n",ylab="AIC", xlab="Iteration")  
lines(aics2)



(1:2701)[aics2==min(aics2)]

## [1] 2287 2288

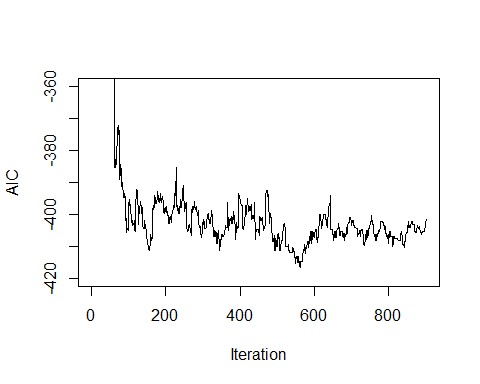
#이번에는 cooling schedule의 횟수를 낮춰보자.  
baseball.dat = read.table('baseball.dat',header=TRUE)  
baseball.dat$freeagent = factor(baseball.dat$freeagent)  
baseball.dat$arbitration = factor(baseball.dat$arbitration)  
baseball.sub = baseball.dat[,-1]  
salary.log = log(baseball.dat$salary)  
n = length(salary.log)  
m = length(baseball.sub[1,])  
cooling3 = c(rep(40,5),rep(60,5),rep(80,5)) #cooling schedule  
tau.start = 10 #시작 온도  
tau = rep(tau.start,15)  
aics3 = NULL   
  
set.seed(1234)  
run3 = rbinom(m,1,.5)  
run.current = run3  
run.vars = baseball.sub[,run.current==1]  
g = lm(salary.log~.,run.vars)  
run.aic = extractAIC(g)[2]  
best.aic3 = run.aic  
aics3 = run.aic  
for(j in 2:15){tau[j] = 0.9\*tau[j-1]}   
  
for(j in 1:15){  
   
 for(i in 1:cooling3[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.aic3){   
 run3 = run.step  
 best.aic3 = run.step.aic}  
 aics3 = c(aics3,run.aic)  
 }  
}  
  
## OUTPUT  
run3 # BEST LIST OF PREDICTORS FOUND

## [1] 1 0 1 0 0 0 0 1 0 1 1 0 1 1 1 1 1 1 0 0 0 0 0 1 1 1 0

best.aic3 # AIC VALUE

## [1] -416.5059

## PLOT OF AIC VALUES  
plot(aics3,ylim=c(-420,-360),type="n",ylab="AIC", xlab="Iteration")  
lines(aics3)



(1:901)[aics3==min(aics3)]

## [1] 561 562

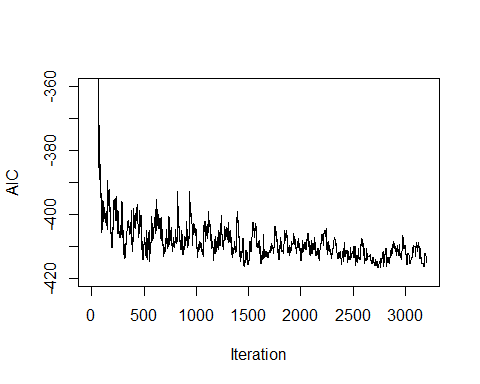
#duration을 높여보자.  
baseball.dat = read.table('baseball.dat',header=TRUE)  
baseball.dat$freeagent = factor(baseball.dat$freeagent)  
baseball.dat$arbitration = factor(baseball.dat$arbitration)  
baseball.sub = baseball.dat[,-1]  
salary.log = log(baseball.dat$salary)  
n = length(salary.log)  
m = length(baseball.sub[1,])  
cooling4 = c(rep(60,8),rep(120,8),rep(220,8)) #cooling schedule  
tau.start = 10   
tau = rep(tau.start,24)  
aics4 = NULL   
  
set.seed(1234)  
run4 = rbinom(m,1,.5)  
run.current = run4  
run.vars = baseball.sub[,run.current==1]  
g = lm(salary.log~.,run.vars)  
run.aic = extractAIC(g)[2]  
best.aic4 = run.aic  
aics4 = run.aic  
for(j in 2:24){tau[j] = 0.9\*tau[j-1]}   
  
for(j in 1:24){  
   
 for(i in 1:cooling4[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.aic4){   
 run4 = run.step  
 best.aic4 = run.step.aic}  
 aics4 = c(aics4,run.aic)  
 }  
}  
  
## OUTPUT  
run4 # BEST LIST OF PREDICTORS FOUND

## [1] 0 1 1 0 0 0 0 1 0 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 1 1 0

best.aic4 # AIC VALUE

## [1] -416.4663

## PLOT OF AIC VALUES  
plot(aics4,ylim=c(-420,-360),type="n",ylab="AIC", xlab="Iteration")  
lines(aics4)



(1:3201)[aics4==min(aics4)]

## [1] 2732 2733 2744 2745 2746 2747 2748 2760 2761

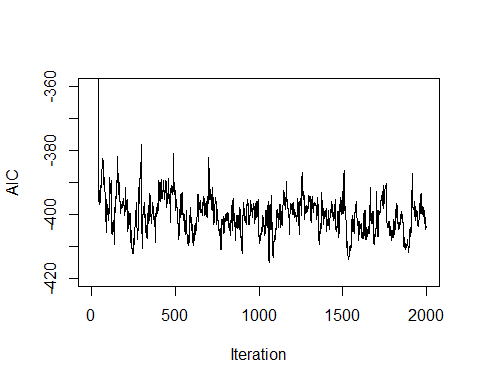
#시작 온도를 높여보자.  
baseball.dat = read.table('baseball.dat',header=TRUE)  
baseball.dat$freeagent = factor(baseball.dat$freeagent)  
baseball.dat$arbitration = factor(baseball.dat$arbitration)  
baseball.sub = baseball.dat[,-1]  
salary.log = log(baseball.dat$salary)  
n = length(salary.log)  
m = length(baseball.sub[1,])  
cooling = c(rep(60,5),rep(120,5),rep(220,5)) #cooling schedule  
tau.start = 20 #10 --> 20  
tau = rep(tau.start,15)  
aics5 = NULL   
  
set.seed(1234)  
run5 = rbinom(m,1,.5)  
run.current = run5  
run.vars = baseball.sub[,run.current==1]  
g = lm(salary.log~.,run.vars)  
run.aic = extractAIC(g)[2]  
best.aic5 = run.aic  
aics5 = 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.aic5){   
 run5 = run.step  
 best.aic5 = run.step.aic}  
 aics5 = c(aics5,run.aic)  
 }  
}  
  
## OUTPUT  
run5 # BEST LIST OF PREDICTORS FOUND

## [1] 0 1 1 0 0 0 0 1 1 1 0 1 1 1 1 1 0 1 0 0 0 0 0 0 1 1 0

best.aic5 # AIC VALUE

## [1] -415.0361

## PLOT OF AIC VALUES  
plot(aics5,ylim=c(-420,-360),type="n",ylab="AIC", xlab="Iteration")  
lines(aics5)



(1:2001)[aics5==min(aics5)]

## [1] 1060

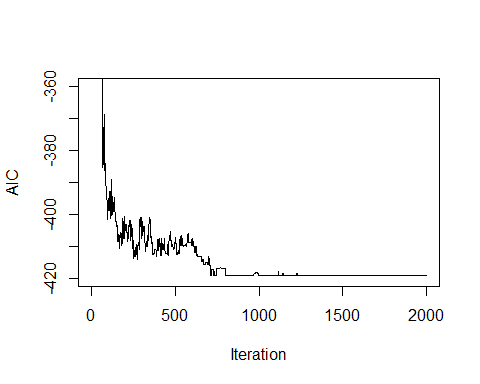
#온도의 강하율을 높여보자.  
baseball.dat = read.table('baseball.dat',header=TRUE)  
baseball.dat$freeagent = factor(baseball.dat$freeagent)  
baseball.dat$arbitration = factor(baseball.dat$arbitration)  
baseball.sub = baseball.dat[,-1]  
salary.log = log(baseball.dat$salary)  
n = length(salary.log)  
m = length(baseball.sub[1,])  
cooling = c(rep(60,5),rep(120,5),rep(220,5)) #cooling schedule  
tau.start = 10   
tau = rep(tau.start,15)  
aics6 = NULL   
  
set.seed(1234)  
run6 = rbinom(m,1,.5)  
run.current = run6  
run.vars = baseball.sub[,run.current==1]  
g = lm(salary.log~.,run.vars)  
run.aic = extractAIC(g)[2]  
best.aic6 = run.aic  
aics6 = run.aic  
for(j in 2:15){tau[j] = 0.7\*tau[j-1]} #0.9에서 0.7로 변경  
  
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.aic6){   
 run6 = run.step  
 best.aic6 = run.step.aic}  
 aics6 = c(aics6,run.aic)  
 }  
}  
  
## OUTPUT  
run6 # 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

best.aic6 # AIC VALUE

## [1] -418.9472

## PLOT OF AIC VALUES  
plot(aics6,ylim=c(-420,-360),type="n",ylab="AIC", xlab="Iteration")  
lines(aics6)



(1:2001)[aics6==min(aics6)]

## [1] 712 713 714 715 716 717 726 727

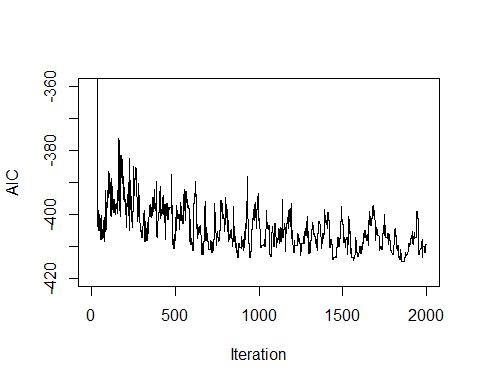
##3-3(b)  
#2-neighborhood  
baseball.dat = read.table('baseball.dat',header=TRUE)  
baseball.dat$freeagent = factor(baseball.dat$freeagent)  
baseball.dat$arbitration = factor(baseball.dat$arbitration)  
baseball.sub = baseball.dat[,-1]  
salary.log = log(baseball.dat$salary)  
n = length(salary.log)  
m = length(baseball.sub[1,])  
cooling = c(rep(60,5),rep(120,5),rep(220,5)) #cooling schedule  
tau.start = 10 #시작 온도  
tau = rep(tau.start,15)  
aics7 = NULL   
  
  
# INITIALIZES STARTING RUN, TEMPERATURE SCHEDULE(이부분은 동일)  
set.seed(1234)  
run7 = rbinom(m,1,.5)  
run.current = run7  
run.vars = baseball.sub[,run.current==1]  
g = lm(salary.log~.,run.vars)  
run.aic = extractAIC(g)[2]  
best.aic7 = run.aic  
aics7 = run.aic  
for(j in 2:15){tau[j] = 0.9\*tau[j-1]}  
  
  
## MAIN  
for(j in 1:15){  
   
 for(i in 1:cooling[j]){  
 pos = sample(1:m,2) #독립변수 중 두 개를 랜덤으로 선택  
 run.step = run.current  
 run.step[pos] = !run.current[pos] #선택된 독립변수를 flip  
 run.vars = baseball.sub[,run.step==1] #flip된 변수 적용  
 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.aic7){   
 run7 = run.step  
 best.aic7 = run.step.aic}  
 aics7 = c(aics7,run.aic)  
 }  
}  
  
## OUTPUT  
run7 # BEST LIST OF PREDICTORS FOUND

## [1] 0 1 1 0 0 1 0 1 0 1 0 1 1 1 1 1 0 0 0 1 0 0 0 0 1 0 1

best.aic7 # AIC VALUE

## [1] -414.8066

## PLOT OF AIC VALUES  
plot(aics7,ylim=c(-420,-360),type="n",ylab="AIC", xlab="Iteration")  
lines(aics7)



(1:2001)[aics7==min(aics7)]

## [1] 1852 1853 1854 1855 1856 1857 1858 1859 1860 1861 1862 1863 1864 1865 1866  
## [16] 1867 1868 1869 1870 1871 1872 1873

#3-neighborhood  
baseball.dat = read.table('baseball.dat',header=TRUE)  
baseball.dat$freeagent = factor(baseball.dat$freeagent)  
baseball.dat$arbitration = factor(baseball.dat$arbitration)  
baseball.sub = baseball.dat[,-1]  
salary.log = log(baseball.dat$salary)  
n = length(salary.log)  
m = length(baseball.sub[1,])  
cooling = c(rep(60,5),rep(120,5),rep(220,5)) #cooling schedule  
tau.start = 10 #시작 온도  
tau = rep(tau.start,15)  
aics8 = NULL   
  
  
# INITIALIZES STARTING RUN, TEMPERATURE SCHEDULE(이부분은 동일)  
set.seed(1234)  
run8 = rbinom(m,1,.5)  
run.current = run8  
run.vars = baseball.sub[,run.current==1]  
g = lm(salary.log~.,run.vars)  
run.aic = extractAIC(g)[2]  
best.aic8 = run.aic  
aics8 = run.aic  
for(j in 2:15){tau[j] = 0.9\*tau[j-1]}  
  
  
## MAIN  
for(j in 1:15){  
   
 for(i in 1:cooling[j]){  
 pos = sample(1:m,3) #독립변수 중 세 개를 랜덤으로 선택  
 run.step = run.current  
 run.step[pos] = !run.current[pos] #선택된 독립변수를 flip  
 run.vars = baseball.sub[,run.step==1] #flip된 변수 적용  
 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.aic8){   
 run8 = run.step  
 best.aic8 = run.step.aic}  
 aics8 = c(aics8,run.aic)  
 }  
}  
  
## OUTPUT  
run8 # BEST LIST OF PREDICTORS FOUND

## [1] 0 1 1 0 0 1 0 1 1 1 0 0 1 1 1 1 1 0 0 1 0 0 0 1 1 1 0

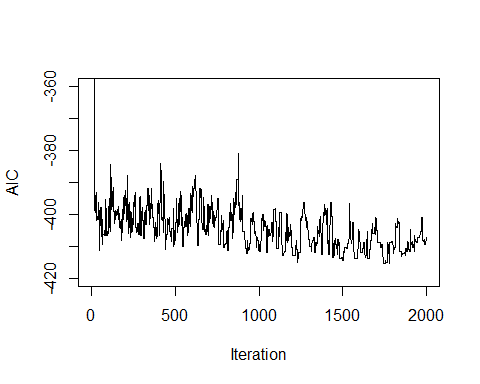
best.aic8 # AIC VALUE

## [1] -415.3731

aics8 # VECTOR OF AIC VALUES

## [1] -118.4017 -129.4263 -129.4263 -121.7759 -122.5572 -120.5938 -132.2811  
## [8] -126.1765 -120.0543 -117.4254 -116.8727 -117.8814 -119.9697 -120.5372  
## [15] -116.4129 -396.9178 -398.0827 -398.0827 -399.5227 -398.8497 -394.0259  
## [22] -394.0259 -395.4572 -398.4514 -396.0504 -396.0504 -393.2958 -401.2559  
## [29] -401.2559 -401.7740 -401.7141 -401.7141 -401.7141 -400.7977 -401.6110  
## [36] -401.6110 -401.6110 -401.6110 -401.6110 -401.6110 -401.6110 -396.2996  
## [43] -400.5089 -404.5098 -411.2005 -400.5388 -398.7913 -400.6656 -403.1093  
## [50] -403.3509 -398.0984 -397.9944 -401.1895 -397.9181 -397.9181 -401.5782  
## [57] -401.5782 -404.6997 -404.6997 -404.6997 -409.4310 -409.4310 -409.4310  
## [64] -404.4218 -401.7321 -406.3270 -406.3270 -406.3270 -406.3270 -406.3270  
## [71] -406.3270 -406.3270 -406.3270 -406.3270 -404.9460 -404.9460 -404.9460  
## [78] -406.6067 -406.6067 -406.6067 -396.3849 -400.8393 -395.3449 -395.3449  
## [85] -400.5140 -400.5140 -401.1016 -398.1467 -398.1467 -406.6901 -406.6901  
## [92] -406.3189 -406.3189 -406.3189 -406.3189 -405.4594 -405.8043 -405.8043  
## [99] -397.4568 -405.8327 -405.8327 -405.8327 -402.7571 -402.7571 -402.7571  
## [106] -394.9345 -394.9345 -400.7795 -399.4265 -405.3495 -405.3495 -396.9670  
## [113] -385.3468 -384.2564 -392.2906 -392.2906 -389.6460 -398.3973 -398.3973  
## [120] -396.2671 -396.2671 -392.8106 -392.8106 -397.5568 -397.0833 -397.0833  
## [127] -397.0833 -391.4437 -391.4437 -398.0754 -398.0754 -398.9950 -398.9950  
## [134] -398.9950 -401.9595 -401.9595 -402.9061 -401.5585 -401.5585 -400.2453  
## [141] -400.2453 -398.6835 -398.6835 -400.4356 -400.4356 -399.5016 -399.5016  
## [148] -399.5016 -399.5016 -399.5016 -398.4586 -400.0256 -400.0256 -400.0256  
## [155] -399.3459 -399.3459 -399.3459 -397.6146 -399.9638 -398.3682 -398.3682  
## [162] -401.4483 -400.2356 -399.6327 -399.6327 -403.9775 -403.9775 -403.9775  
## [169] -403.8244 -402.8580 -404.5192 -402.3719 -403.0254 -403.9584 -403.9584  
## [176] -401.6946 -401.6946 -408.1638 -408.1638 -398.4198 -400.2362 -400.2362  
## [183] -405.1655 -405.8122 -401.7315 -401.7203 -400.6233 -400.9883 -398.2544  
## [190] -398.2544 -394.7327 -399.7448 -399.7448 -397.7493 -394.1973 -395.5299  
## [197] -395.5299 -395.5299 -396.1612 -392.6540 -397.3273 -397.3273 -397.3273  
## [204] -397.3273 -397.3273 -397.3273 -397.9147 -397.9147 -402.0652 -387.8649  
## [211] -391.8016 -394.8141 -399.7665 -399.7665 -399.7665 -403.6223 -406.0644  
## [218] -396.9564 -398.6311 -404.4687 -404.4687 -404.2054 -400.7532 -399.8635  
## [225] -400.4639 -396.1604 -398.1443 -401.3244 -401.4326 -399.2134 -407.2757  
## [232] -407.2757 -403.6458 -403.6458 -399.4787 -405.5260 -405.5260 -405.5260  
## [239] -405.2228 -405.2228 -401.3697 -401.3697 -401.3697 -401.3697 -401.3697  
## [246] -403.0290 -403.0290 -403.0290 -396.3438 -399.6049 -397.6154 -397.6154  
## [253] -395.4274 -395.4274 -395.4274 -394.1211 -394.1211 -394.1211 -394.1211  
## [260] -393.2412 -393.2412 -403.9470 -403.9470 -403.9470 -403.9470 -403.9470  
## [267] -403.9470 -405.5236 -405.2237 -405.2237 -405.2237 -405.2327 -405.2327  
## [274] -402.8507 -402.8507 -402.8507 -406.3633 -406.3633 -406.2895 -406.2895  
## [281] -406.2895 -406.2895 -406.2895 -401.0398 -401.0398 -399.0410 -394.9029  
## [288] -394.4477 -403.0159 -403.0159 -405.7454 -405.7454 -406.5522 -406.5522  
## [295] -402.7257 -398.9840 -398.9840 -399.7829 -399.7829 -399.7829 -400.7226  
## [302] -400.7226 -397.8086 -398.1482 -401.9327 -401.9327 -405.5886 -405.5886  
## [309] -405.5886 -405.5886 -407.4227 -407.4227 -407.4227 -407.4227 -406.8708  
## [316] -400.0007 -399.3293 -400.1369 -400.1369 -403.2925 -398.8815 -398.8815  
## [323] -398.8815 -399.7809 -399.7809 -403.2826 -403.2826 -403.2826 -397.2913  
## [330] -397.2913 -397.2913 -393.4751 -391.9764 -391.9764 -391.9764 -391.9764  
## [337] -391.9764 -391.9764 -391.9764 -398.6990 -396.3130 -395.2548 -394.1427  
## [344] -403.1953 -403.1953 -403.1953 -401.3790 -401.3790 -400.6082 -400.6082  
## [351] -400.6082 -396.6053 -396.6053 -396.6053 -391.9009 -396.8335 -397.5232  
## [358] -397.5232 -397.5232 -398.8664 -398.8664 -396.9994 -396.9994 -401.0619  
## [365] -401.0619 -401.0619 -400.8834 -403.3797 -403.3797 -403.0748 -403.0748  
## [372] -405.1244 -406.5749 -408.4528 -408.4528 -405.7153 -405.7153 -402.5353  
## [379] -404.6317 -404.6317 -404.6317 -406.5827 -406.5827 -406.5827 -407.4116  
## [386] -407.4116 -407.4116 -404.2383 -405.0854 -405.0854 -406.8816 -406.8816  
## [393] -406.8816 -406.8816 -397.2054 -395.7570 -397.6360 -395.9251 -399.0876  
## [400] -399.0876 -399.7750 -399.0723 -399.0723 -399.0723 -406.5016 -396.9103  
## [407] -396.9103 -395.9599 -395.9599 -395.9599 -392.7876 -388.1832 -384.1684  
## [414] -388.2870 -387.3728 -387.3728 -387.3728 -386.5950 -401.5161 -402.0380  
## [421] -402.0380 -402.0380 -402.0380 -401.1006 -400.8552 -400.8552 -403.7240  
## [428] -405.6069 -405.6069 -394.8246 -389.7842 -397.2899 -397.2899 -396.9455  
## [435] -397.0881 -397.0881 -397.0881 -405.2600 -407.2984 -407.2984 -410.8669  
## [442] -408.7929 -408.7929 -407.1734 -406.1987 -406.1987 -406.1987 -406.1987  
## [449] -406.1987 -406.1987 -406.1987 -406.1987 -406.1987 -401.3586 -404.2143  
## [456] -401.4819 -401.4819 -401.8612 -401.8612 -401.8612 -403.0269 -400.8938  
## [463] -400.8938 -404.3957 -403.9125 -403.9125 -405.3268 -405.3268 -406.2749  
## [470] -406.2749 -406.2749 -406.2749 -407.0026 -407.1392 -407.1392 -407.1392  
## [477] -406.8608 -408.4124 -404.3587 -404.3587 -404.3587 -404.3587 -404.3587  
## [484] -404.3587 -404.3587 -404.3587 -398.3026 -398.3026 -402.1493 -403.5722  
## [491] -410.1443 -409.3921 -409.3921 -409.3921 -405.2803 -405.2803 -405.2803  
## [498] -404.2542 -408.1576 -405.8883 -405.1668 -406.7371 -402.4801 -402.4801  
## [505] -402.4801 -402.0745 -402.0745 -394.8758 -399.9585 -402.9467 -401.9943  
## [512] -404.3979 -405.1786 -405.7499 -405.7499 -402.6751 -403.6451 -397.3775  
## [519] -396.5728 -397.7806 -397.7806 -402.7887 -403.0141 -403.0141 -403.0141  
## [526] -403.0141 -395.0955 -395.0955 -398.9415 -398.9415 -395.9738 -392.9242  
## [533] -392.9242 -395.0319 -399.4795 -395.5556 -400.8849 -403.3621 -400.8818  
## [540] -400.8818 -409.9871 -409.9871 -409.9871 -409.9871 -409.9871 -409.9871  
## [547] -407.4350 -407.4350 -407.4350 -407.4350 -403.9390 -403.9390 -403.9390  
## [554] -403.9390 -403.9390 -403.9390 -400.0549 -400.0549 -403.8478 -403.8478  
## [561] -403.8478 -403.7442 -403.7442 -403.7442 -400.0826 -399.7460 -402.8435  
## [568] -402.8435 -400.7516 -406.5924 -400.4831 -400.4831 -400.4831 -400.8818  
## [575] -400.8818 -400.8818 -405.7735 -405.7735 -405.7735 -398.2156 -398.2156  
## [582] -398.2156 -398.1358 -402.9865 -402.9865 -402.9865 -393.3288 -395.8025  
## [589] -398.2708 -398.2708 -403.6575 -398.5851 -398.4131 -398.6399 -398.6399  
## [596] -397.3938 -394.8808 -394.0914 -396.7844 -396.8932 -399.3842 -399.0788  
## [603] -392.8109 -391.3479 -393.9508 -392.1239 -393.1033 -394.2567 -391.7982  
## [610] -391.7982 -391.7982 -391.7982 -391.7982 -388.9818 -388.9818 -391.8353  
## [617] -387.7447 -387.7447 -387.7447 -387.7447 -388.7480 -391.2769 -394.2369  
## [624] -394.2369 -394.2369 -394.4133 -394.4133 -396.4497 -408.9231 -408.9231  
## [631] -408.9231 -408.9231 -409.8335 -409.8335 -409.8335 -404.8619 -404.8619  
## [638] -404.9506 -401.6013 -401.6013 -401.6013 -401.6013 -397.8020 -397.8020  
## [645] -397.8020 -392.0541 -392.0541 -392.0541 -394.8330 -394.8330 -393.6697  
## [652] -393.6697 -392.2101 -396.0496 -394.3023 -399.0760 -400.7597 -400.7597  
## [659] -404.8478 -404.8478 -404.8478 -404.8478 -404.8478 -404.8478 -404.8478  
## [666] -397.4790 -397.4790 -394.7491 -395.5168 -395.4823 -399.1311 -399.1311  
## [673] -399.1311 -400.1438 -405.2122 -405.2122 -405.2122 -405.2122 -404.8595  
## [680] -406.2271 -406.2271 -406.2271 -406.2271 -406.2271 -403.4190 -403.4190  
## [687] -404.1274 -404.1274 -404.1274 -404.8069 -405.4301 -402.8709 -400.7464  
## [694] -397.0122 -397.0122 -398.2311 -405.5308 -406.6935 -403.1642 -403.1642  
## [701] -402.6013 -402.6013 -397.5020 -397.5020 -402.0975 -402.0975 -400.0073  
## [708] -400.0073 -401.2261 -401.2007 -400.9303 -400.9303 -393.9547 -393.9547  
## [715] -393.9547 -395.4546 -398.4444 -398.4444 -396.3180 -396.3180 -396.3180  
## [722] -398.6308 -398.6308 -402.8282 -402.8282 -403.8663 -399.5880 -399.5880  
## [729] -399.1257 -399.1257 -401.2733 -401.2733 -404.1046 -400.9034 -400.9034  
## [736] -400.9034 -400.9034 -401.5416 -401.8016 -401.0528 -401.0528 -401.0528  
## [743] -401.0528 -401.0528 -398.7494 -398.7494 -399.0268 -399.0268 -398.4723  
## [750] -395.1413 -395.1413 -395.0617 -395.2631 -395.2631 -395.1774 -402.0567  
## [757] -402.0567 -405.2301 -405.2301 -409.3468 -409.3468 -409.3468 -409.3468  
## [764] -409.3468 -409.5436 -409.5436 -409.5436 -409.5436 -409.5436 -407.7102  
## [771] -405.0182 -405.0182 -405.0182 -403.4414 -405.3866 -402.8390 -400.4363  
## [778] -400.4363 -404.8219 -404.8219 -404.8219 -404.8219 -404.8219 -404.8219  
## [785] -403.9948 -403.9948 -403.9948 -401.7984 -406.4090 -410.2619 -410.2619  
## [792] -410.2619 -410.2619 -409.5413 -409.5413 -409.5413 -409.5413 -409.5413  
## [799] -409.5413 -409.5413 -409.5413 -409.1030 -409.1030 -405.3088 -405.3088  
## [806] -406.2446 -405.8492 -405.8492 -405.8492 -405.8492 -408.1136 -406.5725  
## [813] -404.9140 -411.1461 -411.1461 -406.5835 -406.5835 -404.0247 -404.0247  
## [820] -404.0247 -404.0247 -399.2822 -399.2822 -399.2822 -399.2822 -399.2822  
## [827] -401.8530 -401.8617 -397.8690 -397.8690 -396.5359 -398.6627 -399.4510  
## [834] -399.4510 -399.4510 -401.9985 -401.9985 -401.0468 -403.2201 -404.6515  
## [841] -407.4863 -404.0772 -404.5648 -404.5648 -395.1657 -397.9207 -405.5838  
## [848] -405.5838 -396.8858 -396.8858 -396.8858 -396.2458 -400.1150 -400.1150  
## [855] -399.3916 -399.3916 -396.9794 -396.9794 -393.2267 -393.2267 -393.2267  
## [862] -395.4201 -395.4201 -395.4201 -396.8064 -393.0690 -389.0247 -389.0247  
## [869] -389.0247 -389.0247 -389.0247 -389.0247 -389.0247 -389.0247 -389.0247  
## [876] -389.0247 -380.8322 -380.8322 -390.3110 -397.2487 -397.2487 -397.3260  
## [883] -396.2646 -399.1701 -401.4217 -401.4217 -402.3378 -402.3378 -402.3378  
## [890] -402.3378 -402.3378 -398.1743 -398.1743 -398.1743 -399.2256 -399.6643  
## [897] -399.6643 -401.5055 -400.4289 -394.5959 -394.1343 -401.4219 -405.3336  
## [904] -405.0660 -405.4732 -405.4732 -405.4732 -405.4732 -405.4732 -405.4732  
## [911] -405.4732 -405.4732 -405.4732 -405.4732 -406.6039 -406.6039 -409.4805  
## [918] -409.4805 -409.4805 -409.4805 -409.4805 -410.5821 -410.5821 -412.3483  
## [925] -412.3483 -412.3483 -412.3483 -412.3483 -410.3942 -410.3942 -410.3942  
## [932] -408.0797 -411.3907 -411.3907 -411.3907 -411.3907 -411.3907 -411.3907  
## [939] -411.3907 -409.1384 -409.1384 -410.9868 -408.9393 -408.9393 -408.9393  
## [946] -408.9393 -408.9393 -408.9393 -408.9393 -408.9393 -401.3869 -401.3869  
## [953] -401.3869 -401.3869 -401.3869 -401.3869 -401.3869 -403.1631 -403.1631  
## [960] -399.8225 -402.9406 -402.9406 -400.0635 -400.0635 -399.3083 -400.0857  
## [967] -399.9747 -402.2277 -402.2277 -402.2277 -402.2277 -402.2277 -403.1580  
## [974] -405.8725 -405.8725 -405.8725 -405.8725 -405.8725 -405.8725 -405.8725  
## [981] -405.2256 -408.2530 -410.4712 -410.4712 -409.3192 -409.3192 -409.3192  
## [988] -407.1917 -407.1917 -407.1917 -407.1917 -407.1917 -408.1537 -408.1537  
## [995] -408.1537 -408.1537 -410.9353 -410.9353 -410.9353 -411.4840 -411.4840  
## [1002] -409.3632 -409.3632 -409.3632 -409.3632 -411.5779 -408.5879 -408.5879  
## [1009] -408.5879 -408.5879 -408.5879 -408.0368 -401.9895 -401.9895 -401.9895  
## [1016] -401.9895 -401.9895 -400.7648 -401.9151 -400.4812 -400.4812 -399.9366  
## [1023] -399.9366 -399.9366 -403.6677 -403.6677 -404.9944 -404.9944 -403.8768  
## [1030] -403.8768 -403.8768 -403.8768 -403.8768 -403.4310 -401.9499 -402.1708  
## [1037] -402.1708 -403.7706 -403.7706 -403.7706 -403.7706 -403.7706 -412.0166  
## [1044] -412.0166 -412.0166 -412.0166 -412.0166 -412.0166 -403.4991 -403.4991  
## [1051] -403.4991 -406.7537 -406.7537 -406.7537 -406.7537 -406.7537 -406.7537  
## [1058] -404.8085 -404.8085 -404.8085 -404.8085 -405.3426 -406.4521 -406.4521  
## [1065] -406.4521 -406.4521 -406.4521 -406.4521 -406.4521 -406.4521 -406.4521  
## [1072] -405.1337 -402.6332 -402.6332 -402.6332 -402.6332 -402.6332 -408.6231  
## [1079] -407.1043 -402.4445 -402.4445 -402.4445 -404.3033 -406.8673 -403.4533  
## [1086] -403.7353 -403.7353 -403.7353 -398.5460 -398.5460 -398.5460 -398.5460  
## [1093] -398.5460 -398.5460 -398.5460 -398.5460 -398.5460 -398.5460 -398.1300  
## [1100] -402.0916 -402.0916 -402.0916 -405.2268 -405.2268 -405.2268 -405.2268  
## [1107] -410.6072 -410.6072 -410.6072 -410.6072 -410.6072 -410.6072 -410.6072  
## [1114] -410.6072 -405.7763 -404.1173 -404.1173 -406.8963 -406.8963 -406.8963  
## [1121] -399.4424 -399.4424 -399.4424 -399.4424 -399.4424 -399.4424 -399.4424  
## [1128] -399.4424 -399.4424 -399.4424 -399.4424 -399.4424 -401.6091 -403.9583  
## [1135] -403.9583 -408.5535 -408.5535 -410.4599 -411.6291 -411.6291 -412.7511  
## [1142] -412.7511 -412.7511 -412.7511 -412.7511 -412.7511 -412.7511 -412.7511  
## [1149] -412.7511 -411.5544 -411.5544 -411.5544 -411.5544 -411.5544 -411.5544  
## [1156] -411.5544 -411.5544 -411.5544 -406.6234 -405.7896 -408.0963 -408.0963  
## [1163] -405.0567 -400.6239 -400.6239 -399.6447 -399.6447 -399.9327 -408.5284  
## [1170] -407.7682 -408.6482 -406.2812 -403.1971 -404.5688 -404.5688 -404.5688  
## [1177] -404.5688 -408.2266 -405.6197 -405.6197 -406.8564 -406.8564 -406.8564  
## [1184] -406.8564 -406.6024 -406.6024 -403.1142 -403.1142 -403.1142 -405.0189  
## [1191] -405.0413 -405.0413 -405.0413 -405.0413 -407.1145 -407.1145 -410.2832  
## [1198] -410.2832 -410.2832 -410.2832 -410.2832 -410.2832 -412.7747 -412.7747  
## [1205] -412.7747 -412.7747 -412.7747 -412.7747 -412.7747 -412.7747 -412.7747  
## [1212] -412.7747 -412.7747 -412.7747 -412.7747 -412.7747 -412.7747 -412.7747  
## [1219] -409.6181 -409.6181 -409.6181 -409.6181 -408.1019 -408.1019 -408.1019  
## [1226] -410.8318 -410.8318 -410.8318 -414.9145 -414.9145 -414.9145 -414.9145  
## [1233] -414.9145 -413.0230 -413.0230 -413.0230 -411.8816 -411.8816 -411.8816  
## [1240] -411.8816 -411.8816 -411.8816 -411.8816 -411.8816 -411.8816 -411.8816  
## [1247] -402.2099 -402.2099 -402.2099 -402.2099 -401.2616 -401.2616 -401.3151  
## [1254] -401.8233 -401.8233 -401.8233 -401.8233 -399.3220 -399.3220 -400.9414  
## [1261] -399.2085 -399.2085 -399.2085 -397.4110 -397.2423 -397.2423 -396.3865  
## [1268] -396.3865 -396.3865 -396.3865 -396.3865 -398.5030 -398.5030 -398.6886  
## [1275] -398.6886 -400.7949 -400.7949 -401.7771 -401.0992 -401.0992 -401.0992  
## [1282] -401.0992 -402.6579 -402.6579 -404.0214 -404.0214 -403.6460 -403.6460  
## [1289] -403.6460 -403.6460 -403.6460 -402.1974 -403.9660 -405.7066 -406.6971  
## [1296] -406.6971 -406.6971 -406.6971 -407.8391 -407.8391 -408.8978 -408.8978  
## [1303] -408.8978 -410.1223 -410.1223 -410.1223 -410.1223 -410.1223 -410.1223  
## [1310] -410.1223 -410.1223 -410.1223 -410.1223 -409.6867 -411.5520 -409.8537  
## [1317] -408.2296 -408.2296 -408.2296 -408.2296 -408.2296 -408.2296 -408.2296  
## [1324] -408.2296 -408.2296 -407.3229 -407.3229 -407.2819 -407.2819 -408.9485  
## [1331] -409.0289 -411.9981 -411.9981 -411.9981 -411.9981 -408.8852 -408.8852  
## [1338] -408.7765 -408.7765 -408.7765 -409.8689 -409.8689 -409.8689 -408.8047  
## [1345] -409.1673 -409.1673 -409.1673 -409.1673 -409.1673 -409.1673 -403.6936  
## [1352] -406.0427 -406.0427 -405.1842 -405.1842 -405.1842 -405.1842 -405.1842  
## [1359] -406.7627 -406.7627 -403.8624 -401.5528 -401.5528 -401.5528 -401.8626  
## [1366] -401.8626 -400.5688 -400.5688 -400.5688 -400.5688 -410.5876 -412.5081  
## [1373] -412.5081 -412.5081 -412.5081 -412.5081 -412.5081 -412.5081 -411.2184  
## [1380] -411.2184 -403.7167 -403.7167 -401.8366 -401.8366 -401.9173 -399.4773  
## [1387] -401.5964 -401.5964 -399.7609 -399.7609 -396.9694 -398.1055 -398.1055  
## [1394] -398.1055 -398.1055 -398.1055 -398.1055 -398.1055 -399.5207 -399.8885  
## [1401] -399.8885 -399.8885 -403.1634 -403.1634 -403.1634 -399.2857 -398.1258  
## [1408] -398.1258 -401.1905 -401.1905 -401.1905 -409.0834 -409.0834 -408.3042  
## [1415] -408.3042 -412.6733 -412.6733 -412.6733 -412.6733 -412.6733 -412.6733  
## [1422] -406.7139 -406.7139 -408.1824 -398.3492 -398.6690 -399.5096 -399.5096  
## [1429] -396.3161 -396.3161 -396.3161 -396.3161 -398.7118 -398.7118 -400.5335  
## [1436] -400.5335 -400.5335 -400.5335 -408.5369 -413.3255 -413.3255 -413.3255  
## [1443] -413.3255 -413.3255 -409.1935 -409.1935 -412.8879 -412.8879 -412.8879  
## [1450] -412.8879 -412.8879 -412.8879 -412.8879 -408.3998 -407.6195 -407.6195  
## [1457] -407.6195 -409.4214 -409.4214 -409.4214 -409.4214 -410.6977 -410.6977  
## [1464] -410.6977 -410.6977 -410.6977 -410.6977 -412.5491 -407.7126 -407.7126  
## [1471] -407.7126 -407.7126 -407.7032 -409.8110 -409.8110 -409.8110 -408.8950  
## [1478] -408.8950 -408.8950 -410.1033 -410.5203 -410.5203 -413.6456 -413.6456  
## [1485] -413.6456 -413.6456 -413.6456 -413.6456 -413.6456 -413.6456 -413.6456  
## [1492] -413.5795 -413.5795 -413.5795 -413.5795 -413.5795 -413.5795 -413.5795  
## [1499] -413.5795 -413.5795 -414.4177 -414.4177 -414.4177 -414.4177 -414.4177  
## [1506] -411.5650 -411.5650 -411.5650 -411.3235 -411.3235 -411.9969 -411.9969  
## [1513] -410.2081 -410.2081 -410.2081 -410.3835 -410.3835 -410.3835 -410.3835  
## [1520] -410.3835 -410.3835 -410.3835 -410.3835 -410.3835 -410.5986 -410.5986  
## [1527] -410.5986 -409.1968 -409.1968 -409.1968 -409.1968 -409.1968 -409.1968  
## [1534] -409.1968 -409.1968 -409.1968 -409.1968 -409.1968 -406.1293 -406.1293  
## [1541] -396.5544 -396.5544 -396.5544 -396.5544 -407.1882 -407.1882 -405.5045  
## [1548] -405.5045 -406.9663 -406.9663 -409.0377 -409.0377 -409.0377 -409.0377  
## [1555] -409.0377 -409.0377 -406.8214 -406.8214 -405.4101 -402.5113 -402.5113  
## [1562] -402.5113 -402.5113 -410.3296 -410.3296 -410.7753 -410.2643 -410.2643  
## [1569] -413.1945 -413.1945 -413.7459 -413.7459 -413.7459 -413.7459 -413.7459  
## [1576] -413.7459 -413.7459 -413.7459 -413.7459 -413.7459 -413.7459 -413.7459  
## [1583] -413.7459 -413.7459 -413.7459 -413.7459 -413.7459 -413.7459 -412.2837  
## [1590] -412.2837 -412.2837 -412.2837 -409.5523 -409.5523 -409.5523 -409.5523  
## [1597] -404.7059 -404.7059 -405.3488 -405.3488 -407.0524 -407.0524 -407.0524  
## [1604] -407.0524 -410.1570 -410.1570 -410.1570 -410.1570 -411.8438 -411.8438  
## [1611] -411.8438 -411.8438 -411.8438 -411.8438 -411.8438 -411.8438 -411.8438  
## [1618] -410.7661 -410.7661 -410.7661 -410.7661 -410.7661 -410.7661 -410.7661  
## [1625] -410.7661 -410.7661 -408.7170 -408.7170 -408.7170 -408.7170 -408.7170  
## [1632] -408.7170 -411.5480 -411.5480 -413.3763 -413.3763 -413.3763 -413.3763  
## [1639] -412.8820 -412.8820 -412.8820 -408.4579 -408.4579 -408.4579 -408.4579  
## [1646] -408.4579 -406.7642 -405.3409 -405.3409 -409.6875 -413.6554 -413.8796  
## [1653] -413.8796 -413.8796 -413.8796 -413.8796 -413.8796 -413.8796 -413.8796  
## [1660] -413.8796 -413.8796 -413.8796 -411.1002 -411.1002 -411.1002 -411.1002  
## [1667] -411.1002 -411.1002 -411.1002 -408.0003 -408.0003 -408.0003 -404.6822  
## [1674] -406.1573 -406.1573 -404.9454 -404.2442 -404.1918 -402.7179 -402.7179  
## [1681] -406.0645 -406.0645 -406.0645 -406.0645 -406.0645 -406.0645 -406.0645  
## [1688] -406.0645 -406.0645 -406.0645 -406.0645 -404.1279 -404.1279 -404.7870  
## [1695] -406.3081 -406.3081 -402.8513 -401.0952 -401.0952 -401.3253 -401.3253  
## [1702] -405.1526 -405.1526 -405.1526 -405.1526 -405.1526 -405.1526 -408.8889  
## [1709] -408.8889 -408.8889 -408.8889 -408.8889 -408.8889 -408.8889 -408.8889  
## [1716] -408.8889 -408.8889 -408.8889 -408.8889 -408.8889 -408.8889 -408.8889  
## [1723] -408.8889 -408.8889 -408.8889 -410.0799 -410.2525 -410.2525 -410.2525  
## [1730] -410.2525 -410.2525 -410.2525 -410.2525 -409.3992 -409.3992 -409.3992  
## [1737] -409.5939 -409.5939 -409.5939 -409.5939 -410.8947 -415.3731 -415.3731  
## [1744] -415.3731 -415.3731 -415.3731 -415.3731 -415.3731 -415.3731 -415.3731  
## [1751] -415.3731 -415.3731 -415.3731 -415.3731 -414.5388 -414.9762 -414.9762  
## [1758] -414.9762 -414.9762 -414.9762 -414.9762 -414.9762 -414.9762 -414.9762  
## [1765] -414.9762 -414.9762 -414.9762 -414.9762 -408.9192 -408.8422 -408.8422  
## [1772] -409.3832 -409.3832 -409.3832 -409.3832 -409.3832 -409.3832 -415.3194  
## [1779] -415.3194 -412.0433 -412.0433 -412.0433 -408.6322 -408.6322 -408.7221  
## [1786] -408.7221 -408.7221 -408.7221 -408.7221 -408.7221 -408.7221 -408.7221  
## [1793] -408.5653 -408.5653 -408.5653 -408.5653 -408.5653 -408.5653 -408.5653  
## [1800] -408.5653 -408.5653 -410.5415 -410.5415 -410.5415 -410.5415 -412.2389  
## [1807] -410.9255 -410.9255 -410.9255 -410.2386 -410.2386 -410.2386 -410.2386  
## [1814] -410.2386 -408.8301 -403.0516 -403.8548 -402.7415 -402.7415 -404.0293  
## [1821] -404.0293 -404.0293 -404.0293 -404.0293 -404.0293 -404.0293 -404.0293  
## [1828] -401.3848 -401.3848 -401.3848 -401.3848 -402.4814 -402.3575 -402.7848  
## [1835] -402.7848 -402.7848 -402.4620 -402.4620 -404.8484 -406.1725 -406.1725  
## [1842] -409.1423 -411.5690 -411.5690 -411.5690 -411.5690 -411.5690 -411.5690  
## [1849] -411.5690 -411.5690 -411.5690 -411.5690 -411.5690 -411.5690 -413.0430  
## [1856] -412.0866 -412.0866 -412.0866 -412.0866 -412.0866 -412.0866 -412.0866  
## [1863] -412.0866 -412.0866 -412.0866 -412.0866 -412.0866 -412.0866 -412.0866  
## [1870] -412.0866 -412.0866 -412.0866 -412.0866 -411.7024 -412.7129 -412.7129  
## [1877] -412.7129 -412.7129 -410.5949 -410.5949 -410.5949 -410.5949 -410.5949  
## [1884] -411.4545 -411.4545 -411.4545 -410.1814 -408.5837 -408.9799 -408.9799  
## [1891] -408.9799 -408.9799 -408.9799 -408.7206 -408.7206 -411.0270 -411.0270  
## [1898] -411.0270 -411.0270 -411.0270 -411.6807 -411.6807 -411.6807 -411.6807  
## [1905] -411.6807 -408.6833 -407.0166 -404.8521 -410.5103 -408.6599 -408.6599  
## [1912] -408.6599 -408.6599 -408.6599 -408.6599 -408.6599 -409.2505 -409.2505  
## [1919] -408.5809 -410.3336 -410.3336 -410.3336 -409.5475 -409.5475 -409.5475  
## [1926] -409.5475 -409.5475 -409.5475 -409.5475 -409.5475 -411.6451 -406.4911  
## [1933] -406.4911 -408.3393 -408.3393 -408.3393 -408.3393 -408.3393 -408.3393  
## [1940] -408.3393 -408.3393 -408.3393 -408.3393 -408.8531 -408.8531 -408.8531  
## [1947] -408.8531 -408.8531 -408.8531 -408.8531 -407.2248 -407.2248 -407.2248  
## [1954] -407.2248 -407.2248 -407.2248 -407.2248 -407.2248 -407.2248 -407.2248  
## [1961] -407.2248 -407.2248 -407.2248 -405.8539 -405.8539 -405.3494 -405.3494  
## [1968] -405.3494 -405.3494 -405.3494 -405.3494 -405.3494 -401.3092 -401.3092  
## [1975] -401.0483 -401.0483 -401.0483 -401.0483 -408.2637 -408.2637 -408.2637  
## [1982] -408.2637 -408.0543 -408.0543 -408.4626 -408.4626 -408.4626 -407.8064  
## [1989] -407.8064 -407.8064 -407.8064 -409.4746 -409.4746 -409.4746 -409.4746  
## [1996] -409.4746 -409.4746 -409.4746 -409.4746 -407.3190 -407.3190

## PLOT OF AIC VALUES  
plot(aics8,ylim=c(-420,-360),type="n",ylab="AIC", xlab="Iteration")  
lines(aics8)



(1:2001)[aics8==min(aics8)]

## [1] 1742 1743 1744 1745 1746 1747 1748 1749 1750 1751 1752 1753 1754

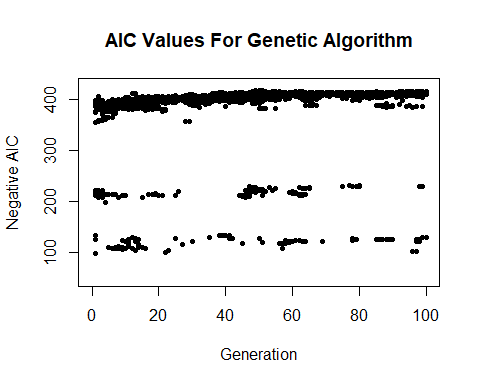
#########################################################################  
### 3.4  
#########################################################################  
## INITIAL VALUES  
baseball.dat = read.table('baseball.dat',header=TRUE)  
baseball.dat$freeagent = factor(baseball.dat$freeagent)  
baseball.dat$arbitration = factor(baseball.dat$arbitration)  
baseball.sub = baseball.dat[,-1]  
salary.log = log(baseball.dat$salary)  
n = length(salary.log)  
m = length(baseball.sub[1,])  
P = 20 #각 generation의 크기  
itr = 100 #generation을 몇 번 돌릴 것인지  
m.rate = .01 #mutation rate  
r = matrix(0,P,1) #Generation의 AIC rank  
phi = matrix(0,P,1)#Generation의 fitness values  
runs = matrix(0,P,m)  
runs.next = matrix(0,P,m)  
runs.aic = matrix(0,P,1)  
aics = matrix(0,P,itr)  
run = NULL  
best.aic = 0  
best.aic.gen = rep(0,itr)  
  
#Starting generation 설정, FITNESS VALUES  
set.seed(1234)   
for(i in 1:P){  
 runs[i,] = rbinom(m,1,.5) #random으로 variable selection  
 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) #starting genertation의 aic에 rank를 매겨주자.  
phi = 2\*r/(P\*(P+1)) #rank를 이용하여 fitness value 구해줌.  
best.aic.gen[1]=best.aic #starting genertation의 best.aic값.  
  
  
## MAIN  
for(j in 1:itr-1){  
  
 # Generation을 이어가자. 부모 중 첫 번째는 Fitness value를 기준으로 좋은 것을 뽑고  
 # 두 번째는 완전히 랜덤으로 뽑는다.   
 for(i in 1:10){  
 p1 = sample(1:P,1,prob=phi)  
 parent.1 = runs[p1,]   
 parent.2 = runs[sample(c(1:P)[-p1],1),] #중복이 되지 않도록 하자.  
 pos = sample(1:(m-1),1) #분리가 되는 지점을 정해주자.  
 mutate = rbinom(m,1,m.rate) #mutation rate에 기반해서 돌연변이가 일어나는 변수를 선택  
 runs.next[i,] = c(parent.1[1:pos],parent.2[(pos+1):m]) #다음 세대 앞 부분(돌연변이 적용 전)  
 runs.next[i,] = (runs.next[i,]+mutate)%%2 #다음 세대 앞 부분(돌연변이 적용)  
 mutate = rbinom(m,1,m.rate)  
 runs.next[P+1-i,] = c(parent.2[1:pos],parent.1[(pos+1):m]) #다음 세대 뒷 부분(돌연변이 적용 전)  
 runs.next[P+1-i,] = (runs.next[P+1-i,]+mutate)%%2 #다음 세대 뒷 부분(돌연변이 적용)  
 }  
 runs = runs.next  
  
 # New generation에서의 aic와 fitness value 업데이트.  
 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 0 0 1 0 1 0 1 0 0 1 1 1 1 0 0 0 1 0 0 0 1 1 1 0

best.aic # AIC VALUE

## [1] -416.8813

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



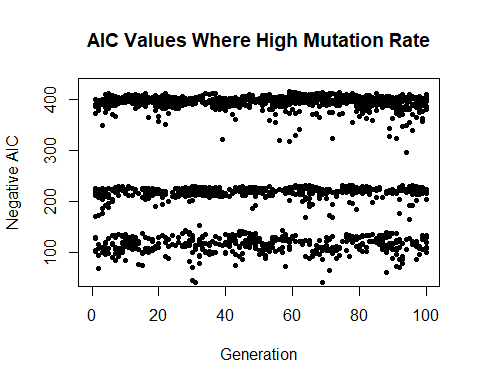
##3-4(a)  
#mutation rates를 조금 높게 설정해보자.  
baseball.dat = read.table('baseball.dat',header=TRUE)  
baseball.dat$freeagent = factor(baseball.dat$freeagent)  
baseball.dat$arbitration = factor(baseball.dat$arbitration)  
baseball.sub = baseball.dat[,-1]  
salary.log = log(baseball.dat$salary)  
n = length(salary.log)  
m = length(baseball.sub[1,])  
P = 20   
itr = 100   
m.rate = .1 #mutation rate 0.01 --> 0.1  
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,itr)  
run = NULL  
best.aic = 0  
best.aic.gen = rep(0,itr)  
  
set.seed(1234)   
for(i in 1:P){  
 runs[i,] = rbinom(m,1,.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   
  
  
for(j in 1:itr-1){  
   
 for(i in 1:10){  
 p1 = sample(1:P,1,prob=phi)  
 parent.1 = runs[p1,]   
 parent.2 = runs[sample(c(1:P)[-p1],1),]  
 pos = sample(1:(m-1),1)  
 mutate = rbinom(m,1,m.rate)  
 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,m.rate)  
 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

## [1] 1 1 1 0 0 0 0 1 0 1 0 1 1 1 1 1 0 0 0 1 0 0 0 0 1 1 0

best.aic

## [1] -414.2821

## PLOT OF AIC VALUES  
plot(-aics,xlim=c(0,itr),ylim=c(50,425),type="n",ylab="Negative AIC",  
 xlab="Generation",main="AIC Values Where High Mutation Rate")  
for(i in 1:itr){points(rep(i,P),-aics[,i],pch=20)}



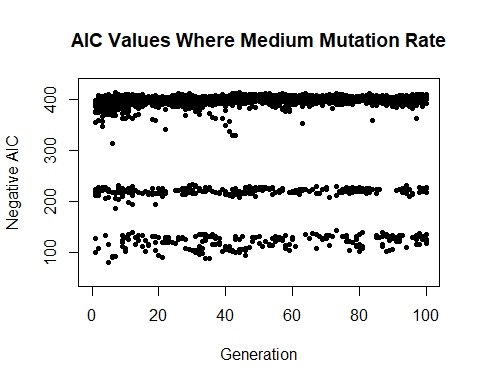
#이번에는 그 중간으로 설정  
baseball.dat = read.table('baseball.dat',header=TRUE)  
baseball.dat$freeagent = factor(baseball.dat$freeagent)  
baseball.dat$arbitration = factor(baseball.dat$arbitration)  
baseball.sub = baseball.dat[,-1]  
salary.log = log(baseball.dat$salary)  
n = length(salary.log)  
m = length(baseball.sub[1,])  
P = 20   
itr = 100   
m.rate = .05 #mutation rate 0.01 --> 0.05  
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,itr)  
run = NULL  
best.aic = 0  
best.aic.gen = rep(0,itr)  
  
set.seed(1234)   
for(i in 1:P){  
 runs[i,] = rbinom(m,1,.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   
  
  
for(j in 1:itr-1){  
   
 for(i in 1:10){  
 p1 = sample(1:P,1,prob=phi)  
 parent.1 = runs[p1,]   
 parent.2 = runs[sample(c(1:P)[-p1],1),]  
 pos = sample(1:(m-1),1)  
 mutate = rbinom(m,1,m.rate)  
 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,m.rate)  
 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

## [1] 0 1 1 1 0 0 0 1 1 1 0 0 1 1 1 1 0 0 0 1 0 0 1 1 1 1 0

best.aic

## [1] -412.5183

## PLOT OF AIC VALUES  
plot(-aics,xlim=c(0,itr),ylim=c(50,425),type="n",ylab="Negative AIC",  
 xlab="Generation",main="AIC Values Where Medium Mutation Rate")  
for(i in 1:itr){points(rep(i,P),-aics[,i],pch=20)}



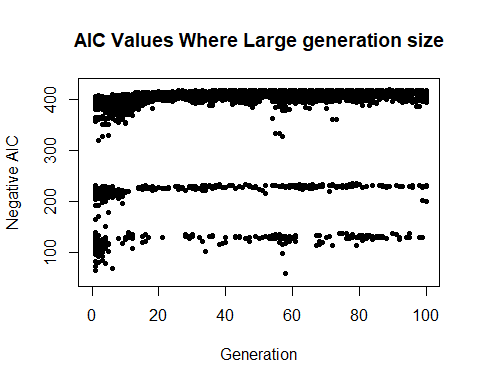
##3-4(b)  
#generation size를 굉장히 크게 해보자.  
baseball.dat = read.table('baseball.dat',header=TRUE)  
baseball.dat$freeagent = factor(baseball.dat$freeagent)  
baseball.dat$arbitration = factor(baseball.dat$arbitration)  
baseball.sub = baseball.dat[,-1]  
salary.log = log(baseball.dat$salary)  
n = length(salary.log)  
m = length(baseball.sub[1,])  
P = 80   
itr = 100   
m.rate = .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,itr)  
run = NULL  
best.aic = 0  
best.aic.gen = rep(0,itr)  
  
set.seed(1234)   
for(i in 1:P){  
 runs[i,] = rbinom(m,1,.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   
  
for(j in 1:itr-1){  
   
 for(i in 1:40){  
 p1 = sample(1:P,1,prob=phi)  
 parent.1 = runs[p1,]   
 parent.2 = runs[sample(c(1:P)[-p1],1),]  
 pos = sample(1:(m-1),1)  
 mutate = rbinom(m,1,m.rate)  
 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,m.rate)  
 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

## [1] 1 0 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

best.aic

## [1] -418.9421

## PLOT OF AIC VALUES  
plot(-aics,xlim=c(0,itr),ylim=c(50,425),type="n",ylab="Negative AIC",  
 xlab="Generation",main="AIC Values Where Large generation size")  
for(i in 1:itr){points(rep(i,P),-aics[,i],pch=20)}



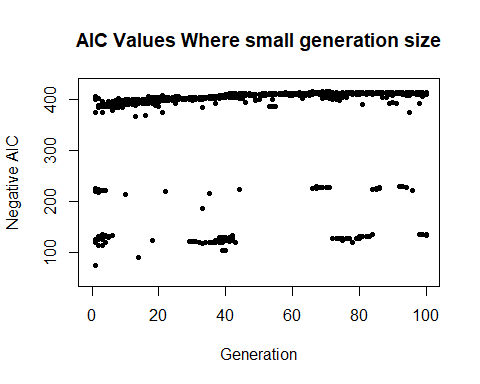
#generation size를 굉장히 작게 해보자.  
baseball.dat = read.table('baseball.dat',header=TRUE)  
baseball.dat$freeagent = factor(baseball.dat$freeagent)  
baseball.dat$arbitration = factor(baseball.dat$arbitration)  
baseball.sub = baseball.dat[,-1]  
salary.log = log(baseball.dat$salary)  
n = length(salary.log)  
m = length(baseball.sub[1,])  
P = 8   
itr = 100   
m.rate = .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,itr)  
run = NULL  
best.aic = 0  
best.aic.gen = rep(0,itr)  
  
set.seed(1234)   
for(i in 1:P){  
 runs[i,] = rbinom(m,1,.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   
  
  
for(j in 1:itr-1){  
   
 for(i in 1:4){  
 p1 = sample(1:P,1,prob=phi)  
 parent.1 = runs[p1,]   
 parent.2 = runs[sample(c(1:P)[-p1],1),]  
 pos = sample(1:(m-1),1)  
 mutate = rbinom(m,1,m.rate)  
 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,m.rate)  
 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

## [1] 0 1 1 0 0 0 0 1 0 1 0 0 1 1 1 1 1 0 0 0 0 0 0 1 0 0 0

best.aic

## [1] -414.7214

## PLOT OF AIC VALUES  
plot(-aics,xlim=c(0,itr),ylim=c(50,425),type="n",ylab="Negative AIC",  
 xlab="Generation",main="AIC Values Where small generation size")  
for(i in 1:itr){points(rep(i,P),-aics[,i],pch=20)}



##3-4(c)  
#i는 위에서 했던 방식과 동일하다.  
#ii  
baseball.dat = read.table('baseball.dat',header=TRUE)  
baseball.dat$freeagent = factor(baseball.dat$freeagent)  
baseball.dat$arbitration = factor(baseball.dat$arbitration)  
baseball.sub = baseball.dat[,-1]  
salary.log = log(baseball.dat$salary)  
n = length(salary.log)  
m = length(baseball.sub[1,])  
P = 20   
itr = 100   
m.rate = .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,itr)  
run = NULL  
best.aic = 0  
best.aic.gen = rep(0,itr)  
  
set.seed(1234)   
for(i in 1:P){  
 runs[i,] = rbinom(m,1,.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   
  
  
## MAIN  
for(j in 1:itr-1){  
   
 for(i in 1:10){  
 p1 = sample(1:P,1,prob=phi)  
 parent.1 = runs[p1,]   
 parent.2 = runs[sample(c(1:P)[-p1],1, prob=phi[-p1]),] #prob를 추가해주자.  
 pos = sample(1:(m-1),1)   
 mutate = rbinom(m,1,m.rate)   
 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,m.rate)  
 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  
   
 # New generation에서의 aic와 fitness value 업데이트.  
 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 1 1 1 1 1 0 0 0 0 0 0 0 0 1 1 0

best.aic # AIC VALUE

## [1] -418

## PLOT OF AIC VALUES  
plot(-aics,xlim=c(0,itr),ylim=c(50,425),type="n",ylab="Negative AIC",  
 xlab="Generation",main="AIC Values when both proportional")  
for(i in 1:itr){points(rep(i,P),-aics[,i],pch=20)}

