Optimization

Dennis Do

12/8/2021

1)

```
set.seed(2020)
J <- 10
p <- 0.3
A <- emails <- matrix(NA, J, J)
lambdas <- c(2, 3) ## c(lambda_0, lambda_1)
for(i in 1:(J - 1)){
for(j in (i + 1):J){
A[i, j] <- A[j, i] <- sample(c(1, 0), 1, prob = c(p, 1 - p))
emails[i, j] <- emails[j, i] <- rpois(n = 1, lambda = lambdas[A[i, j] + 1])
}
}
emails[1, ]
```

```
##  [1] NA  1  2  0  1  2  4  2  2  4
```

```
print(emails)
```

```
##       [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10]
##  [1,]   NA    1    2    0    1    2    4    2    2     4
##  [2,]    1   NA    1    0    6    1    3    2    1     6
##  [3,]    2    1   NA    1    4    1    2    5    2     3
##  [4,]    0    0    1   NA    1    1    3    0    1     2
##  [5,]    1    6    4    1   NA    2    2    3    3     1
##  [6,]    2    1    1    1    2   NA    4    2    1     0
##  [7,]    4    3    2    3    2    4   NA    3    1     1
##  [8,]    2    2    5    0    3    2    3   NA    1     1
##  [9,]    2    1    2    1    3    1    1    1   NA     1
## [10,]    4    6    3    2    1    0    1    1    1    NA
```

There are NA's because you cant email yourself, which is why there are NA's at the same numbers.

2) There are duplicates bescause it is symmetric and the values are reflected.

3)

```
sum(emails[(upper.tri(emails, diag = F))])
```

```
## [1] 90
```

There were 90 emails during the observation period

4)

```
emails_ll <- function(par = null, X = null, log = T){
out <- sum(log( ((1-par[3])* (dpois(x =X[(upper.tri(X, diag = F))], lambda = par[1]))) +
((par[3])) * (dpois(x = X[(upper.tri(X, diag = F))], lambda = par[2])))
return((out))}
```

```
emails_ll(par = c(2, 3, 0.3), X = emails)
```

```
## [1] -77.41598
```

```
emails_ll(par = c(1, 2, 0.5), X = emails)
```

```
## [1] -79.74631
```

5)

```
optim_1 <- optim(par = c(1, 2, .5), fn = function(par){
-emails_ll(X = emails, par = par)}, method = "L-BFGS-B",
lower = c(-Inf, -Inf, 0), upper = c(Inf, Inf, 1))
optim_2 <- optim(par = c(2, 3, .2), fn = function(par){
-emails_ll(X = emails, par = par)}, method = "L-BFGS-B",
lower = c(-Inf, -Inf, 0), upper = c(Inf, Inf, 1))
optim_3 <- optim(par = c(2, 5, .1), fn = function(par){
-emails_ll(X = emails, par = par)}, method = "L-BFGS-B",
lower = c(-Inf, -Inf, 0), upper = c(Inf, Inf, 1))
print(optim_1$par)
```

```
## [1] 1.8164113 3.3950213 0.1162892
```

```
print(optim_2$par)
```

```
## [1] 1.8164217 3.3952296 0.1162731
```

```
print(optim_3$par)
```

```
## [1] 1.8164309 3.3952068 0.1162742
```

6)

```
optim_4 <- optim(par = c(15, 2, .9), fn = function(par){
-emails_ll(X = emails, par = par)}, method = "L-BFGS-B", lower = c(-Inf, -Inf, 0), upper = c(Inf, Inf,
print(optim_4$par)
```

```
## [1] 14.98556  2.00000  1.00000
```

There weren't any initial values that didn't lead to converge at the global optimum

7)

```
library(GA)
```

```
## Warning: package 'GA' was built under R version 4.1.2
```

```
## Loading required package: foreach
```

```
## Warning: package 'foreach' was built under R version 4.1.2
```

```
## Loading required package: iterators
```

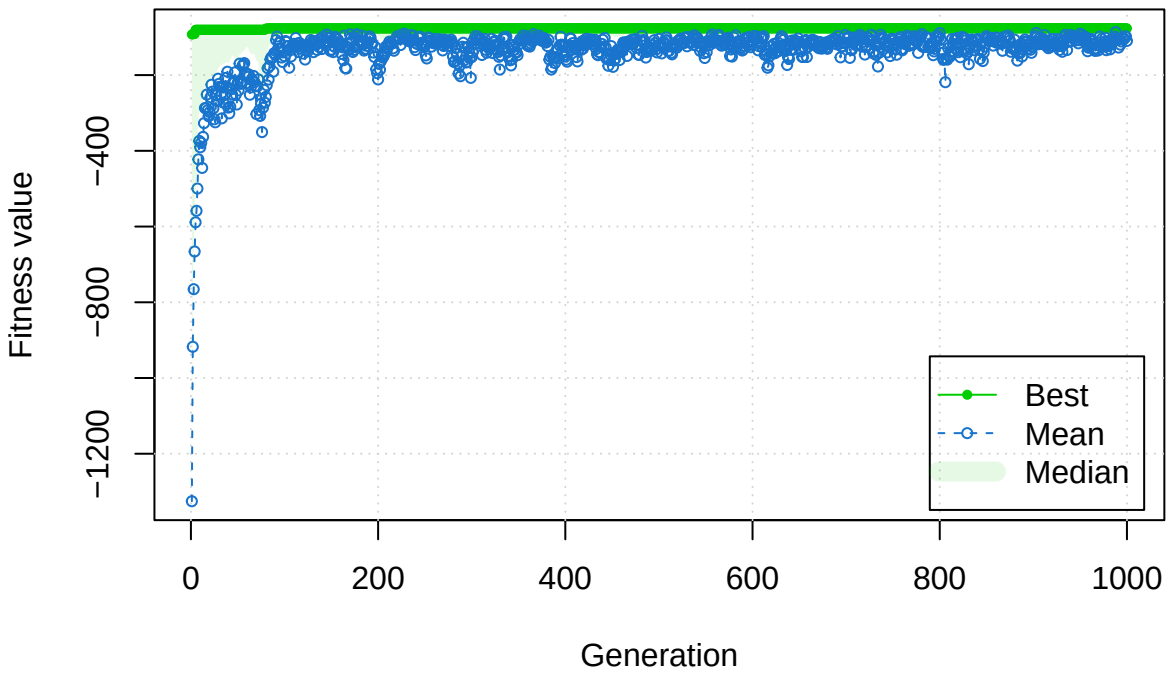
```
## Warning: package 'iterators' was built under R version 4.1.2
```

```
## Package 'GA' version 3.2.2
## Type 'citation("GA")' for citing this R package in publications.
```

```
##
## Attaching package: 'GA'
```

```
## The following object is masked from 'package:utils':
##
##     de
```

```
set.seed(120721)
par <- c(2, 3, .5)
mix_ga <- ga(type = "real-valued", fitness = function(par, X){
emails_ll(X= emails, par = par)},
X = emails, lower = c(0,0,0), upper = c(100,100, 1), popSize = 100,
maxiter = 1000, pmutation = 0.1, monitor =F)
plot(mix_ga)
```



8) Either optimization algorithms works and there is no preference, one of them is harder to understand and follow.