STAT 243 Final Group Project

Dongyu Lang Amanda Mok Kehsin Su Junyi Tang

December 14, 2017

The final version of the project resides at `https://github.com/amandamok/GA`

1 Solution description

This GA package primarily works on objects of two classes: `chromosome` and `population`. Objects of class `chromosome` contain a vector of 0's and 1's indicating a predictor's inclusion in a linear model, as well as the associated fitness (defaults to AIC) of that model. Objects of class `population` contain the original data (with the response variable as the first column), a list of `chromosome` objects that represent the models in the population at that generation, and the model with the best fitness out of that generation's population.

The following functions perform operations at the chromosome level. `initChrom()` initializes an object of class `chromosome`; if the variable inclusion vector is not provided, then the function will provide a random vector of 0's and 1's. `convertFormula()` will translate that vector into a formula object for input into `glm()`. `evalFitness()` will return the "fitness" (defaults to AIC, but can handle other functions that operate on `glm` objects) of a chromosome. `mutateChrom` will convert 0's to 1's and 1's to 0's at some probability `pMutate` at each locus on a single chromosome.

The following functions perform operations at the population level. `getFitness()` retrieves the fitness scores for a population of `chromosome` objects. `crossover()` takes two `chromosomes` and induces a "crossover" event at a random locus, producing a single offspring model. `mutatePop()` performs mutations (as dictated by `mutateChrom` at the population level. The function `nextGen` performs all the steps of each generation/iteration: removing the `pSelect`*100% models with the lowest fitness, repopulating with new `chromosome` objects using `crossover()`, and randomly performing mutations on all `chromosomes` in the population with `mutatePop()`.

Finally, a wrapper function `select()` initializes the population and performs the genetic algorithm, given user-supplied data (with the user supplying the name of the response variable), size of the population at each generation `popSize`, the proportion of the population to select off at each generation `pSelect`, the probability of mutation at an individual locus on an individual chromosome `pMutate`, the number of iterations to run the genetic algorithm `max_iter`, the fitness function to minimize `fitfunc`, and the `family` parameter to input into `glm()`. The wrapper function returns the a list with the formula, variable inclusion vector, and associated fitness of the best model after all the iterations; it also plots the best fitness and the average fitness across the population per generation.

2 Testing

Each function in the package is tested if the output is of the correct class. When appropriate, the function is tested for when the user provides the incorrect input. If the function provides a consistent result (i.e. `evalFitness` and `convertFormula`), the output of the function is compared to results if another method was used.

3 Example implementation

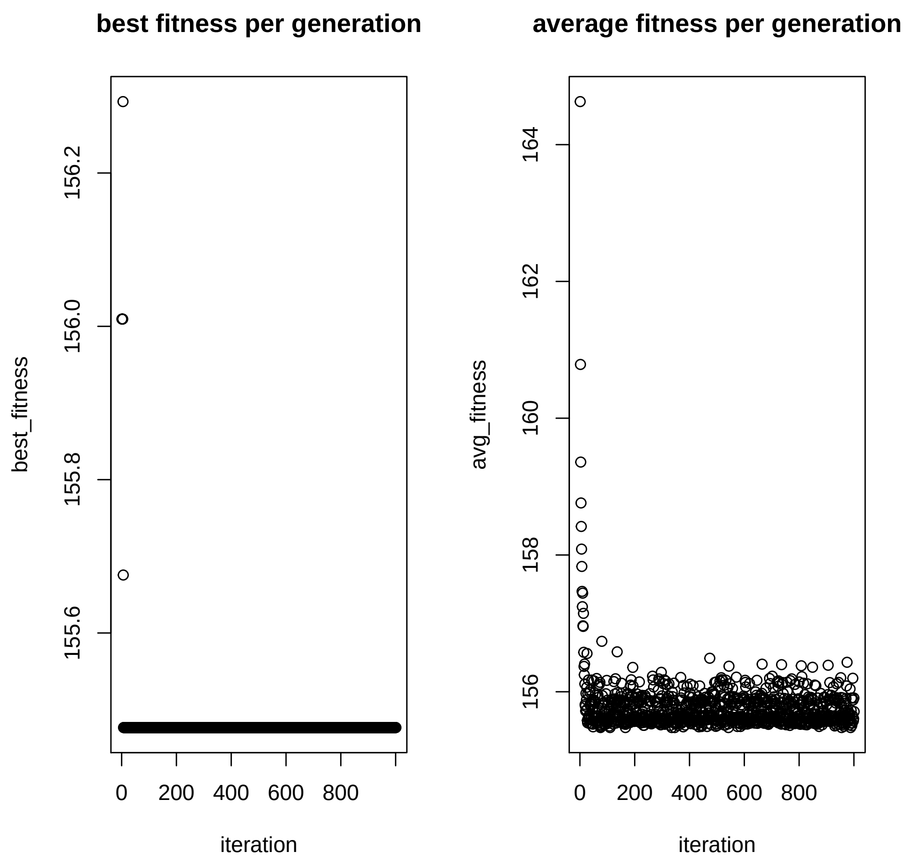
```
devtools::install_github(paste0("amandamok", "/GA"), force=T)

## Warning in strptime(x, fmt, tz = "GMT"):  unknown timezone 'default/America/Los_Angeles'
## Downloading GitHub repo amandamok/GA@master
## from URL https://api.github.com/repos/amandamok/GA/zipball/master
## Installing GA
## '/Library/Frameworks/R.framework/Resources/bin/R' --no-site-file
\
##  --no-environ --no-save --no-restore --quiet CMD INSTALL  \
##  '/private/var/folders/n2/f_58kj8x371d4syzjrhvlk1m0000gn/T/Rtmp2rmamw/devtools11cf5ae1b0c
\
##  --library='/Users/pandagoneamok/Library/R/3.4/library' --install-tests
##
## Reloading installed GA

library(GA)

set.seed(1)
results = select(mtcars, "mpg", popSize=50)

## [1] "Finished the 100th iteration."
## [1] "Finished the 200th iteration."
## [1] "Finished the 300th iteration."
## [1] "Finished the 400th iteration."
## [1] "Finished the 500th iteration."
## [1] "Finished the 600th iteration."
## [1] "Finished the 700th iteration."
## [1] "Finished the 800th iteration."
## [1] "Finished the 900th iteration."
## [1] "Finished the 1000th iteration."
## [1] "This select function reaches the number of maximum iterations.The best model selecte
```



```
results$model

## [1] "mpg  ~  cyl + hp + wt"

results$fitness

## [1] 155.4766
```

4 Contributions

Group members contributed to the final project as follows:

Dongyu Lang: package tests (initialization)

Amanda Mok: helper functions that operate on `chromosome` and `population` objects; project write-up (mutation, crossover)

Kehsin Su: package documentation (model selection)

Junyi Tang: wrapper function and plotting of results (write up and function wrap up)

Note: parentheses is the original part of works. Considering consistency and the efficiency, we finally used Ameanda's version, but all the members have written up their functions at the beginning.