Optimizing gene expression with `GeneGA`

Zhenpeng Li, Xiaochen Bo

October 17, 2016

Contents

1 Introduction

Biological engineering has driven the demand of achieving high-level expression of heterologous genes. There are many factors that can influence the gene expression, and these factors can be divided into two categories, one relating to the synonymous mutaions of gene, such as codon bias, mRNA secondary structure and the other having no relationship with synonymous mutations, such as expression vectors design, gene dosage and promoter strength. Codon bias and folding energy have been deemed as two main mechanisms of synonymous mutations to modulate the protein abundance(**?**). A recent study of expression of a diverse library of GFP gene in E.coli concluded that mRNA folding and associated rates of translation initiation play a predominant role in shaping expression levels of individual genes, whereas codon bias influences global translation efficiency and cellular fitness(**?**). Many tools have been developed to optimize gene for increasing its expression level, such as OPTIMIZER, GeneDesign and Gene Designer, while almost all of them merely consider codon bias to optimize genes. Here, we put forward a framework to optimize gene considering both codon bias and mRNA secondary structure using Genetic algorithm. The `GeneGA` package includes the information of highly expressed genes of almost 200 genomes and can be used to optimize the expression level of a gene for heterologous gene expression using rules that have been found or to explore the rules dominating gene expression.

2 Implementation

`GeneGA` uses genetic algorithm to optimize the relationship between codon bias and mRNA secondary structure. Codon adaption index(CAI) is used to quantify codon bias, which can be computed by *cai* function in `seqinr` package, while minimum free energy is used to quantify mRNA secondary structure, which can be computed by *fold* function. Certain region can be specified to optimize the relationship between codon bias and mRNA secondary structure, while codons in the other regions will be replaced by their correspondence most preference codons. Meanwhile, `GeneGA` also has the option to let the user specify the ramp region(**?**), i.e. the first 30-50 codons of genes, which has been suggested to have low translation efficiency and serve as an optimal and robust means to reduce

ribosomal traffic jams. When ramp and the specified region are intersecting, the intersectant region will be optimized to have lower CAI and higher minimum free energy, while the other region will be optimized to have higher CAI and higher minimum free energy.
The GA procedure is as follows:
1) Generating a population
At the start, the specified sequence is translated to amino acid sequence, then popSize random sequences are generated by sampling the synonymous codon of each amino acid.
2) Calculating the objective function values
Calculate the value of objective function for each member of the population.
If the ramp is not considered or the end of the ramp region does not lie in the selected region while ramp is considered:
$E = R(CAI)^2 + R(MFE)^2$,
If the end of the ramp region lies in the selected region:
$E = R(1/CAI_1)^2 + R(CAI_2)^2 + R(MFE)^2$,
If the end of the ramp region lies after the position of selected region:
$E = R(1/CAI)^2 + R(MFE)^2$,
In the formulas, R(X) represents the rank number of X in the population by increasing order. CAI and MFE denote the CAI value and minimum free energy of the member in the population respectively, while CAI_1 and CAI_2 denote the intersectant region of ramp and selected region and the region that is not intersected respectively.
3) Selection
Compute the expect number of each sequence based on the objective function values, the number of that sequence in the new population is determined by the integer part of expect number, while the digit part of expect number will be undergone roulette algorithm to determine its number in the new population.
4) Crossover
With probability crossoverRate, two member of the population exchange their sequences at random chosen point.
5) Mutation
With probability mutationChance, each codon of sequence will change its codon by random sampling from its synonymous codons.

3 Functions and examples

Users are free to choose the factors to optimize the gene. The function *GeneGA* considers both codon bias and mRNA secondary structure to optimize their relationship, *GeneFoldGA* only takes mRNA secondary structure into account and result in the largest minimum free energy of the mRNA or selected region, while *GeneCodon* merely optimizes the codon bias of gene. Detailed description of these functions can be accessed from the reference manual. Two *show* methods are provided to display the results of *GeneGA* and *GeneFoldGA*, meanwhile, two *plotGeneGA* methods can be used to visualize the variation of optimized and mean overall evaluation values and variable values during the optimizing progress. Moreover, the package also contains *wSet*, which is a data frame with 200 genomes on 64 codons. Users can also compute w table by themselves using specified highly expressed genes of given species or tissue and use the w table

by adding it to *wSet*. For example, on the assumption that "EGFP.fasta" is file containing highly expressed gene. By using the following codes, w table can be computed:

```
> library(GeneGA)
> seqfile=system.file("sequence", "EGFP.fasta", package="GeneGA")
> aa=read.fasta(seqfile)
> rscu=uco(unlist(aa), index="rscu")
> w_value=rscu # w_value is the w table we need computing
> names(w_value)=names(rscu)
> names=sapply(names(rscu), function(x) translate(s2c(x)))
> amino=hash()
> for(i in unique(names)){
+         amino[[i]]=max(rscu[which(names==i)])
+ }
> for(i in 1:length(names)){
+         w_value[i]=rscu[[names(rscu)[i]]]/amino[[translate(s2c(names(rscu)[i]))]]
+ }
```

Taking Enhanced Green Fluorescent Protein(EGFP) as an example, we use *GeneGA* to optimize EGFP by both considering its codon bias and mRNA secondary structure. The procedure is as follows:
1) Input the gene sequence. Users can input the sequence as string directly or read the sequence of fasta format, take fasta format sequence as example:

```
> seqfile=system.file("sequence", "EGFP.fasta", package="GeneGA")
> seq=unlist(getSequence(read.fasta(seqfile), as.string=TRUE))
```

2) Implementation of the *GeneGA*, the region is specified between 1 and 60. It should be noted that the designated region must be a multiple of three and in accordance with the ORF(Open reading frame) of gene. Users can also optionally add the regulatory segment before the start codon or design ramp region by using frontSeq or ramp parameters. Meanwhile, the parameters controling the Genetic algorithm processes, such as popSize, iters, crossoverRate and mutationChance, can be flexibly adjusted to archive ideal results. Generally, longer region needs larger popSize and iters, while larger crossoverRate and mutationChance can archive a sooner convergence of results.

```
> GeneGA.result=GeneGA(sequence=seq, popSize=40, iters=150, crossoverRate=0.3, mutationCha
```

3) Display the results and plot the variation of optimized and mean overall evaluation values and variable values during the optimizing progress. The *show* method will display the first three distinctive and optimum sequences, as well as their overall evaluation values, CAI values and minimum free energys. The *plotGeneGA* method can visualize the variation of optimized and mean overall evaluation values and variable values during the progress that genetic algorithm performed.

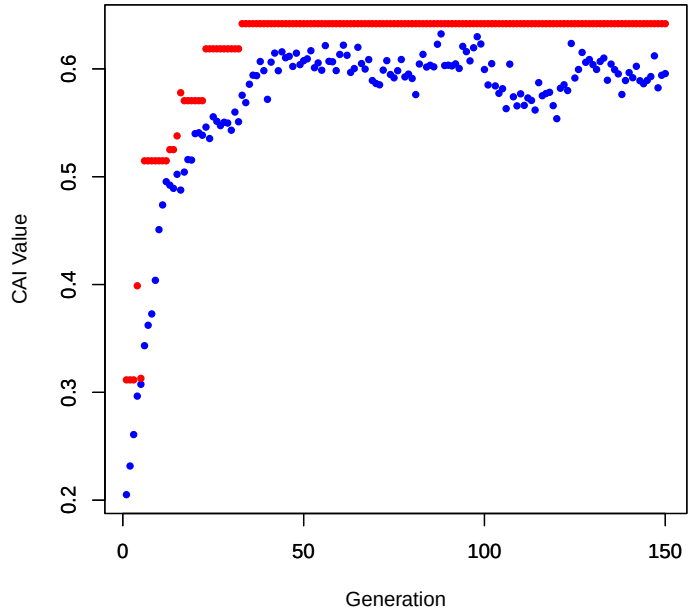
```
> show(GeneGA.result)

  GA Settings:
  Population size       = 40
  Number of Generations = 150
```

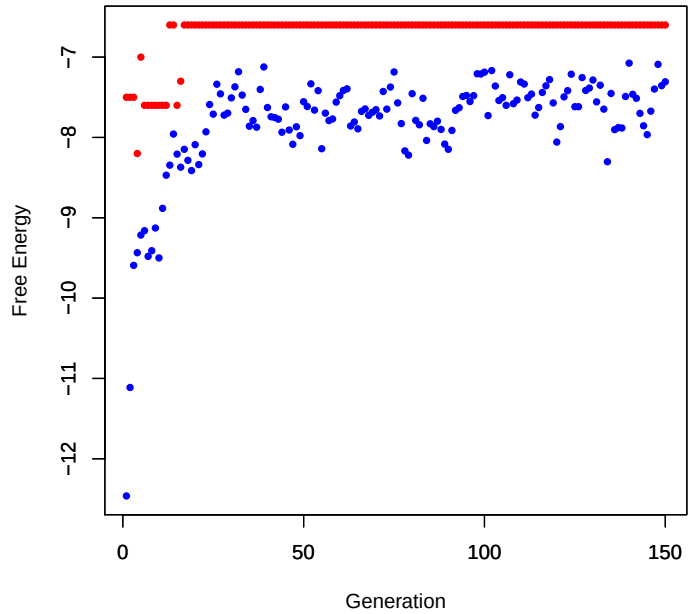
```
 crossoverRate         = 0.3
 Mutation Chance       = 0.05

evaluaton value =  2180
free energy     =  -6.6
CAI  value      =  0.6419005
ATGGTTAGTAAAGGCGAAGAACTGTTCACTGGTGTTGTTCCGATTTTGGTTGAACTGGACGGTGACGTTA
ACGGTCACAAATTCTCTGTTTCTGGTGAAGGTGAAGGTGACGCTACCTACGGTAAACTGACCCTGAAATT
CATCTGCACCACCGGTAAACTGCCGGTTCCGTGGCCGACCCTGGTTACCACCCTGACCTACGGTGTTCAG
TGCTTCTCTCGTTACCCGGACCACATGAAACAGCACGACTTCTTCAAATCTGCTATGCCGGAAGGTTACG
TTCAGGAACGTACCATCTTCTTCAAAGACGACGGTAACTACAAAACCCGTGCTGAAGTTAAATTCGAAGG
TGACACCCTGGTTAACCGTATCGAACTGAAAGGTATCGACTTCAAAGAAGACGGTAACATCCTGGGTCAC
AAACTGGAATACAACTACAACTCTCACAACGTTTACATCATGGCTGACAAACAGAAAAACGGTATCAAAG
TTAACTTCAAAATCCGTCACAACATCGAAGACGGTTCTGTTCAGCTGGCTGACCACTACCAGCAGAACAC
CCCGATCGGTGACGGTCCGGTTCTGCTGCCGGACAACCACTACCTGTCTACCCAGTCTGCTCTGTCTAAA
GACCCGAACGAAAAACGTGACCACATGGTTCTGCTGGAATTCGTTACCGCTGCTGGTATCACCCTGGGTA
TGGACGAACTGTACAAATAA
 evaluaton value =  1664
 free energy     =  -9.1
 CAI  value      =  0.7886552
 ATGGTTAGTAAAGGCGAAGAACTGTTCACTGGTGTTGTTCCGATTCTGGTTGAACTGGACGGTGACGTTA
ACGGTCACAAATTCTCTGTTTCTGGTGAAGGTGAAGGTGACGCTACCTACGGTAAACTGACCCTGAAATT
CATCTGCACCACCGGTAAACTGCCGGTTCCGTGGCCGACCCTGGTTACCACCCTGACCTACGGTGTTCAG
TGCTTCTCTCGTTACCCGGACCACATGAAACAGCACGACTTCTTCAAATCTGCTATGCCGGAAGGTTACG
TTCAGGAACGTACCATCTTCTTCAAAGACGACGGTAACTACAAAACCCGTGCTGAAGTTAAATTCGAAGG
TGACACCCTGGTTAACCGTATCGAACTGAAAGGTATCGACTTCAAAGAAGACGGTAACATCCTGGGTCAC
AAACTGGAATACAACTACAACTCTCACAACGTTTACATCATGGCTGACAAACAGAAAAACGGTATCAAAG
TTAACTTCAAAATCCGTCACAACATCGAAGACGGTTCTGTTCAGCTGGCTGACCACTACCAGCAGAACAC
CCCGATCGGTGACGGTCCGGTTCTGCTGCCGGACAACCACTACCTGTCTACCCAGTCTGCTCTGTCTAAA
GACCCGAACGAAAAACGTGACCACATGGTTCTGCTGGAATTCGTTACCGCTGCTGGTATCACCCTGGGTA
TGGACGAACTGTACAAATAA
 evaluaton value =  1604
 free energy     =  -4.7
 CAI  value      =  0.350598
 ATGGTTAGTAAAGGAGAAGAATTATTCACTGGTGTTGTTCCGATTTTGGTTGAACTCGACGGTGACGTTA
ACGGTCACAAATTCTCTGTTTCTGGTGAAGGTGAAGGTGACGCTACCTACGGTAAACTGACCCTGAAATT
CATCTGCACCACCGGTAAACTGCCGGTTCCGTGGCCGACCCTGGTTACCACCCTGACCTACGGTGTTCAG
TGCTTCTCTCGTTACCCGGACCACATGAAACAGCACGACTTCTTCAAATCTGCTATGCCGGAAGGTTACG
TTCAGGAACGTACCATCTTCTTCAAAGACGACGGTAACTACAAAACCCGTGCTGAAGTTAAATTCGAAGG
TGACACCCTGGTTAACCGTATCGAACTGAAAGGTATCGACTTCAAAGAAGACGGTAACATCCTGGGTCAC
AAACTGGAATACAACTACAACTCTCACAACGTTTACATCATGGCTGACAAACAGAAAAACGGTATCAAAG
TTAACTTCAAAATCCGTCACAACATCGAAGACGGTTCTGTTCAGCTGGCTGACCACTACCAGCAGAACAC
CCCGATCGGTGACGGTCCGGTTCTGCTGCCGGACAACCACTACCTGTCTACCCAGTCTGCTCTGTCTAAA
GACCCGAACGAAAAACGTGACCACATGGTTCTGCTGGAATTCGTTACCGCTGCTGGTATCACCCTGGGTA
TGGACGAACTGTACAAATAA
```

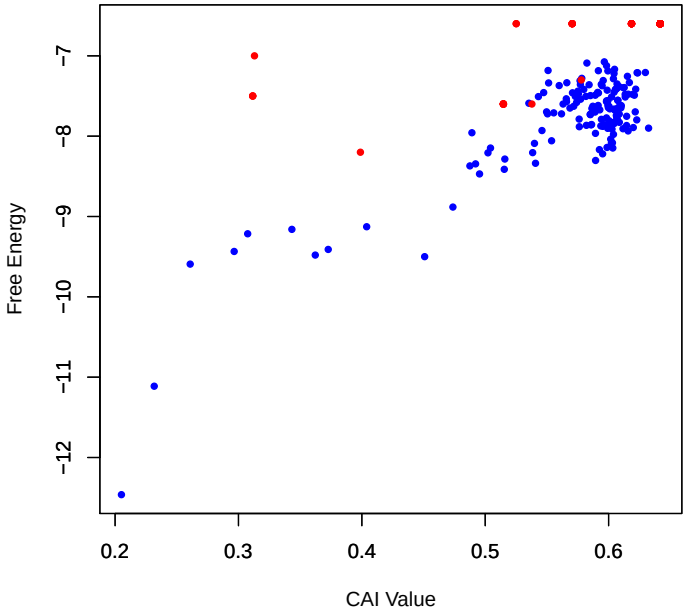
```
> plotGeneGA(GeneGA.result, type=1)
```




```
> plotGeneGA(GeneGA.result, type=2)
```




```
> plotGeneGA(GeneGA.result, type=3)
```



4 Installation notes

The `GeneGA` package depends on three other R packages: one Bioconductor package and two CRAN packages. Other than these R packages, Vienna RNA Package(http://www.tbi.univie.ac.at/ ivo/RNA/) should also be installed on your operating system.