

Package ‘dGAselID’

July 22, 2025

Type Package

Title Genetic Algorithm with Incomplete Dominance for Feature Selection

Version 1.2

Author Nicolae Teodor Melita

Maintainer Nicolae Teodor Melita <nt_melita@yahoo.com>

Description Feature selection from high dimensional data using a diploid genetic algorithm with Incomplete Dominance for genotype to phenotype mapping and Random Assortment of chromosomes approach to recombination.

Depends R (>= 3.3.1), Biobase, MLInterfaces

Imports genefilter, ALL, grDevices, graphics, stats, utils

License MIT + file LICENSE

LazyData TRUE

RoxygenNote 5.0.1

NeedsCompilation no

Repository CRAN

Date/Publication 2017-07-10 05:02:55 UTC

Contents

AnalyzeResults	2
Crossover	3
dGAselID	4
Elitism	6
EmbryonicSelection	7
EvaluationFunction	8
frameShiftMutation	9
Individuals	10
InitialPopulation	10
largeSegmentDeletion	11
nonSenseMutation	12

PlotGenAlg	13
pointMutation	13
RandomAssortment	14
RandomizePop	15
splitChromosomes	15
transposon	16
wholeChromosomeDeletion	17
Index	18

AnalyzeResults	<i>AnalyzeResults</i>
----------------	-----------------------

Description

Ranks individuals according to their fitness and records the results.

Usage

AnalyzeResults(individuals, results, randomAssortment = TRUE, chrConf)

Arguments

- individuals Population of individuals with diploid genotypes.
- results Results returned by EvaluationFunction().
- randomAssortment
 Random Assortment of Chromosomes for recombinations. The default value is TRUE.
- chrConf Configuration of chromosomes returned by splitChromosomes().

Examples

```
## Not run:
library(genefilter)
library(ALL)
data(ALL)
bALL = ALL[, substr(ALL$BT,1,1) == "B"]
smallALL = bALL[, bALL$mol.biol %in% c("BCR/ABL", "NEG")]
smallALL$mol.biol = factor(smallALL$mol.biol)
smallALL$BT = factor(smallALL$BT)
f1 <- pOverA(0.25, log2(100))
f2 <- function(x) (IQR(x) > 0.5)
f3 <- ttest(smallALL$mol.biol, p=0.1)
ff <- filterfun(f1, f2, f3)
selectedsmallALL <- genefilter(exprs(smallALL), ff)
smallALL = smallALL[selectedsmallALL, ]
rm(f1)
rm(f2)
rm(f3)
```

```

rm(ff)
rm(bALL)
sum(selectedsmallALL)
set.seed(1357)

population0<-InitialPopulation(smallALL, 14, 10, FALSE)
individuals0<-Individuals(population0)
results0<-EvaluationFunction(smallALL, individuals0, response="mol.biol",
                             method=knn.cvI(k=3, l=2), trainTest="LOG")
chrConf0<-splitChromosomes(smallALL)
iterRes0<-AnalyzeResults(individuals0, results0, randomAssortment=TRUE, chrConf0)

## End(Not run)

```

Crossover

Crossover

Description

Two-point crossover operator.

Usage

```
Crossover(c1, c2, chrConf)
```

Arguments

c1	Set of chromosomes.
c2	Set of chromosomes.
chrConf	Configuration of chromosomes returned by splitChromosomes().

Examples

```

## Not run:
library(ALL)
data(ALL)

demoALL<-ALL[1:12,1:8]
set.seed(1357)
population02<-InitialPopulation(demoALL, 2, 4, FALSE)
chrConf02<-splitChromosomes(demoALL, 2)
chrConf02
population02[1:2,]
Crossover(population02[1,], population02[2,], chrConf02)

## End(Not run)

```

dGAselID

*dGAselID***Description**

Initializes and starts the search with the genetic algorithm.

Usage

```
dGAselID(x, response, method = knn.cvI(k = 3, l = 2), trainTest = "LOG",
  startGenes, populationSize, iterations, noChr = 22, elitism = NA,
  ID = "ID1", pMutationChance = 0, nSMutationChance = 0,
  fSMutationChance = 0, lSDeletionChance = 0, wChrDeletionChance = 0,
  transposonChance = 0, randomAssortment = TRUE, embryonicSelection = NA,
  EveryGeneInInitialPopulation = TRUE, nnetSize = NA, nnetDecay = NA,
  rdaAlpha = NA, rdaDelta = NA, ...)
```

Arguments

x	Dataset in ExpressionSet format.
response	Response variable
method	Supervised classifier for fitness evaluation. Most of the supervised classifiers in MLInterfaces are acceptable. The default is knn.cvI(k=3, l=2).
trainTest	Cross-validation method. The default is "LOG".
startGenes	Genes in the genotypes at initialization.
populationSize	Number of genotypes in initial population.
iterations	Number of iterations.
noChr	Number of chromosomes. The default value is 22.
elitism	Elite population in percentages.
ID	Dominance. The default value is "ID1". Use "ID2" for Incomplete Dominance.
pMutationChance	Chance for a Point Mutation to occur. The default value is 0.
nSMutationChance	Chance for a Non-sense Mutation to occur. The default value is 0.
fSMutationChance	Chance for a Frameshift Mutation to occur. The default value is 0.
lSDeletionChance	Chance for a Large Segment Deletion to occur. The default value is 0.
wChrDeletionChance	Chance for a Whole Chromosome Deletion to occur. The default value is 0.
transposonChance	Chance for a Transposon Mutation to occur. The default value is 0.

randomAssortment	Random Assortment of Chromosomes for recombinations. The default value is TRUE.
embryonicSelection	Remove chromosomes with fitness < specified value. The default value is NA.
EveryGeneInInitialPopulation	Request for every gene to be present in the initial population. The default value is TRUE.
nnetSize	for nnetI. The default value is NA.
nnetDecay	for nnetI. The default value is NA.
rdaAlpha	for rdaI. The default value is NA.
rdaDelta	for rdaI. The default value is NA.
...	Additional arguments.

Value

The output is a list containing 5 named vectors, records of the evolution:

DGenes	The occurrences in selected genotypes for every gene,
dGenes	The occurrences in discarded genotypes for every gene,
MaximumAccuracy	Maximum accuracy in every generation,
MeanAccuracy	Average accuracy in every generation,
MinAccuracy	Minimum accuracy in every generation,
BestIndividuals	Best individual in every generation.

Examples

```
## Not run:
library(genefilter)
library(ALL)
data(ALL)
bALL = ALL[, substr(ALL$BT,1,1) == "B"]
smallALL = bALL[, bALL$mol.biol %in% c("BCR/ABL", "NEG")]
smallALL$mol.biol = factor(smallALL$mol.biol)
smallALL$BT = factor(smallALL$BT)
f1 <- pOverA(0.25, log2(100))
f2 <- function(x) (IQR(x) > 0.5)
f3 <- ttest(smallALL$mol.biol, p=0.1)
ff <- filterfun(f1, f2, f3)
selectedsmallALL <- genefilter(exprs(smallALL), ff)
smallALL = smallALL[selectedsmallALL, ]
rm(f1)
rm(f2)
rm(f3)
rm(ff)
rm(bALL)
```

```

sum(selectedsmallALL)

set.seed(149)
res<-dGAselID(smallALL, "mol.biol", trainTest=1:79, startGenes=12, populationSize=200,
              iterations=150, noChr=5, pMutationChance=0.0075, elitism=4)

## End(Not run)

```

Elitism

Elitism

Description

Operator for elitism.

Usage

```
Elitism(results, elitism, ID)
```

Arguments

results	Results returned by EvaluationFunction().
elitism	Elite population in percentages.
ID	Dominance. The default value is "ID1". Use "ID2" for Incomplete Dominance.

Examples

```

## Not run:
library(genefilter)
library(ALL)
data(ALL)
bALL = ALL[, substr(ALL$BT,1,1) == "B"]
smallALL = bALL[, bALL$mol.biol %in% c("BCR/ABL", "NEG")]
smallALL$mol.biol = factor(smallALL$mol.biol)
smallALL$BT = factor(smallALL$BT)
f1 <- pOverA(0.25, log2(100))
f2 <- function(x) (IQR(x) > 0.5)
f3 <- ttest(smallALL$mol.biol, p=0.1)
ff <- filterfun(f1, f2, f3)
selectedsmallALL <- genefilter(exprs(smallALL), ff)
smallALL = smallALL[selectedsmallALL, ]
rm(f1)
rm(f2)
rm(f3)
rm(ff)
rm(bALL)
sum(selectedsmallALL)
set.seed(1357)

```

```

population0<-InitialPopulation(smallALL, 14, 8, FALSE)
individuals0<-Individuals(population0)
results0<-EvaluationFunction(smallALL, individuals0, response="mol.biol",
                             method=knn.cvI(k=3, l=2), trainTest="LOG")
Elitism(results0, 25, ID="ID1")
Elitism(results0, 25, ID="ID2")

## End(Not run)

```

EmbryonicSelection	<i>EmbryonicSelection</i>
--------------------	---------------------------

Description

Function for deleting individuals with a fitness below a specified threshold.

Usage

```
EmbryonicSelection(population, results, embryonicSelection)
```

Arguments

population	Population of individuals with diploid genotypes.
results	Results returned by EvaluationFunction().
embryonicSelection	Threshold value. The default value is NA.

Examples

```

## Not run:
library(genefilter)
library(ALL)
data(ALL)
bALL = ALL[, substr(ALL$BT,1,1) == "B"]
smallALL = bALL[, bALL$mol.biol %in% c("BCR/ABL", "NEG")]
smallALL$mol.biol = factor(smallALL$mol.biol)
smallALL$BT = factor(smallALL$BT)
f1 <- pOverA(0.25, log2(100))
f2 <- function(x) (IQR(x) > 0.5)
f3 <- ttest(smallALL$mol.biol, p=0.1)
ff <- filterfun(f1, f2, f3)
selectedsmallALL <- genefilter(exprs(smallALL), ff)
smallALL = smallALL[selectedsmallALL, ]
rm(f1)
rm(f2)
rm(f3)
rm(ff)
rm(bALL)
sum(selectedsmallALL)

```

```

set.seed(1357)

population0<-InitialPopulation(smallALL, 14, 8, FALSE)
individuals0<-Individuals(population0)
results0<-EvaluationFunction(smallALL, individuals0, response="mol.biol",
                             method=knn.cvI(k=3, l=2), trainTest="LOG")
EmbryonicSelection(individuals0, results0, 0.5)

## End(Not run)

```

EvaluationFunction	<i>EvaluationFunction</i>
--------------------	---------------------------

Description

Evaluates the individuals' fitnesses.

Usage

```

EvaluationFunction(x, individuals, response, method, trainTest, nnetSize = NA,
                  nnetDecay = NA, rdaAlpha = NA, rdaDelta = NA, ...)

```

Arguments

<code>x</code>	Dataset in ExpressionSet format.
<code>individuals</code>	Population of individuals with diploid genotypes.
<code>response</code>	Response variable.
<code>method</code>	Supervised classifier for fitness evaluation. Most of the supervised classifiers in MLInterfaces are acceptable. The default is <code>knn.cvI(k=3, l=2)</code> .
<code>trainTest</code>	Cross-validation method. The default is "LOG".
<code>nnetSize</code>	for <code>nnetI</code> . The default value is NA.
<code>nnetDecay</code>	for <code>nnetI</code> . The default value is NA.
<code>rdaAlpha</code>	for <code>rdaI</code> . The default value is NA.
<code>rdaDelta</code>	for <code>rdaI</code> . The default value is NA.
<code>...</code>	Additional arguments.

Examples

```

## Not run:
library(genefilter)
library(ALL)
data(ALL)
bALL = ALL[, substr(ALL$BT,1,1) == "B"]
smallALL = bALL[, bALL$mol.biol %in% c("BCR/ABL", "NEG")]
smallALL$mol.biol = factor(smallALL$mol.biol)
smallALL$BT = factor(smallALL$BT)

```



```

f1 <- pOverA(0.25, log2(100))
f2 <- function(x) (IQR(x) > 0.5)
f3 <- ttest(smallALL$mol.biol, p=0.1)
ff <- filterfun(f1, f2, f3)
selectedsmallALL <- genefilter(exprs(smallALL), ff)
smallALL = smallALL[selectedsmallALL, ]
rm(f1)
rm(f2)
rm(f3)
rm(ff)
rm(bALL)
sum(selectedsmallALL)
set.seed(1357)

population0<-InitialPopulation(smallALL, 14, 8, FALSE)
individuals0<-Individuals(population0)
results<-EvaluationFunction(smallALL, individuals0, response="mol.biol",
                           method=knn.cvI(k=3, l=2), trainTest="LOG")

## End(Not run)

```

frameShiftMutation	<i>frameShiftMutation</i>
--------------------	---------------------------

Description

Operator for the frameshift mutation.

Usage

```
frameShiftMutation(individuals, chrConf, mutationChance)
```

Arguments

individuals	dataset returned by Individuals().
chrConf	Configuration of chromosomes returned by splitChromosomes().
mutationChance	Chance for a frameshift mutation to occur.

Examples

```

## Not run:
library(ALL)
data(ALL)

demoALL<-ALL[1:12,1:8]

set.seed(1234)
population<-InitialPopulation(demoALL, 4, 9)
individuals<-Individuals(population)

```

```
chrConf<-splitChromosomes(demoALL, 2)
chrConf
individuals

set.seed(123)
frameShiftMutation(individuals, chrConf, 20)

## End(Not run)
```

Individuals	<i>Individuals</i>
-------------	--------------------

Description

Generates individuals with diploid genotypes.

Usage

Individuals(population)

Arguments

population Population of haploid genotypes.

Examples

```
## Not run:
library(ALL)
data(ALL)

demoALL<-ALL[1:12,1:8]

population02<-InitialPopulation(demoALL, 20, 4, FALSE)
individuals02<-Individuals(population02)

## End(Not run)
```

InitialPopulation	<i>InitialPopulation</i>
-------------------	--------------------------

Description

Generates an initial randomly generated population of haploid genotypes.

Usage

```
InitialPopulation(x, populationSize, startGenes,
  EveryGeneInInitialPopulation = TRUE)
```

Arguments

x Dataset in ExpressionSet format.
populationSize Number of genotypes in initial population.
startGenes Genes in the genotypes at initialization.
EveryGeneInInitialPopulation Request for every gene to be present in the initial population. The default value is TRUE.

Examples

```
## Not run:
library(ALL)
data(ALL)

demoALL<-ALL[1:12,1:8]

population01<-InitialPopulation(demoALL, 4, 4)
population02<-InitialPopulation(demoALL, 20, 4, FALSE)

## End(Not run)
```

largeSegmentDeletion *largeSegmentDeletion*

Description

Operator for the large segment deletion.

Usage

```
largeSegmentDeletion(individuals, chrConf, mutationChance)
```

Arguments

individuals dataset returned by Individuals().
chrConf Configuration of chromosomes returned by splitChromosomes().
mutationChance Chance for a large segment deletion mutation to occur.

Examples

```
## Not run:
library(ALL)
data(ALL)

demoALL<-ALL[1:12,1:8]

set.seed(1234)
```

```

population<-InitialPopulation(demoALL, 4, 9)
individuals<-Individuals(population)

chrConf<-splitChromosomes(demoALL, 2)
chrConf
individuals

set.seed(123)
largeSegmentDeletion(individuals, chrConf, 20)

## End(Not run)

```

nonSenseMutation	<i>nonSenseMutation</i>
------------------	-------------------------

Description

Operator for the nonsense mutation.

Usage

```
nonSenseMutation(individuals, chrConf, mutationChance)
```

Arguments

individuals dataset returned by Individuals().
chrConf Configuration of chromosomes returned by splitChromosomes().
mutationChance Chance for a nonsense mutation to occur.

Examples

```

## Not run:
library(ALL)
data(ALL)

demoALL<-ALL[1:12,1:8]

set.seed(1234)
population<-InitialPopulation(demoALL, 4, 9)
individuals<-Individuals(population)

chrConf<-splitChromosomes(demoALL, 2)
chrConf
individuals

set.seed(123)
nonSenseMutation(individuals, chrConf, 20)

## End(Not run)

```

PlotGenAlg

PlotGenAlg

Description

Function for graphically representing the evolution.

Usage

```
PlotGenAlg(DGenes, dGenes, maxEval, meanEval)
```

Arguments

DGenes	Occurences of genes as dominant.
dGenes	Occurences of genes as recessive. For future developments.
maxEval	Maximum fitness.
meanEval	Average fitness.

Examples

```
## Not run:
#Graphical representation of the evolution after each generation.
#Intended to be used by dGAselID() only.
#Please refer to the example for dGAselID().

## End(Not run)
```

pointMutation

pointMutation

Description

Operator for the point mutation.

Usage

```
pointMutation(individuals, mutationChance)
```

Arguments

individuals	dataset returned by Individuals().
mutationChance	chance for a point mutation to occur.

Examples

```
## Not run:
library(ALL)
data(ALL)

demoALL<-ALL[1:12,1:8]

set.seed(1234)
population<-InitialPopulation(demoALL, 4, 9)
individuals<-Individuals(population)

individuals
set.seed(123)
pointMutation(individuals, 4)

## End(Not run)
```

RandomAssortment

RandomAssortment

Description

Random assortment of chromosomes operator.

Usage

```
RandomAssortment(newChrs, chrConf)
```

Arguments

newChrs	Set of chromosomes.
chrConf	Configuration of chromosomes returned by splitChromosomes().

Examples

```
## Not run:
library(ALL)
data(ALL)

demoALL<-ALL[1:12,1:8]

population02<-InitialPopulation(demoALL, 2, 4, FALSE)
chrConf02<-splitChromosomes(demoALL, 4)

set.seed(1357)
cr1<-Crossover(population02[1,], population02[2,], chrConf02)
RandomAssortment(cr1, chrConf02)
cr1
chrConf02
```

```
## End(Not run)
```

RandomizePop

RandomizePop

Description

Generates a random population for the next generation.

Usage

```
RandomizePop(population)
```

Arguments

population Population of chromosome sets in current generation.

Examples

```
## Not run:
library(ALL)
data(ALL)

demoALL<-ALL[1:12,1:8]

population01<-InitialPopulation(demoALL, 4, 4)
population01
RandomizePop(population01)

## End(Not run)
```

splitChromosomes

splitChromosomes

Description

Divides the genotypes into sets with a desired number of chromosomes.

Usage

```
splitChromosomes(x, noChr = 22)
```

Arguments

x Dataset in ExpressionSet format.
noChr Desired number of chromosomes. The default value is 22.

Examples

```
## Not run:
library(ALL)
data(ALL)

demoALL<-ALL[1:12,1:8]

splitChromosomes(demoALL, 3)
splitChromosomes(demoALL)

## End(Not run)
```

transposon

transposon

Description

Operator for transposons.

Usage

```
transposon(individuals, chrConf, mutationChance)
```

Arguments

individuals dataset returned by Individuals().
chrConf Configuration of chromosomes returned by splitChromosomes().
mutationChance Chance for a transposon mutation to occur.

Examples

```
## Not run:
library(ALL)
data(ALL)

demoALL<-ALL[1:12,1:8]

set.seed(1234)
population<-InitialPopulation(demoALL, 4, 9)
individuals<-Individuals(population)

chrConf<-splitChromosomes(demoALL, 2)
chrConf
individuals

set.seed(123)
transposon(individuals, chrConf, 20)

## End(Not run)
```

```
wholeChromosomeDeletion  
  wholeChromosomeDeletion
```

Description

Operator for the deletion of a whole chromosome.

Usage

```
wholeChromosomeDeletion(individuals, chrConf, mutationChance)
```

Arguments

individuals	dataset returned by Individuals().
chrConf	Configuration of chromosomes returned by splitChromosomes().
mutationChance	Chance for a deletion of a whole chromosome mutation to occur.

Examples

```
## Not run:  
library(ALL)  
data(ALL)  
  
demoALL<-ALL[1:12,1:8]  
  
set.seed(1234)  
population<-InitialPopulation(demoALL, 4, 9)  
individuals<-Individuals(population)  
  
chrConf<-splitChromosomes(demoALL, 2)  
chrConf  
individuals  
  
set.seed(123)  
wholeChromosomeDeletion(individuals, chrConf, 20)  
  
## End(Not run)
```

Index

AnalyzeResults, [2](#)
Crossover, [3](#)
dGAselID, [4](#)
Elitism, [6](#)
EmbryonicSelection, [7](#)
EvaluationFunction, [8](#)
frameShiftMutation, [9](#)
Individuals, [10](#)
InitialPopulation, [10](#)
largeSegmentDeletion, [11](#)
nonSenseMutation, [12](#)
PlotGenAlg, [13](#)
pointMutation, [13](#)
RandomAssortment, [14](#)
RandomizePop, [15](#)
splitChromosomes, [15](#)
transposon, [16](#)
wholeChromosomeDeletion, [17](#)