Package: GARS (via r-universe)

November 1, 2026

Type Package

Date 2020-09-04

Title GARS: Genetic Algorithm for the identification of Robust Subsets of variables in high-dimensional and challenging datasets

Version 1.32.0

Author Mattia Chiesa <mattia.chiesa@hotmail.it>, Luca Piacentini <luca.piacentini@cardiologicomonzino.it>

Maintainer Mattia Chiesa <mattia.chiesa@hotmail.it>

Description Feature selection aims to identify and remove redundant, irrelevant and noisy variables from high-dimensional datasets. Selecting informative features affects the subsequent classification and regression analyses by improving their overall performances. Several methods have been proposed to perform feature selection: most of them relies on univariate statistics, correlation, entropy measurements or the usage of backward/forward regressions. Herein, we propose an efficient, robust and fast method that adopts stochastic optimization approaches for high-dimensional. GARS is an innovative implementation of a genetic algorithm that selects robust features in high-dimensional and challenging datasets.

License GPL (>= 2)

Encoding UTF-8

LazyData true

VignetteBuilder knitr

RoxygenNote 6.1.1

biocViews Classification, FeatureExtraction, Clustering

Imports DaMiRseq, MLSeq, stats, methods, SummarizedExperiment

Suggests BiocStyle, knitr, testthat

Depends R (>= 3.5), ggplot2, cluster

Config/pak/sysreqs cmake libfreetype6-dev libglpk-dev make default-jdk libbz2-dev libicu-dev libjpeg-dev liblzma-dev libpng-dev libuv1-dev libxml2-dev libssl-dev xz-utils zlib1g-dev

Repository Bioconductor 3.23

Date/Publication 2026-04-28 12:47:23 UTC

RemoteUrl https://github.com/bioc/GARS

RemoteRef RELEASE_3_23

RemoteSha 851e6551191b438d3db6c7b19e47b9490f215e6c

Contents

AllPop . . . 2
FitScore . . . 3
GARS . . . 4
GARS_classes . . . 4
GARS_create_rnd_population . . . 5
GARS_Crossover . . . 6
GARS_data_norm . . . 7
GARS_Elitism . . . 7
GARS_fit_list . . . 8
GARS_FitFun . . . 9
GARS_Fitness_score . . . 10
GARS_GA . . . 10
GARS_Mutation . . . 12
GARS_PlotFeaturesUsage . . . 13
GARS_PlotFitnessEvolution . . . 14
GARS_pop_list . . . 15
GARS_popul . . . 15
GARS_res_GA . . . 16
GARS_Selection . . . 16
GarsSelectedFeatures-class . . . 17
LastPop . . . 18
MatrixFeatures . . . 18

Index **20**

`AllPop` *Accessors for the 'AllPop' slot of a GarsSelectedFeatures object.*

Description

The AllPop slot contains the list of populations

Usage

```
AllPop(x)

## S4 method for signature 'GarsSelectedFeatures'
AllPop(x)
```

Arguments

x	a GarsSelectedFeatures object

Value

a list containing all the populations

Author(s)

Mattia Chiesa, Luca Piacentini

Examples

```
data(GARS_res_GA)
ex_pop <- AllPop(GARS_res_GA)
```

FitScore *Accessors for the 'FitScore' slot of a GarsSelectedFeatures object.*

Description

The FitScore slot contains the fitness values over the generations

Usage

```
FitScore(x)

## S4 method for signature 'GarsSelectedFeatures'
FitScore(x)
```

Arguments

x	a GarsSelectedFeatures object

Value

a vector containing the fitness scores

Author(s)

Mattia Chiesa, Luca Piacentini

Examples

```
data(GARS_res_GA)
ex_pop <- FitScore(GARS_res_GA)
```

GARS *GARS package for a robust feature selection of high-dimensional data*

Description

The main function of GARS is GARS_GA, which implements a clustering-based Genetic Algorithm to select Robust Subsets of features in high-dimensional datasets. The user can extract the results of GARS_GA, exploiting the assessor methods: MatrixFeatures, LastPop, AllPop and FitScore.

Details

See the package vignette, by typing `vignette("GARS")` to discover all the GARS_GA functions.

Author(s)

Mattia Chiesa, Giada Maioli, Luca Piacentini

GARS_classes *RNA-seq dataset for testing GARS*

Description

The class labels of the sample dataset

Usage

```
GARS_classes
```

Format

A vector of type "factor" with 58 elements: 29 labelled as "N" and 29 labelled as "T".

Value

An example data for testing GARS package

GARS_create_rnd_population
Create a random chromosomes population

Description

This function creates the initial random population of chromosomes

Usage

```
GARS_create_rnd_population(data, chr.len, chr.num = 1000)
```

Arguments

data	A SummarizedExperiment object or a matrix or a data.frame. In case of matrix or data.frame: • Rows and Cols have to be, respectively, observations and features. The variables are tipically genes; • GARS also accept other -omic features as well as any continuous or factorial variables (e.g. sex, age, cholesterol level,...); • Usually the number of observation is « than the number of features ,
chr.len	The length of chromosomes. This value corresponds to the desired length of the feature set.
chr.num	The number of chromosomes to generate. Default is 1000

Value

A matrix representing the chromosomes population: each column is a chromosome and each element correspond to the feature position in 'data'

Author(s)

Mattia Chiesa, Luca Piacentini

Examples

```
# use example data:
data(GARS_data_norm)
GARS_create_rnd_population(GARS_data_norm, chr.len=10, chr.num=100)
```

GARS_Crossover *Perform the one-point and the two-point Crossover*

Description

This function implements the one-point and the two-point cross-over.

Usage

```
GARS_Crossover(chr.pop, co.rate = 0.8, type = c("one.p", "two.p"),
  one.p.quart = c("I.quart", "II.quart", "III.quart"))
```

Arguments

chr.pop	A matrix or a data.frame representing the chromosomes population: each column is a chromosome and each element corresponds to the feature position in the data matrix
co.rate	The probability of each random couple of chromosomes to swap some parts. It must be between 0 and 1. Default is 0.8
type	The type of crossover method; one-point ("one.p") and two-point ("two.p") are allowed. Default is "one.p"
one.p.quart	The position of the cromosome where performing the crossover, if "one.p" is selected. The first quartile ("I.quart"), the second quartile ("II.quart", i.e. the median) and the third quartile ("III.quart") are allowed. Default is "I.quart"

Value

A matrix representing the "crossed" population. The dimensions of this matrix are the same of 'chr.pop'

Author(s)

Mattia Chiesa, Luca Piacentini

See Also

GARS_Mutation, GARS_Selection, GARS_Elitism,

Examples

```
data(GARS_popul)
crossed_pop <- GARS_Crossover(GARS_popul, co.rate=0.9)
crossed_pop <- GARS_Crossover(GARS_popul, type="two.p")
crossed_pop <- GARS_Crossover(GARS_popul, type="one.p",
one.p.quart= "II.quart")
```

GARS_data_norm *RNA-seq dataset for testing GARS*

Description

An RNA-seq normalized matrix to test several GARS functions; this dataset was obtained using the DaMirseq package to normalize the raw count matrix present in MLSeq package.

Usage

```
GARS_data_norm
```

Format

A matrix of 157 genes (columns) and 58 samples (rows)

Value

An example data for testing GARS package

GARS_Elitism *Separate chromosome on the basis of the Fitness Scores*

Description

This function splits the chromosome population in two parts allowing the best chromosomes to be preserved from the "evolutionary" steps: Selection, Crossover and Mutation.

Usage

```
GARS_Elitism(chr.pop, fitn.values, n.elit = 10)
```

Arguments

chr.pop	A matrix or a data.frame representing the chromosomes population: each column is a chromosome and each element corresponds to the feature position in the data matrix
fitn.values	A numeric vector where each element corresponds to the fitness score of each chromosome in 'chr.pop'
n.elit	The number of best chromosomes to be selected by elitism. This number must be even. Default is 10

Value

A list containing:

- The population of best chromosomes selected by elitism.
- The population of chromosomes not selected by elitism.
- The fitness values of best chromosomes selected by elitism.
- The fitness values of chromosomes not selected by elitism.

Author(s)

Mattia Chiesa, Luca Piacentini

See Also

GARS_Mutation, GARS_Selection, GARS_Crossover, GARS_FitFun,

Examples

```
data(GARS_popul)
data(GARS_Fitness_score)
pop_list <- GARS_Elitism(GARS_popul, GARS_Fitness_score)
```

GARS_fit_list *RNA-seq dataset for testing GARS*

Description

A numeric vector with the maximum fitness score for each iteration

Usage

```
GARS_fit_list
```

Format

A numeric vector with 100 fitness scores

Value

An example data for testing GARS package

GARS_FitFun *This function implements the Fitness Function of GARS*

Description

In GARS the Fitness Function consists in calculating the Averaged Silhouette Index after a Multi-Dimensional Scaling

Usage

```
GARS_FitFun(data, classes, chr.pop)
```

Arguments

data	A SummarizedExperiment object or a matrix or a data.frame. In case of matrix or data.frame: • Rows and Cols have to be, respectively, observations and features. The variables are tipically genes; • GARS also accept other -omic features as well as any continuous or factorial variables (e.g. sex, age, cholesterol level,...); • Usually the number of observation is « than the number of features ,
classes	A vector of type "factor" with nrow(data) elements. Each element represents the class label for each observation.
chr.pop	A matrix or a data.frame representing the chromosomes population: each column is a chromosome and each element corresponds to the feature position in the expression data matrix

Value

A numeric vector where each element corresponds to the fitness score of each chromosome in 'chr.pop'

Author(s)

Mattia Chiesa, Luca Piacentini

See Also

GARS_create_rnd_population

Examples

```
# use example data:
data(GARS_data_norm)
data(GARS_classes)
data(GARS_popul)
fitness_scores <- GARS_FitFun(GARS_data_norm, GARS_classes, GARS_popul)
```

GARS_Fitness_score *RNA-seq dataset for testing GARS*

Description

A numeric vector with the fitness scores for each chromosome in a single generation

Usage

```
GARS_Fitness_score
```

Format

A numeric vector with 50 fitness scores

Value

An example data for testing GARS package

GARS_GA *The wrapper fuction to use GARS*

Description

This function allows the users to run all GARS funtion at once. This is the easier and recommended way to use GARS.

Usage

```
GARS_GA(data, classes, chr.num = 1000, chr.len, generation = 500,
  co.rate = 0.8, mut.rate = 0.01, n.elit = 10, type.sel = c("RW",
  "TS"), type.co = c("one.p", "two.p"), type.one.p.co = c("I.quart",
  "II.quart", "III.quart"), n.gen.conv = 80, plots = c("yes", "no"),
  n.Feat_plot = 10, verbose = c("yes", "no"))
```

Arguments

data	A SummarizedExperiment object or a matrix or a data.frame. In case of matrix or data.frame: • Rows and Cols have to be, respectively, observations and features. The variables are tipically genes; • GARS also accept other -omic features as well as any continuous or factorial variables (e.g. sex, age, cholesterol level,...); • Usually the number of observation is « than the number of features ,
classes	The class vector
chr.num	The number of chromosomes to generate. Default is 1000
chr.len	The length of chromosomes. This value corresponds to the desired length of the feature set
generation	The maximum number of generations. Default is 1000
co.rate	The probability of each random couple of chromosomes to swap some parts. It must be between 0 and 1. Default is 0.8
mut.rate	The probability to apply a random mutation to each element. It must be between 0 and 1. Default is 0.01
n.elit	The number of best chromosomes to be selected by elitism. This number must be even. Default is 10
type.sel	The type of selection method; Roulette Wheel ("RW") and Tournament Selection ("TS") are allowed. Default is "RW"
type.co	The type of crossover method; one-point ("one.p") and two-point ("two.p") are allowed. Default is "one.p"
type.one.p.co	The position of the cromosome where performing the crossover, if "one.p" is selected. The first quartile ("I.quart"), the second quartile ("II.quart", i.e. the median) and the third quartile ("III.quart") are allowed. Default is "I.quart"
n.gen.conv	The number of consecutive generations with the same maximum fitness score.
plots	If graphs have to be plotted; "yes" or "no" are allowed. Default is "yes"
n.Feat_plot	The number of features to be plotted
verbose	If statistics have to be printed; "yes" or "no" are allowed. Default is "yes"

Value

A GarsSelectedFeatures object, containg:

data_red a matrix of selected features

last_pop a matrix containg the last chromosome population

pop_list a list containing all the populations produced over the generations

fit_list a numeric vector containing the maximum fitness scores, computed in each generation

Author(s)

Mattia Chiesa, Luca Piacentini

Examples

```
# use example data:
data(GARS_data_norm)
data(GARS_classes)

res_ex <- GARS_GA(GARS_data_norm,
   GARS_classes,
   chr.num = 100,
   chr.len=10,
   generation = 5,
   co.rate = 0.8,
   mut.rate = 0.1,
   n.elit = 10,
   type.sel = "RW",
   type.co ="one.p",
   type.one.p.co = "II.quart",
   n.gen.conv = 80,
   plots = "no",
   verbose = "no")
```

GARS_Mutation *Perform the Mutation step*

Description

This function implements the mutation step in the GA. First, it checks and replace duplicate features in each chromosomes; then, random mutation are applied to the entire population.

Usage

```
GARS_Mutation(chr.pop, mut.rate = 0.01, totFeats)
```

Arguments

chr.pop	A matrix or a data.frame representing the chromosomes population: each column is a chromosome and each element correspond to the feature position in the data matrix
mut.rate	The probability to apply a random mutation to each element. It must be between 0 and 1. Default is 0.01
totFeats	The total number of features. Often, it corresponds to number of columns of the data matrix

Value

A matrix representing the "mutated" population. The dimensions of this matrix are the same of 'chr.pop'

Author(s)

Mattia Chiesa, Luca Piacentini

See Also

GARS_Elitism, GARS_Selection, GARS_Crossover,

Examples

```
# use example data:
data(GARS_popul)
data(GARS_data_norm)

mutated_pop <- GARS_Mutation(GARS_popul, mut.rate=0.1,
 dim(GARS_data_norm)[2])
```

GARS_PlotFeaturesUsage

A bubble chart to assess the usage of each features

Description

This function allows assessing visually how many times a feature is selected across the generations. In principle, a highly recurring feature is more likely to be important.

Usage

```
GARS_PlotFeaturesUsage(popul.list, allFeat, nFeat = length(allFeat))
```

Arguments

popul.list	A SummarizedExpression object
allFeat	A character vector containing the list of the all features name. Often, it corresponds to the columns name of the data matrix.
nFeat	The number of features which have to be plotted. Default is 'length(allFeat)'

Value

A bubble chart where each plotted feature is represented by a colored circle. A feature is important (i.e. conserved) if the size is wide and the color tends to red; the smaller the size, the lighter the color and less informative the feature.

Author(s)

Mattia Chiesa, Luca Piacentini

See Also

GARS_PlotFitnessEvolution

Examples

```
# use example data:
data(GARS_data_norm)
data(GARS_pop_list)
allfeat_names <- colnames(GARS_data_norm)
GARS_PlotFeaturesUsage(GARS_pop_list, allfeat_names, nFeat = 10)
```

GARS_PlotFitnessEvolution

Plot the maximum fitness scores for each generation

Description

This function plots the maximum fitness scores for each generation

Usage

```
GARS_PlotFitnessEvolution(fitness.scores)
```

Arguments

`fitness.scores`	A numeric vector where each element corresponds to the fitness score

Value

A plot which represent the evolution of the fitness score across the generations

Author(s)

Mattia Chiesa, Luca Piacentini

See Also

GARS_PlotFeaturesUsage

Examples

```
# use example data:
data(GARS_fit_list)
GARS_PlotFitnessEvolution(GARS_fit_list)
```

GARS_pop_list *RNA-seq dataset for testing GARS*

Description

A list containing 100 of consecutive chromosomes populations

Usage

```
GARS_pop_list
```

Format

A list with 100 consecutive chromosomes populations

Value

An example data for testing GARS package

GARS_popul *RNA-seq dataset for testing GARS*

Description

A matrix to test several GARS functions, representing a chromosome population

Usage

```
GARS_popul
```

Format

A matrix of 20 rows (features) and 50 columns (chromosomes)

Value

An example data for testing GARS package

GARS_res_GA *A GarsSelectedFeatures object for testing GARS*

Description

An object representing the output of GARS_GA

Usage

```
GARS_res_GA
```

Format

A GarsSelectedFeatures

Value

An example data for testing GARS package

GARS_Selection *Perform the "Roulette Wheel" or the "Tournament" selection*

Description

This function implements two kind of GA Selection step: the "Roulette Wheel" and the "Tournament" selection.

Usage

```
GARS_Selection(chr.pop, type = c("RW", "TS"), fitn.values)
```

Arguments

chr.pop	A matrix or a data.frame representing the chromosomes population: each column is a chromosome and each element corresponds to the feature position in the data matrix
type	The type of selection method; Roulette Wheel ("RW") and Tournament Selection ("TS") are allowed. Default is "RW"
fitn.values	A numeric vector where each element corresponds to the fitness score of each chromosome in 'chr.pop'

Value

A matrix representing the "selected" population. The dimensions of this matrix are the same of 'chr.pop'.

Author(s)

Mattia Chiesa, Luca Piacentini

See Also

GARS_Mutation, GARS_Crossover, GARS_Elitism,

Examples

```
# use example data:
data(GARS_popul)
data(GARS_Fitness_score)
selected_pop <- GARS_Selection(GARS_popul, "RW", GARS_Fitness_score)
```

GarsSelectedFeatures-class

The output class 'GarsSelectedFeatures'

Description

The output class for GARS_GA function

Slots

`data_red` a matrix containing the expression values for the selected feature

`last_pop` a matrix containing the chromosome population of the last generation

`pop_list` a list containing all the populations produced over the generations

`fit_list` a vector containing the maximum fitness scores

Examples

```
showClass("GarsSelectedFeatures")
```

LastPop *Accessors for the 'LastPop' slot of a GarsSelectedFeatures object.*

Description

The LastPop slot contains the last chromosome population

Usage

```
LastPop(x)

## S4 method for signature 'GarsSelectedFeatures'
LastPop(x)
```

Arguments

x	a GarsSelectedFeatures object

Value

a matrix containing the last population

Author(s)

Mattia Chiesa, Luca Piacentini

Examples

```
data(GARS_res_GA)
ex_pop <- LastPop(GARS_res_GA)
```

MatrixFeatures *Accessors for the 'MatrixFeatures' slot of a GarsSelectedFeatures object.*

Description

The MatrixFeatures slot contains the reduced dataset

Usage

```
MatrixFeatures(x)

## S4 method for signature 'GarsSelectedFeatures'
MatrixFeatures(x)
```

Arguments

x	a GarsSelectedFeatures object

Value

a matrix with the reduced dataset

Author(s)

Mattia Chiesa, Luca Piacentini

Examples

```
data(GARS_res_GA)
ex_matrix <- MatrixFeatures(GARS_res_GA)
```

Index

∗ **datasets**
GARS_classes, 4
GARS_data_norm, 7
GARS_fit_list, 8
GARS_Fitness_score, 10
GARS_pop_list, 15
GARS_popul, 15
GARS_res_GA, 16
∗ **package**
GARS, 4

AllPop, 2, *4*
AllPop,GARS-AllPop *(*AllPop*)*, 2
AllPop,GarsSelectedFeatures-method *(*AllPop*)*, 2

FitScore, 3, *4*
FitScore,GARS-FitScore *(*FitScore*)*, 3
FitScore,GarsSelectedFeatures-method *(*FitScore*)*, 3

GARS, 4
GARS-package *(*GARS*)*, 4
GARS_classes, 4
GARS_create_rnd_population, 5, *9*
GARS_Crossover, 6, *8*, *13*, *17*
GARS_data_norm, 7
GARS_Elitism, *6*, 7, *13*, *17*
GARS_fit_list, 8
GARS_FitFun, *8*, 9
GARS_Fitness_score, 10
GARS_GA, *4*, 10
GARS_Mutation, *6*, *8*, 12, *17*
GARS_PlotFeaturesUsage, 13, *14*
GARS_PlotFitnessEvolution, 14, *14*
GARS_pop_list, 15
GARS_popul, 15
GARS_res_GA, 16
GARS_Selection, *6*, *8*, *13*, 16
GarsSelectedFeatures-class, 17

LastPop, *4*, 18
LastPop,GARS-LastPop *(*LastPop*)*, 18
LastPop,GarsSelectedFeatures-method *(*LastPop*)*, 18

MatrixFeatures, *4*, 18
MatrixFeatures,GARS-MatrixFeatures *(*MatrixFeatures*)*, 18
MatrixFeatures,GarsSelectedFeatures-method *(*MatrixFeatures*)*, 18