Package ‘RFreak’

February 19, 2015

Version 0.3-0
Date 2014-02-19
Title R/FrEAK interface
Author Robin Nunkesser <Robin.Nunkesser@hshl.de>
Maintainer Robin Nunkesser <Robin.Nunkesser@hshl.de>
Imports methods, rJava (>= 0.5.0)
Description An R interface to a modified version of the Free Evolutionary Algorithm Kit FrEAK. FrEAK is a toolkit written in Java to design and analyze evolutionary algorithms. Both the R interface and an extended version of FrEAK are contained in the RFreak package. For more information on FrEAK see http://sourceforge.net/projects/freak427/.

LazyLoad yes
License GPL (>= 2)
SystemRequirements Java (>= 5.0)
NeedsCompilation no
Repository CRAN
Date/Publication 2014-02-20 22:40:52

R topics documented:

data.logicfs .................................................. 2
evolreg-class ............................................. 2
executeSchedule .......................................... 3
FreakReturn-class ......................................... 4
GPAS-class .................................................. 4
GPASDiscrimination ........................................ 5
GPASInteractions .......................................... 6
launchScheduleEditor ................................. 8
LTSevol ......................................................... 9
predict-method ............................................ 10
robreg.evol ............................................... 11

Index 13
data.logicsfs  

Example Data of GPAS

Description

data.logicsfs contains two objects: a simulated matrix data.logicsfs of 400 observations (rows) and 15 variables (columns) and a vector cl.logicsfs of length 400 containing the class labels of the observations.

Each variable is categorical with realizations 1, 2 and 3. The first 200 observations are cases, the remaining are controls. If one of the following expressions is TRUE, then the corresponding observation is a case:

SNP1 == 3
SNP2 == 1 AND SNP4 == 3
SNP3 == 3 AND SNP5 == 3 AND SNP6 == 1

where SNP1 is in the first column of data.logicsfs, SNP2 in the second, and so on.

evolreg-class  

Class "evolreg"

Description

Encapsulates information returned from FrEAK computing robust regression. For compatibility reasons also called ltsEA.

Objects from the Class

An evolreg object holds four slots.

Slots

summary: A data.frame with a summary of the FrEAK run (inherited from "FreakReturn")
best: The best subset found
coefficients: Vector of coefficient estimates
crit: The value of the objective function of the used regression method

Extends

Class "FreakReturn", directly.

Author(s)

Robin Nunkesser <Robin.Nunkesser@hshl.de>

See Also

"FreakReturn", robreg.evol
executeSchedule

executeSchedule  

**Description**

Executes a schedule created by the FrEAK schedule editor and returns a summary of the result.

**Usage**

```r
eexecuteSchedule(freakfile = "schedule.freak")
```

**Arguments**

- **freakfile**   File containing the schedule to be executed

**Value**

Returns an object of class `FreakReturn` enwrapping a data.frame in its only slot `summary` containing information about the last population of the executed schedule. For each individual in the last population the following information is contained:

- **run**   The run the individual was found in
- **generation**   The generation the individual was created in
- **objective value(s)**   The objective value(s) as returned by the fitness function
- **individual**   A string representation of the individual

**Warning**

To obtain a result, the schedule needs to have a stopping criterion and the observer "Result" and the view "R Return" which are automatically preselected when using `launchScheduleEditor`.

**Author(s)**

Robin Nunkesser <Robin.Nunkesser@hshl.de>

**See Also**

- `launchScheduleEditor`
- "FreakReturn"

**Examples**

```r
## Not run:
# Start the schedule editor and set up a schedule
launchScheduleEditor()

# Execute the set up schedule.
executeSchedule()

## End(Not run)
```
Class "FreakReturn"

Description
Encapsulates information returned from FrEAK

Objects from the Class
A FreakReturn object holds only one slot, containing a data.frame with the summary of an executed FrEAK run.

Slots
summary: A data.frame with informations for each returned individual on the run the individual was found in, the generation the individual was created in, the objective value(s) as returned by the fitness function, and the individual itself.

Author(s)
Robin Nunkesser <robin.nunkesser@hshl.de>

See Also
"ltsea", "gpas"

Class "GPAS"

Description
Encapsulates information returned from FrEAK computing the GPAS algorithm.

Objects from the Class
A GPAS object holds two slots.

Slots
summary: A data.frame with a summary of the FrEAK run (inherited from "FreakReturn")
trees: The Java objects representing the individuals returned by the FrEAK run

Methods
predict Method to obtain predictions of an individual based on new predictors
**GPASDiscrimination**

**Extends**
Class "FreakReturn", directly.

**Author(s)**
Robin Nunkesser <robin.nunkesser@hshl.de>

**See Also**
"FreakReturn", GPASDiscrimination, GPASInteractions

---

**GPASDiscrimination**  
*Execute the GPAS algorithm for discrimination*

---

**Description**
Working on categorical data with binary response, the algorithm searches for multi-valued logic expressions in disjunctive normal form discriminating between response 0 and response 1. The algorithm is intended for genetic association studies on SNP data.

**Usage**
```r
GPASDiscrimination(respNtrain, predsNtrain, respNtest=NULL, 
predsNtest=NULL, runs = 1, generations = 10000)
```

**Arguments**
- `respNtrain`: Vector with the response variables of the training data set
- `predsNtrain`: Matrix or data frame with the predictors of the training data set
- `respNtest`: Optional vector with the response variables of the test data set
- `predsNtest`: Optional matrix or data frame with the predictors of the test data set
- `runs`: Number of independent runs of GPAS
- `generations`: Number of generations after which the algorithm will be stopped

**Value**
Returns an object of class GPAS with a data frame in its slot summary containing information about the last population of the executed discrimination runs. For each individual in the last population the following information is contained:
- `data set`: Either 'training' or 'test' or omitted
- `run`: The run the individual was found in
- `generation`: The generation the individual was created in
- `objective value 1`: Correctly predicted cases
Objective value 2
Correctly predicted controls

Objective value 3
Length of the individual

Individual
A string representation of the individual

Author(s)
Robin Nunkesser <Robin.Nunkesser@hshl.de>

References

See Also
"GPAS", *GPASInteractions*

Examples

```r
# load example data
data(data.logicfs)

# execute GPAS to discriminate between cases and controls
GPASDiscrimination(cl.logicfs, data.logicfs, runs=1, generations=1000)
```

**GPASInteractions**
*Execute the GPAS algorithm for feature selection*

Description
Identification of interesting (high order) SNP interactions. The algorithm works on categorical data with binary response and delivers multi-valued logic expressions in disjunctive normal form typically explaining subsets of the data and an interaction tree containing interesting interactions.

Usage

```r
GPASInteractions(resp, preds, runs = 1, generations = 10000,
savegraph = "interactions.dot", occurrences=10, ratio=0.1)
```
Arguments

resp    Vector with the response variables
preds   Matrix or data frame with the predictors
runs    Number of independent runs of GPAS
generations Number of generations after which the algorithm will be stopped
savegraph Name of the file the resulting GraphViz graph will be saved to
occurences The minimum number of times an interaction has to occur to be included in the graph
ratio   The minimal ratio a single literal has to occur in relation to his ancestor in the interaction graph

Value

Returns an object of class GPAS with a data frame in its slot summary containing information about the last population of the executed runs. For each individual in the last population the following information is contained:

run      The run the individual was found in
generation The generation the individual was created in
objective value 1 Sum of correctly predicted cases and controls
objective value 2 Correctly predicted controls
objective value 3 Length of the individual
individual A string representation of the individual

Author(s)

Robin Nunkesser <robin.nunkesser@hshl.de>

References


GraphViz: http://www.graphviz.org/

See Also

“GPAS”, GPASDiscrimination
launchScheduleEditor

Examples

# load example data
data(data.logicfs)

# execute GPAS to search for interesting interactions
GPASInteractions(cl.logicfs, data.logicfs, runs=1, generations=1000)

launchScheduleEditor  Launches a graphical schedule editor

Description

Launches a graphical schedule editor for FrEAK schedules. A schedule contains the algorithm and simulation options for the desired evolutionary algorithm.

Usage

launchScheduleEditor(saveTo = "schedule.freak", load = NULL)

Arguments

saveTo  Name of the file the schedule is saved to
load    Name of an (optional) file that should be loaded to the editor

Details

Setting up a schedule consists of seven algorithm specific steps and two simulation specific steps. To set up an evolutionary algorithm it is necessary to choose a search space, a fitness function, an optional genotype-mapper, an algorithm graph, a set of stopping criteria, a population model, and an initial population. The simulation specific steps allow the user to choose views and observers (the necessary ones for the R interface are preselected, views depending on the FrEAK GUI are not supported). The last step consists of choosing batches (only one batch supported) and the number of independent runs. A detailed instruction on how to set up schedules can be found in the User’s Guide of FrEAK.

Warning

Editing schedules has to be finished by pressing the "Finish" button to obtain a file including the schedule. The observer "Result" and the view "R Return" - which are preselected - and a stopping criterion are necessary for the R interface to work.

Author(s)

Robin Nunkesser <Robin.Nunkesser@hshl.de>
References
FrEAK User's Guide:

See Also
executeSchedule

Examples
```r
## not run:
# Start the schedule editor and set up a schedule
launchScheduleEditor()

## end(not run)
```

##  Least Trimmed Squares Robust Regression

### Description
Carries out least trimmed squares (LTS) robust regression with an evolutionary algorithm. The LTS regression method minimizes the sum of the $h$ smallest squared residuals. Deprecated. Use `robreg.evol` instead.

### Usage
```r
## deprecated:
LTSevol(y, x, h = NULL, adjust = FALSE, runs = 1, generations = 10000)
```

### Arguments
- **y**: Vector with the response variables
- **x**: Matrix or data frame containing the explanatory variables
- **h**: Parameter determining the trimming
- **adjust**: Whether to perform intercept adjustment at each step
- **runs**: Number of independent runs
- **generations**: Number of generations after which the algorithm will be stopped

### Value
The function `LTSevol` returns an object of class "LtsEA". This object contains:
- **summary**: Summary of the FrEAK run
- **best**: The best subset found
- **coefficients**: Vector of coefficient estimates
- **crit**: The value of the objective function of the LTS regression method, i.e., the sum of the $h$ smallest squared raw residuals
predict-method

Author(s)

Robin Nunkesser <Robin.Nunkesser@hshl.de>

References


See Also

"ltsEA"

Examples

# load example data
data(stackloss)

# compute LTS regression
LTSvol(stackloss[, 4], stackloss[, 1:3], adjust=TRUE, runs=1, generations=1000)

predict-method

Predict using new predictors

Description

Takes an individual from a GPAS object and predicts on the basis of new predictors.

Usage

## S4 method for signature 'GPAS'
predict(object, individual, preds)

Arguments

object Object of class GPAS
individual Number of the individual to use
preds New predictors

Value

Returns a vector with the new predictions.
Description

Carries out robust regression with an evolutionary algorithm. \texttt{ltsreg}, \texttt{ltareg}, \texttt{lmsreg}, \texttt{lqsreg}, and \texttt{lqdreg} are wrappers.

Usage

\begin{verbatim}
robreg.evol(x, y, method = c("lts", "lta", "lms", "lqs", "lqd"),
quantile=NULL, adjust=FALSE, runs=1, generations=10000, duration=0)

## wrappers:
ltsreg.evol(...)
ltares.evol(...)
lmsreg.evol(...)
lqsreg.evol(...)
lqdreg.evol(...)
\end{verbatim}

Arguments

\begin{itemize}
\item \texttt{x} \hspace{1cm} Matrix or data frame containing the explanatory variables
\item \texttt{y} \hspace{1cm} Vector with the response variables
\item \texttt{method} \hspace{1cm} The method to be used. One of "lts", "lta", "lms", "lqs", and "lqd".
\item \texttt{quantile} \hspace{1cm} The quantile to be used: see Details.
\item \texttt{adjust} \hspace{1cm} Whether to perform intercept adjustment at each step
\item \texttt{runs} \hspace{1cm} Number of independent runs
\item \texttt{generations} \hspace{1cm} Number of generations after which the algorithm will be stopped
\item \texttt{duration} \hspace{1cm} Duration in seconds after which the algorithm will be stopped
\end{itemize}

Details

Suppose there are \(n\) data points and \(p\) regressors, including any intercept.

The first four methods minimize some function of the sorted squared residuals. For methods "lqs" and "lms" it is the quantile squared residual, and for "lts" ("lts") it is the sum of the quantile smallest squared (absolute) residuals. "lqd" minimizes approximately the quartile of the absolute residual differences.
Value
The function `robreg.evol` returns an object of class "evolreg". This object contains:

- `summary`: Summary of the FrEAK run
- `best`: The best subset found
- `coefficients`: Vector of coefficient estimates
- `crit`: The value of the objective function of the regression method

Author(s)
Robin Nunkesser <Robin.Nunkesser@hshl.de>

References

See Also
"evolreg"

Examples
```r
# load example data
data(stackloss)

# compute different regressions
robreg.evol(stackloss[, 1:3], stackloss[, 4], method = "lts", generations = 1000)
lqsreg.evol(stackloss[, 1:3], stackloss[, 4], generations = 1000)
lqreg.evol(stackloss[, 1:3], stackloss[, 4], generations = 1000)
```
Index

*Topic classes
  evolreg-class, 2
  FreakReturn-class, 4
  GPAS-class, 4
*Topic datasets
  data.logicsfs, 2
*Topic interface
  executeSchedule, 3
  GPASDiscrimination, 5
  GPASInteractions, 6
  launchScheduleEditor, 8
  LTSevol, 9
  robreg.evol, 11
*Topic methods
  predict-method, 10
*Topic optimize
  executeSchedule, 3
  launchScheduleEditor, 8
*Topic robust
  LTSevol, 9
  robreg.evol, 11
*Topic tree
  GPASDiscrimination, 5
  GPASInteractions, 6

cl.logicsfs (data.logicsfs), 2

data.logicsfs, 2

evolreg, 12
  evolreg-class, 2
  executeSchedule, 3, 9

FreakReturn, 2–5
  FreakReturn-class, 4

GPAS, 4, 6, 7, 11
  GPAS-class, 4
  GPASDiscrimination, 5, 5, 7, 11
  GPASInteractions, 5, 6, 6, 11