Package: copulaedas (via r-universe)

January 10, 2025

Type Package

Title Estimation of Distribution Algorithms Based on Copulas

Version 1.4.3

Description Provides a platform where EDAs (estimation of distribution algorithms) based on copulas can be implemented and studied. The package offers complete implementations of various EDAs based on copulas and vines, a group of well-known optimization problems, and utility functions to study the performance of the algorithms. Newly developed EDAs can be easily integrated into the package by extending an S4 class with generic functions for their main components.

License GPL (>= 2)

URL https://github.com/yasserglez/copulaedas

Depends methods

Imports copula, vines, mvtnorm, truncnorm

Suggests cec2005benchmark, cec2013

Collate EDA.R edaSeed.R edaSelect.R edaOptimize.R edaReport.R edaReplace.R edaTerminate.R edaRun.R margins.R problems.R edaIndepRuns.R edaCriticalPopSize.R CEDA.R VEDA.R

NeedsCompilation no

Author Yasser Gonzalez-Fernandez [aut, cre], Marta Soto [aut]

Maintainer Yasser Gonzalez-Fernandez <ygonzalezfernandez@gmail.com>

Date/Publication 2018-07-29 05:50:03 UTC

Additional_repositories https://cranhaven.r-universe.dev

Config/pak/sysreqs make libgsl0-dev

Repository https://cranhaven.r-universe.dev

RemoteUrl https://github.com/cranhaven/cranhaven.r-universe.dev

RemoteRef package/copulaedas

RemoteSha a4fba9371da534d1cd78a17c4e0a8674cb2bd029

RemoteSubdir copulaedas

Contents

`CEDA-class` *Class for Copula EDAs*

Description

Extends the `EDA` class to implement EDAs based on multivariate copulas. Objects are created by calling the `CEDA` function.

Details

Copula EDAs (CEDA) are a class of EDAs that model the search distributions using a multivariate copula. These algorithms estimate separately the univariate marginal distributions and the dependence structure from the selected population. The dependence structure is represented through a multivariate copula. The following instances of CEDA are implemented.

- If the dependence structure is modeled using a product copula, the resulting algorithm corresponds to the Univariate Marginal Distribution Algorithm (UMDA) for the continuous domain (Larrañaga et al. 1999, 2000).
- If the dependence structure is modeled using a normal copula, the resulting algorithm corresponds to the Gaussian Copula Estimation of Distribution Algorithm (GCEDA) (Soto et al. 2007; Arderí 2007). If non-normal marginal distributions are used, the correlation matrix is calculated using the inversion of Kendall's tau for each pair of variables (Demarta and McNeil 2005). The correction proposed in (Rousseeuw and Molenberghs 1993) is applied if the resulting correlation matrix is not positive-definite. If normal marginal distributions are used, the correlation matrix is estimated directly from the selected population using the `cor` function.

The following parameters are recognized by the functions that implement the edaLearn and edaSample methods for the CEDA class.

copula Multivariate copula. Supported values are: "indep" (independence or product copula) and "normal" (normal copula). Default value: "normal".

margin Marginal distributions. If this argument is "xxx", the algorithm will search for three functions named fxxx. pxxx and qxxx to fit each marginal distribution and evaluate the cumulative distribution function and its inverse, respectively. Default value: "norm".

popSize Population size. Default value: 100.

Slots

name: See the documentation of the slot in the EDA class.

parameters: See the documentation of the slot in the EDA class.

Methods

edaLearn signature(eda = "CEDA"): The edaLearnCEDA function.

edaSample signature(eda = "CEDA"): The edaSampleCEDA function.

References

Arderí RJ (2007). Algoritmo con estimación de distribuciones con cópula gaussiana. Bachelor's thesis, University of Havana, Cuba.

Demarta S, McNeil AJ (2005). The t Copula and Related Copulas. *International Statistical Review*, **73**(1), 111–129.

Gonzalez-Fernandez Y, Soto M (2014). **copulaedas**: An R Package for Estimation of Distribution Algorithms Based on Copulas. *Journal of Statistical Software*, **58**(9), 1-34. http://www.jstatsoft.org/v58/i09/.

Larrañaga P, Etxeberria R, Lozano JA, Peña JM (1999). Optimization by Learning and Simulation of Bayesian and Gaussian Networks. Technical Report EHU-KZAA-IK-4/99, University of the Basque Country.

Larrañaga P, Etxeberria R, Lozano JA, Peña JM (2000). Optimization in Continuous Domains by Learning and Simulation of Gaussian Networks. In *Proceedings of the Workshop in Optimization by Building and Using Probabilistic Models in the Genetic and Evolutionary Computation Conference (GECCO 2000)*, pp. 201–204.

Rousseeuw P, Molenberghs G (1993). Transformation of Nonpositive Semidefinite Correlation Matrices. *Communications in Statistics: Theory and Methods*, **22**, 965–984.

Soto M, Ochoa A, Arderí RJ (2007). Gaussian Copula Estimation of Distribution Algorithm. Technical Report ICIMAF 2007-406, Institute of Cybernetics, Mathematics and Physics, Cuba. ISSN 0138-8916.

Examples

```
setMethod("edaTerminate", "EDA", edaTerminateEval)
setMethod("edaReport", "EDA", edaReportSimple)

UMDA <- CEDA(copula = "indep", margin = "norm",
    popSize = 200, fEval = 0, fEvalTol = 1e-03)
UMDA@name <- "Univariate Marginal Distribution Algorithm"

GCEDA <- CEDA(copula = "normal", margin = "norm",
    popSize = 200, fEval = 0, fEvalTol = 1e-03)
GCEDA@name <- "Gaussian Copula Estimation of Distribution Algorithm"

resultsUMDA <- edaRun(UMDA, fSphere, rep(-600, 5), rep(600, 5))
resultsGCEDA <- edaRun(GCEDA, fSphere, rep(-600, 5), rep(600, 5))

show(resultsUMDA)
show(resultsGCEDA)
```

EDA-class *Base Class for EDAs*

Description

Base class of all the classes that implement EDAs in the package. This is a virtual class, no object may be created from it.

Slots

`name`**:** Object of class `character`. Name of the EDA.

`parameters`**:** Object of class `list`. Parameters of the EDA.

Methods

edaSeed `signature(eda = "EDA")`: Seeding method. Default: `edaSeedUniform`.

edaSelect `signature(eda = "EDA")`: Selection method. Default: `edaSelectTruncation`.

edaOptimize `signature(eda = "EDA")`: Local optimization method. Default: `edaOptimizeDisabled`.

edaReplace `signature(eda = "EDA")`: Replacement method. Default: `edaReplaceComplete`.

edaReport `signature(eda = "EDA")`: Reporting method. Default: `edaReportDisabled`.

edaTerminate `signature(eda = "EDA")`: Termination method. Default: `edaTerminateMaxGen`.

show `signature(object = "EDA")`: Print a textual representation of the EDA.

References

Gonzalez-Fernandez Y, Soto M (2014). **copulaedas**: An R Package for Estimation of Distribution Algorithms Based on Copulas. *Journal of Statistical Software*, **58**(9), 1-34. http://www.jstatsoft.org/v58/i09/.

See Also

CEDA, VEDA, edaRun.

---

edaCriticalPopSize *Critical Population Size*

Description

Determine the critical population size using a bisection method.

Usage

```
edaCriticalPopSize(eda, f, lower, upper, fEval, fEvalTol,
    totalRuns = 30, successRuns = totalRuns, lowerPop = 2,
    upperPop = NA, stopPercent = 10, verbose = FALSE)
```

Arguments

eda	EDA instance.
f	Objective function.
lower	Lower bounds of the variables of the objective function.
upper	Upper bounds of the variables of the objective function.
fEval	Optimum value of the objective function.
fEvalTol	A run is considered successful if the difference between fEval and the best found solution is less than fEvalTol.
totalRuns	Total number of runs.
successRuns	Required number of successfully runs.
lowerPop	Lower bound of the initial interval for the population.
upperPop	Upper bound of the initial interval for the population.
stopPercent	Stop percent.
verbose	Print progress information.

Details

This function determines the minimum population size required by the EDA to reach the value fEval of the objective function in successRuns runs out of a total of totalRuns independent runs (critical population size).

The population size is determined using a bisection method starting with the interval delimited by lowerPop and upperPop. The bisection procedure stops when the estimated population size is less than stopPercent percent away from the critical population size. If either lowerPop or upperPop is not specified, the algorithm will determine an initial interval based on the value of the popSize parameter and then continue using the bisection method.

See (Pelikan 2005) for a pseudocode of a similar algorithm.

Value

Either NULL if the critical population size was not determined or an EDAResults instance with the results of the runs of the EDA using the critical population size.

References

Gonzalez-Fernandez Y, Soto M (2014). **copulaedas**: An R Package for Estimation of Distribution Algorithms Based on Copulas. *Journal of Statistical Software*, **58**(9), 1-34. http://www.jstatsoft.org/v58/i09/.

Pelikan M (2005). *Hierarchical Bayesian Optimization Algorithm. Toward a New Generation of Evolutionary Algorithms*. Springer-Verlag.

See Also

EDA, edaRun.

Examples

```
setMethod("edaReport", "EDA", edaReportDisabled)
setMethod("edaTerminate", "EDA",
    edaTerminateCombined(edaTerminateEval,
        edaTerminateMaxEvals))

UMDA <- CEDA(copula = "indep", margin = "norm",
    fEval = 0, fEvalTol = 1e-03, maxEvals = 10000)
UMDA@name <- "Univariate Marginal Distribution Algorithm"

results <- edaCriticalPopSize(UMDA, fSphere, rep(-600, 10),
    rep(600, 10), 0, 1e-03, totalRuns = 30, successRuns = 30,
    lowerPop = 50, upperPop = 100, verbose = TRUE)

show(results)
summary(results)
```

---

edaIndepRuns *Independent Runs*

Description

Execute independent runs.

Usage

```
edaIndepRuns(eda, f, lower, upper, runs, verbose = FALSE)
```

Arguments

eda	EDA instance.
f	Objective function.
lower	Lower bounds of the variables of the objective function.
upper	Upper bounds of the variables of the objective function.
runs	Number of runs.
verbose	Print information after each run and a final summary.

Value

An EDAResults instance.

References

Gonzalez-Fernandez Y, Soto M (2014). **copulaedas**: An R Package for Estimation of Distribution Algorithms Based on Copulas. *Journal of Statistical Software*, **58**(9), 1-34. http://www.jstatsoft.org/v58/i09/.

See Also

EDA, edaRun.

Examples

```
setMethod("edaReport", "EDA", edaReportSimple)
setMethod("edaTerminate", "EDA",
    edaTerminateCombined(edaTerminateMaxGen,
        edaTerminateEval))

DVEDA <- VEDA(vine = "DVine", copulas = c("normal"),
    indepTestSigLevel = 0.01, margin = "norm", popSize = 200,
    maxGens = 50, fEval = 0, fEvalTol = 1e-03)
DVEDA@name <- "D-vine Estimation of Distribution Algorithm"

results <- edaIndepRuns(DVEDA, fSphere, rep(-600, 5), rep(600, 5), 5)

show(results)
summary(results)
```

---

edaOptimize *Local Optimization Methods*

Description

Methods for the edaOptimize generic function.

Usage

```
edaOptimizeDisabled(eda, gen, pop, popEval, f, lower, upper)
```

Arguments

eda	EDA instance.
gen	Generation.
pop	Matrix with one row for each solution in the population.
popEval	Vector with the evaluation of each solution in pop.
f	Objective function.
lower	Lower bounds of the variables of the objective function.
upper	Upper bounds of the variables of the objective function.

Details

Local optimization methods improve the solutions sampled by the search distribution. These methods can also be used to implement repairing strategies for constrained problems in which the simulated solutions may be unfeasible and some strategy to repair these solutions is available.

The following local optimization methods are implemented.

`edaOptimizeDisabled` Disable local optimization. This is the default method of the `edaOptimize` generic function.

Value

A `list` with the following components.

pop	Matrix with one row for each solution in the optimized population.
popEval	Vector with the evaluation of each solution in pop.

References

Gonzalez-Fernandez Y, Soto M (2014). **copulaedas**: An R Package for Estimation of Distribution Algorithms Based on Copulas. *Journal of Statistical Software*, **58**(9), 1-34. http://www.jstatsoft.org/v58/i09/.

edaReplace *Replacement Methods*

Description

Methods for the edaReplace generic function.

Usage

```
edaReplaceComplete(eda, gen, pop, popEval, sampledPop, sampledEval)
edaReplaceRTR(eda, gen, pop, popEval, sampledPop, sampledEval)
```

Arguments

eda	EDA instance.
gen	Generation.
pop	Matrix with one row for each solution in the population.
popEval	Vector with the evaluation of each solution in pop.
sampledPop	Matrix with one row for each solution sampled in the current generation.
sampledEval	Vector with the evaluation of the candidate solutions in sampledPop.

Details

Replacement methods combine the candidate solutions sampled in the current generation with the candidate solutions from the population of the previous generation. The following replacement methods are implemented.

edaReplaceComplete The population sampled in the current generation completely replaces the population of the previous generation. This is the default method of the edaReplace generic function.

edaReplaceRTR Restricted Tournament Replacement is a niching method that can be used to promote the preservation of alternative candidate solutions. See (Pelikan 2005) for a pseudocode of the algorithm implemented here. The parameter windowSize specifies the window size (default value: min(ncol(pop), nrow(pop) / 2)).

Value

A list with the following components.

pop	Matrix with one row for each solution in the new population.
popEval	Vector with the evaluation of each solution in pop.

References

Gonzalez-Fernandez Y, Soto M (2014). **copulaedas**: An R Package for Estimation of Distribution Algorithms Based on Copulas. *Journal of Statistical Software*, **58**(9), 1-34. http://www.jstatsoft.org/v58/i09/.

edaReport *Reporting Methods*

Description

Methods for the edaReport generic function.

Usage

```
edaReportDisabled(eda, gen, fEvals, model, pop, popEval)
edaReportSimple(eda, gen, fEvals, model, pop, popEval)
edaReportDumpPop(eda, gen, fEvals, model, pop, popEval)
edaReportDumpSelectedPop(eda, gen, fEvals, model, pop, popEval)
edaReportCombined(...)
```

Arguments

eda	EDA instance.
gen	Generation.
fEvals	Evaluations of the objective function.
model	Model learned in the current generation.
pop	Matrix with one row for each solution in the population.
popEval	Vector with the evaluation of each solution in pop.
...	Functions that implement reporting methods.

Details

Reporting methods provide progress information during the execution of the EDA. The following reporting methods are implemented.

edaReportDisabled Disable reporting progress. This is the default method of the edaReport generic function.

edaReportSimple Print one line at each generation with the number of generations, and the minimum, mean and standard deviation of the evaluation of the candidate solutions in the population.

edaReportDumpPop Save the population at each generation in a different plain-text file in the current working directory. The names of the files are pop_1.txt, pop_2.txt, and so on.

edaReportDumpSelectedPop Save the selected population at each generation in a different plain-text file in the current working directory. The names of the files are sel_1.txt, sel_2.txt, and so on.

edaReportCombined Execute all the reporting methods specified in ....

References

Gonzalez-Fernandez Y, Soto M (2014). **copulaedas**: An R Package for Estimation of Distribution Algorithms Based on Copulas. *Journal of Statistical Software*, **58**(9), 1-34. http://www.jstatsoft.org/v58/i09/.

`EDAResult-class` *Class for the Results of a Run of an EDA*

Description

Results of a run of an EDA. Objects are created by calling the edaRun function.

Slots

`eda`: Object of class `EDA`.

`f`: Object of class `function`. Objective function.

`lower`: Object of class `numeric`. Lower bounds of the variables of the objective function.

`upper`: Object of class `numeric`. Upper bounds of the variables of the objective function.

`numGens`: Object of class `numeric`. Number of generations.

`fEvals`: Object of class `numeric`. Number of evaluations of the objective function.

`bestEval`: Object of class `numeric`. Best evaluation of the objective function.

`bestSol`: Object of class `numeric`. Best solution.

`cpuTime`: Object of class `numeric`. Run time of the algorithm in seconds.

Methods

show `signature(object = "EDAResult")`: Prints the results.

References

Gonzalez-Fernandez Y, Soto M (2014). **copulaedas**: An R Package for Estimation of Distribution Algorithms Based on Copulas. *Journal of Statistical Software*, **58**(9), 1-34. http://www.jstatsoft.org/v58/i09/.

See Also

EDA, edaRun.

EDAResults-class *Class for the Results of a Sequence of Runs of an EDA*

Description

Results of a sequence of independent runs of an EDA. This class is just a wrapper for a `list` object containing `EDAResult` instances. Objects are created by calling the `edaIndepRuns` function.

Methods

show `signature(object = "EDAResults")`: Prints a table with the results.

summary `signature(object = "EDAResults")`: Prints a summary of the results.

References

Gonzalez-Fernandez Y, Soto M (2014). **copulaedas**: An R Package for Estimation of Distribution Algorithms Based on Copulas. *Journal of Statistical Software*, **58**(9), 1-34. http://www.jstatsoft.org/v58/i09/.

See Also

`EDA`, `edaIndepRuns`.

edaRun *Main Loop of an EDA*

Description

Main loop of an EDA.

Usage

```
edaRun(eda, f, lower, upper)
```

Arguments

`eda`	`EDA` instance.
`f`	Objective function.
`lower`	Lower bounds of the variables of the objective function.
`upper`	Upper bounds of the variables of the objective function.

Details

EDAs are implemented using S4 classes with generic functions for its main parts: seeding (edaSeed), selection (edaSelect), learning (edaLearn), sampling (edaSample), replacement (edaReplace), local optimization (edaOptimize), termination (edaTerminate), and reporting (edaReport). The following pseudocode illustrates the interactions between all the generic functions. It is a simplified version of the implementation of the edaRun function.

```
gen <- 0
fEvals <- 0
terminate <- FALSE

while (!terminate) {
  gen <- gen + 1

  if (gen == 1) {
    model <- NULL
    pop <- edaSeed(lower, upper)
    # Set popEval to the evaluation of each solution in pop.
    # Update fEvals.
    r <- edaOptimize(gen, pop, popEval, f, lower, upper)
    pop <- r$pop; popEval <- r$popEval
  } else {
    s <- edaSelect(gen, pop, popEval)
    selectedPop <- pop[s, ]; selectedEval <- popEval[s]
    model <- edaLearn(gen, model, selectedPop, selectedEval,
                      lower, upper)
    sampledPop <- edaSample(gen, model, lower, upper)
    # Set sampledEval to the evaluation of each solution
    # in sampledPop. Update fEvals.
    r <- edaOptimize(gen, sampledPop, sampledEval, f, lower, upper)
    sampledPop <- r$pop; sampledEval <- r$popEval
    r <- edaReplace(gen, pop, popEval, sampledPop, sampledEval)
    pop <- r$pop; popEval <- r$popEval
  }

  edaReport(gen, fEvals, model, pop, popEval)
  terminate <- edaTerminate(gen, fEvals, pop, popEval)
}
```

Value

An EDAResult instance.

References

Gonzalez-Fernandez Y, Soto M (2014). **copulaedas**: An R Package for Estimation of Distribution Algorithms Based on Copulas. *Journal of Statistical Software*, **58**(9), 1-34. http://www.jstatsoft.org/v58/i09/.

See Also

EDA, EDAResult, edaIndepRuns.

Examples

```
setMethod("edaReport", "EDA", edaReportSimple)
setMethod("edaTerminate", "EDA",
    edaTerminateCombined(edaTerminateMaxGen,
        edaTerminateEval))

DVEDA <- VEDA(vine = "DVine", copulas = c("normal"),
    indepTestSigLevel = 0.01, margin = "norm",
    popSize = 200, maxGens = 50, fEval = 0,
    fEvalTol = 1e-03)
DVEDA@name <- "D-vine Estimation of Distribution Algorithm"

result <- edaRun(DVEDA, fSphere, rep(-600, 5), rep(600, 5))

show(result)
```

---

edaSeed *Seeding Methods*

Description

Methods for the edaSeed generic function.

Usage

```
edaSeedUniform(eda, lower, upper)
```

Arguments

eda	EDA instance.
lower	Lower bounds of the variables of the objective function.
upper	Upper bounds of the variables of the objective function.

Details

Seeding methods create the initial population. The length of the lower and upper vectors determine the number of variables of the objective function. The following seeding methods are implemented.

edaSeedUniform Sample each variable from a continuous uniform distribution in the interval determined by lower and upper. The parameter popSize sets the number of solutions in the population (default value: 100). This is the default method of the edaSeed generic function.

Value

A matrix with one column for each variable of the objective function and one row for each solution in the population.

References

Gonzalez-Fernandez Y, Soto M (2014). **copulaedas**: An R Package for Estimation of Distribution Algorithms Based on Copulas. *Journal of Statistical Software*, **58**(9), 1-34. http://www.jstatsoft.org/v58/i09/.

`edaSelect` *Selection Methods*

Description

Methods for the `edaSelect` generic function.

Usage

```
edaSelectTruncation(eda, gen, pop, popEval)
edaSelectTournament(eda, gen, pop, popEval)
```

Arguments

`eda`	`EDA` instance.
`gen`	Generation.
`pop`	Matrix with one row for each solution in the population.
`popEval`	Vector with the evaluation of each solution in `pop`.

Details

Selection methods determine the solutions to be modeled by the search distribution (selected population). These solutions are usually the most promising solutions of the population. The following selection methods are implemented.

`edaSelectTruncation` In truncation selection, the `100 * truncFactor` percent of the solutions with the best evaluation in the population are selected. The parameter `truncFactor` specifies the truncation factor (default value: `0.3`). This is the default method of the `edaSelect` generic function.

`edaSelectTournament` In tournament selection, a group of solutions are randomly picked from the population and the best one is selected. This process is repeated as many times as needed to complete the selected population. The parameter `tournamentSize` specifies the number of solutions randomly picked from the population (default value: `2`), `selectionSize` specifies the size of the selected population (default value: `nrow(pop)`), and `replacement` specifies whether to sample with replacement or not (default value: `TRUE`).

Value

An integer vector with the indexes of the solutions selected from pop.

References

Gonzalez-Fernandez Y, Soto M (2014). **copulaedas**: An R Package for Estimation of Distribution Algorithms Based on Copulas. *Journal of Statistical Software*, **58**(9), 1-34. http://www.jstatsoft.org/v58/i09/.

Pelikan M (2005). *Hierarchical Bayesian Optimization Algorithm. Toward a New Generation of Evolutionary Algorithms*. Springer-Verlag.

---

edaTerminate *Termination Methods*

Description

Methods for the edaTerminate generic function.

Usage

```
edaTerminateMaxGen(eda, gen, fEvals, pop, popEval)
edaTerminateMaxEvals(eda, gen, fEvals, pop, popEval)
edaTerminateEval(eda, gen, fEvals, pop, popEval)
edaTerminateEvalStdDev(eda, gen, fEvals, pop, popEval)
edaTerminateCombined(...)
```

Arguments

eda	EDA instance.
gen	Generation.
fEvals	Evaluations of the objective function.
pop	Matrix with one row for each solution in the population.
popEval	Vector with the evaluation of each solution in pop.
...	Functions that implement termination methods.

Details

Termination methods decide when to stop the main loop of the EDA. The following termination methods are implemented.

edaTerminateMaxGen Stop when a maximum number of generations has been reached. The parameter maxGen specifies the number of generations (default value: 100). This is the default method of the edaTerminate generic function.

edaTerminateMaxEvals Stop when a maximum number of evaluations of the objective function has been reached. The parameter maxEvals specifies the number of evaluations (default value: 1000.)

`edaTerminateEval` Stop when a given value of the objective function has been reached. The parameters `fEval` (default value: `0`) and `fEvalTol` (default value: `1e-06`) set the value of the objective function and the tolerance, respectively.

`edaTerminateEvalStdDev` Stop when the standard deviation of the evaluation of the solutions in the population is less than the value given by the parameter `fEvalStdDev` (default value: `1e-02`).

`edaTerminateCombined` Evaluate all the termination criteria specified in `...` and stop if (at least) one of them returns `TRUE`.

Value

A `logical` value that indicates if the algorithm should stop.

References

Gonzalez-Fernandez Y, Soto M (2014). **copulaedas**: An R Package for Estimation of Distribution Algorithms Based on Copulas. *Journal of Statistical Software*, **58**(9), 1-34. http://www.jstatsoft.org/v58/i09/.

`margins` *Marginal Distributions*

Description

Functions that implement marginal distributions.

Usage

```
fnorm(x, lower, upper)

ftruncnorm(x, lower, upper)

fkernel(x, lower, upper)
pkernel(q, X, h)
qkernel(p, X, h)

ftrunckernel(x, lower, upper)
ptrunckernel(q, a, b, X, h)
qtrunckernel(p, a, b, X, h)
```

Arguments

`x`, `q`	Vector of quantiles.
`lower`, `a`	Lower bound of the variable.
`upper`, `b`	Upper bound of the variable.
`p`	Vector of probabilities
`X`	Observations of the variable.
`h`	Bandwidth of the kernel.

Details

The functions fnorm, pnorm, and qnorm implement the normal marginal distributions for EDAs with the margin parameter set to "norm". The fnorm function fits the parameters, it returns a list object with the mean (mean component) and the standard deviation (sd component). These components determine the values of the corresponding arguments of the pnorm and qnorm functions.

The functions ftruncnorm, ptruncnorm, and qtruncnorm implement the normal marginal distributions for EDAs with the margin parameter set to "truncnorm". The ftruncnorm function fits the parameters, it returns a list object with the lower and upper bounds (a and b components, respectively), the mean (mean component) and the standard deviation (sd component). These components determine the values of the corresponding arguments of the ptruncnorm and qtruncnorm functions.

The functions fkernel, pkernel, and qkernel implement the kernel-smoothed empirical marginal distributions for EDAs with the margin parameter set to "kernel". The fkernel function fits the marginal distribution, it returns a list object with the observations of the variable (X component) and the bandwidth of a Gaussian kernel density estimator (h component). The bandwidth is calculated using Silverman's rule of thumb (see bw.nrd0). The components of the list object returned by fkernel are used as aditional arguments in the pkernel and qkernel functions. The pkernel function calculates the empirical cumulative distribution function. The expression of the empirical cumulative distribution function includes the modification used in the copula context to avoid problems in the boundary of the $[0, 1]$ interval. The qkernel function uses the Gaussian kernel density estimator fitted by fkernel to evaluate the inverse of the cumulative distribution function, following the procedure suggested in (Azzalini 1981).

The functions ftrunckernel, ptrunckernel, and qtrunckernel implement the truncated kernel-smoothed empirical marginal distributions for EDAs with the margin parameter set to "trunckernel". The distribution is computed from the corresponding kernel-smoothed empirical marginal distributions without truncation by following the procedure illustrated in (Nadarajah and Kotz 2006).

References

Azzalini, A (1981) A Note on the Estimation of a Distribution Function and Quantiles by a Kernel Method, *Biometrika*, **68**, 326-328.

Gonzalez-Fernandez Y, Soto M (2014). **copulaedas**: An R Package for Estimation of Distribution Algorithms Based on Copulas. *Journal of Statistical Software*, **58**(9), 1-34. http://www.jstatsoft.org/v58/i09/.

Nadarajah S, Kotz S (2006) R Programs for Computing Truncated Distributions, *Journal of Statistical Software*, **16**, Code Snippet 2.

See Also

pnorm, qnorm, ptruncnorm, qtruncnorm.

problems *Benchmark Problems*

Description

Implementation of a group of well-known benchmark problems typically used to evaluate the performance of EDAs and other numerical optimization algorithms for unconstrained global optimization.

Usage

```
fAckley(x)
fGriewank(x)
fRosenbrock(x)
fRastrigin(x)
fSphere(x)
fSummationCancellation(x)
```

Arguments

x	A vector to be evaluated in the function.

Details

The definition of the functions for a vector $\boldsymbol{x} = (x_1, \ldots, x_n)$ is given below.

$$\texttt{fAckley}(\boldsymbol{x}) = -20 \exp\left(-0.2 \sqrt{\frac{1}{n} \sum_{i=1}^{n} x^2}\right) - \exp\left(\frac{1}{n} \sum_{i=1}^{n} \cos\left(2\pi x_i\right)\right) + 20 + \exp\left(1\right)$$

$$\texttt{fGriewank}(\boldsymbol{x}) = 1 + \sum_{i=1}^{n} \frac{x_i^2}{4000} - \prod_{i=1}^{n} \cos\left(\frac{x_i}{\sqrt{i}}\right)$$

$$\texttt{fRastrigin}(\boldsymbol{x}) = \sum_{i=1}^{n} \left(x_i^2 - 10\cos\left(2\pi x_i\right) + 10\right)$$

$$\texttt{fRosenbrock}(\boldsymbol{x}) = \sum_{i=1}^{n-1} \left(100\left(x_{i+1} - x_i^2\right)^2 + \left(1 - x_i\right)^2\right)$$

$$\texttt{fSphere}(\boldsymbol{x}) = \sum_{i=1}^{n} x_i^2$$

$$\texttt{fSummationCancellation}(\boldsymbol{x}) = \frac{-1}{10^{-5} + \sum_{i=1}^{n} |y_i|},\ y_1 = x_1,\ y_i = y_{i-1} + x_i$$

Ackley, Griewank, Rastrigin, Rosenbrock, and Sphere are minimization problems. Summation Cancellation is originally a maximization problem but it is expressed here as a minimization problem. Ackley, Griewank, Rastrigin and Sphere have their global optimum at $\boldsymbol{x} = (0, \ldots, 0)$ with evaluation 0. Rosenbrock has its global optimum at $\boldsymbol{x} = (1, \ldots, 1)$ with evaluation 0. Summation Cancellation has its global optimum at $\boldsymbol{x} = (0, \ldots, 0)$ with evaluation -10^5. See (Bengoetxea et al. 2002; Bosman and Thierens 2006; Chen and Lim 2008) for a description of the functions.

Value

The value of the function for the vector `x`.

References

Bengoetxea E, Miquélez T, Lozano JA, Larrañaga P (2002). Experimental Results in Function Optimization with EDAs in Continuous Domain. In P Larrañaga, JA Lozano (eds.), *Estimation of Distribution Algorithms. A New Tool for Evolutionary Computation*, pp. 181–194. Kluwer Academic Publisher

Bosman PAN, Thierens D (2006). Numerical Optimization with Real-Valued Estimation of Distribution Algorithms. In M Pelikan, K Sastry, E Cantú-Paz (eds.), *Scalable Optimization via Probabilistic Modeling. From Algorithms to Applications*, pp. 91–120. Springer-Verlag.

Chen Yp, Lim MH (eds.) (2008). Linkage in Evolutionary Computation. Springer-Verlag. ISBN 978-3-540-85067-0.

Gonzalez-Fernandez Y, Soto M (2014). **copulaedas**: An R Package for Estimation of Distribution Algorithms Based on Copulas. *Journal of Statistical Software*, **58**(9), 1-34. http://www.jstatsoft.org/v58/i09/.

Examples

```
all.equal(fAckley(rep(0, 10)), 0)
all.equal(fGriewank(rep(0, 10)), 0)
all.equal(fRastrigin(rep(0, 10)), 0)
all.equal(fRosenbrock(rep(1, 10)), 0)
all.equal(fSphere(rep(0, 10)), 0)
all.equal(fSummationCancellation(rep(0, 10)), -1e+05)
```

VEDA-class *Class for Vine EDAs*

Description

Extends the `EDA` class to implement EDAs based on vines. Objects are created by calling the `VEDA` function.

Details

Vine EDAs (VEDAs) are a class of EDAs (Soto and Gonzalez-Fernandez 2010; Gonzalez-Fernandez 2011) that model the search distributions using vines. Vines are graphical models that represent high-dimensional distributions by decomposing the multivariate density into conditional bivariate copulas, unconditional bivariate copulas, and one-dimensional densities (Joe 1996; Bedford and Cooke 2001; Aas et al. 2009; Kurowicka and Cooke 2006). In particular, VEDAs are based on the simplified pair-copula construction (Hobaek Haff et al. 2010). Similarly to Copula EDAs, these algorithms estimate separately the univariate marginal distributions and the dependence structure from the selected population. Instead of representing the dependence structure using a single multivariate copula, VEDAs can model a rich variety of dependencies by combining bivariate copulas that belong to different families. The following instances of VEDA are implemented.

- C-vine EDA (CVEDA), that models the search distributions using C-vines (Soto and Gonzalez-Fernandez 2010; Gonzalez-Fernandez 2011).
- D-vine EDA (DVEDA), that models the search distributions using D-vines (Soto and Gonzalez-Fernandez 2010; Gonzalez-Fernandez 2011).

Greedy heuristics based on the empirical Kendall's tau between each variable in the selected population are used to determine the structure of the C-vines and D-vines in CVEDA and DVEDA, respectively (Brechmann 2010).

The selection of each bivariate copula in both decompositions starts with an independence test (Genest and Rémillard 2004; Genest et al. 2007). The independence copula is selected if there is not enough evidence against the null hypothesis of independence at a given significance level. In the other case, the parameters of a group of candidate copulas are estimated and the one that minimizes a distance to the empirical copula is selected. A Cramér-von Mises statistic is used as the measure of distance (Genest and Rémillard 2008).

The parameters of all the candidate copulas but the t copula are estimated using the inversion of Kendall's tau. In the case of the t copula, the correlation coefficient is computed using the inversion of Kendall's tau and the degrees of freedom are estimated by maximum likelihood with the correlation parameter fixed (Demarta and McNeil 2005).

To simplify the construction of the vines the truncation strategy presented in (Brechmann 2010) is applied. If a vine is truncated at a given tree, all the copulas in the subsequent trees are assumed to be product copulas. By default, a model selection procedure based on AIC (Akaike Information Criterion) is applied to detect the required number of trees, but it is also possible to base the selection on BIC (Bayesian Information Criterion) or completely disable the truncation strategy. Also, a maximum number of dependence trees of the vine can be set, which may be helpful when dealing with high-dimensional problems.

The following parameters are recognized by the functions that implement the `edaLearn` and `edaSample` methods for the `VEDA` class.

`vine` Vine type. Supported values are: `"CVine"` (Canonical vine) and `"DVine"` (D-vine). Default value: `"DVine"`.

`trees` Maximum number of dependence trees of the vine. The default is to estimate a full vine.

`truncMethod` Method used to automatically truncate the vine if enough dependence is captured in the first trees. Supported values are: `"AIC"`, `"BIC"` and `""` (no truncation). Default value: `"AIC"`.

`copulas` A `character` vector specifying the candidate copulas. Supported values are: `"normal"` (normal copula), `"t"` (t copula), `"clayton"` (Clayton copula), `"frank"` (Frank copula), and `"gumbel"` (Gumbel copula). Default value: `c("normal")`.

`indepTestSigLevel` Significance level of the independence test. Default value: `0.01`.

`margin` Marginal distributions. If this argument is `"xxx"`, the algorithm will search for three functions named `fxxx`. `pxxx` and `qxxx` to fit each marginal distribution and evaluate the cumulative distribution function and its inverse, respectively. Default value: `"norm"`.

`popSize` Population size. Default value: `100`.

Slots

`name`**:** See the documentation of the slot in the `EDA` class.

`parameters`**:** See the documentation of the slot in the `EDA` class.

Methods

edaLearn `signature(eda = "CEDA")`: The `edaLearnCEDA` function.

edaSample `signature(eda = "CEDA")`: The `edaSampleCEDA` function.

References

Aas K, Czado C, Frigessi A, Bakken H (2009). Pair-Copula Constructions of Multiple Dependence. *Insurance: Mathematics and Economics*, **44**(2), 182–198.

Bedford T, Cooke RM (2001). Probability Density Decomposition for Conditionally Dependent Random Variables Modeled by Vines. *Annals of Mathematics and Artificial Intelligence*, **32**(1), 245–268.

Brechmann EC (2010). Truncated and Simplified Regular Vines and Their Applications. Diploma thesis, University of Technology, Munich, Germany.

Demarta S, McNeil AJ (2005). The t Copula and Related Copulas. *International Statistical Review*, **73**(1), 111–129.

Genest C, Rémillard B (2004). Tests of Independence or Randomness Based on the Empirical Copula Process. *Test*, **13**(2), 335–369.

Genest C, Quessy JF, Rémillard B (2007). Asymptotic Local Efficiency of Cramér-von mises Tests for Multivariate Independence. *The Annals of Statistics*, **35**, 166–191.

Genest C, Rémillard B (2008). Validity of the Parametric Bootstrap for Goodness-of-Fit Testing in Semiparametric Models. *Annales de l'Institut Henri Poincaré: Probabilités et Statistiques*, **44**, 1096–1127.

Gonzalez-Fernandez Y (2011). Algoritmos con estimación de distribuciones basados en cópulas y vines. Bachelor's thesis, University of Havana, Cuba.

Gonzalez-Fernandez Y, Soto M (2014). **copulaedas**: An R Package for Estimation of Distribution Algorithms Based on Copulas. *Journal of Statistical Software*, **58**(9), 1-34. http://www.jstatsoft.org/v58/i09/.

Hobaek Haff I, Aas K, Frigessi A (2010). On the Simplified Pair-Copula Construction — Simply Useful or Too Simplistic? Journal of Multivarite Analysis, **101**, 1145–1152.

Joe H (1996). Families of m-variate Distributions with Given Margins and $m(m-1)/2$ Bivariate Dependence Parameters. In L Röschendorf, B Schweizer, MD Taylor (eds.), *Distributions with fixed marginals and related topics*, pp. 120–141.

Soto M, Gonzalez-Fernandez Y (2010). Vine Estimation of Distribution Algorithms. Technical Report ICIMAF 2010-561, Institute of Cybernetics, Mathematics and Physics, Cuba. ISSN 0138-8916.

Examples

```
setMethod("edaTerminate", "EDA", edaTerminateEval)
setMethod("edaReport", "EDA", edaReportSimple)

CVEDA <- VEDA(vine = "CVine",
    copulas = c("normal", "clayton", "frank", "gumbel"),
    indepTestSigLevel = 0.01, margin = "norm",
    popSize = 200, fEval = 0, fEvalTol = 1e-03)
CVEDA@name <- "C-vine Estimation of Distribution Algorithm"

DVEDA <- VEDA(vine = "DVine",
    copulas = c("normal", "clayton", "frank", "gumbel"),
    indepTestSigLevel = 0.01, margin = "norm",
    popSize = 200, fEval = 0, fEvalTol = 1e-03)
DVEDA@name <- "D-vine Estimation of Distribution Algorithm"

resultsCVEDA <- edaRun(CVEDA, fSphere, rep(-600, 5), rep(600, 5))
resultsDVEDA <- edaRun(DVEDA, fSphere, rep(-600, 5), rep(600, 5))

show(resultsCVEDA)
show(resultsDVEDA)
```

Index