Package 'acopula'

September 10, 2023

Type Package

Title Modelling Dependence with Multivariate Archimax (or any User-Defined Continuous) Copulas

Version 0.9.4

Date 2023-09-10

Author Tomas Bacigal

Maintainer Tomas Bacigal <bacigal@math.sk>

Suggests mvtnorm

Description Archimax copulas are mixture of Archimedean and EV copulas. The package provides definitions of several parametric families of generator and dependence function, computes CDF and PDF, estimates parameters, tests for goodness of fit, generates random sample and checks copula properties for custom constructs. In 2-dimensional case explicit formulas for density are used, contrary to higher dimensions when all derivatives are linearly approximated. Several non-archimax families (normal, FGM, Plackett) are provided as well.

License GPL-2

Encoding UTF-8

NeedsCompilation no

Repository CRAN

Date/Publication 2023-09-10 20:30:02 UTC

R topics documented:

acopula-package 2
copula 3
depfun 4
generator 7
nderive 8
nintegrate 9
vpartition 10
xCopula 11
xPareto 16

Index **18**

acopula-package *Modelling dependence with multivariate Archimax (or any user-defined continuous) copulas.*

Description

Archimax copulas are mixture of Archimedean and EV copulas. The package provides definitions of several parametric families of generator and dependence function, computes CDF and PDF, estimates parameters, tests for goodness of fit, generates random sample and checks copula properties for custom constructs. In 2-dimensional case explicit formulas for density are used, in the contrary to higher dimensions when all derivatives are linearly approximated. Several non-archimax families (normal, FGM, Plackett) are provided as well.

Details

Generators, dependence functions and generic copula definition lists are generated by functions named starting with `gen`, `dep` and `cop`, respectively. Definition lists are supplied to functions `xCopula` where `x` stands for `p` (probability distribution f., CDF), `d` (probability density f., PDF), `r` (random sampling), `c` (conditional CDF), `q` (quantile function), `e` (parameter estimation), `g` (goodness-of-fit test), `is` (copula properties check).

Author(s)

Tomas Bacigal

Maintainer: Tomas Bacigal <bacigal@math.sk>

References

Bacigál, T. (2012): Recent tools for modelling dependence with copulas and R. *Forum Statisticum Slovacum* 8(1), 62–67.

Capéraá, P., Fougéres, A.-L., Genest, C. (2000): Bivariate distributions with given extreme value attractor. *J.Multivariate Anal.* 72(1) 30–49.

Mesiar, R., Jágr, V. (2012): d-Dimensional dependence functions and Archimax copulas. *Fuzzy Sets and Systems*, in press.

Bacigál, T. (2013): R Package to Handle Archimax or Any User-Defined Continuous Copula Construction. Aggregation Functions in Theory and in Practise. Springer Berlin Heidelberg, 75–84.

`copula` *Generic copulae definitions*

Description

Produce a list containing CDF and/or PDF of an copula with parameters bounds.

Usage

```
copula(name,...)

copFGM(...)
copGumbel(...)
copNormal(dim=2,...)
copPlackett(...)
copProduct(...)
```

Arguments

`name`	character. Code name for copula, identical with the part after `cop`.
`dim`	integer. Copula dimension.
`...`	named arguments. Items of the copula definition list to be redefined. Argument names must NOT be cropped.

Details

Currently implemented copula families:

family	**CDF** $C(t) =$	p**ar.range**	**special cases**
Farlie-Gumbel-Morgenstern	$\prod_i t_i(p\prod_i(1-t)+1)$	[-1,1]	0(Π)
Gumbel-Hougaard	$\exp((-\sum_i(-\log t_i)^p)^{1/p})$	[1,Inf]	1(Π),Inf(M)
normal	$\Phi_P(\Phi^{-1}(t_1),...)$	]-1,1[	-1(W),0(Π),1(M)
Plackett	$\frac{A-\sqrt{A^2-4\prod_i t_i p(p-1)}}{2(p-1)}; A=1+(p-1)\sum_i t_i$	[0,Inf]	0(W),1(Π),Inf(M)
product	$-\prod_i t_i$		Π

where Φ^{-1} is quantile function of standard normal and Φ_P is CDF of multivariate normal distribution with correlation matrix P containing copula parameters.

Value

`parameters`	numeric vector to be used whenever parameters of copula are not supplied to procedure that use it, or as starting values in estimation.
`pcopula,dcopula`	function of two arguments. The first is vector of copula arguments, the another is vector of parameters.

rcopula	function of two arguments. The first is copula dimension or length of random sample (normal copula), the another is vector of parameters.
kendall,spearman	list. Correlation coefficient as function of copula parameter (coef), its inverse (icoef) and range (bounds). Available only for 1-parameter families.
lower,upper	numeric. Parameters boundary.
id	character. Identification of copula family.

Author(s)

Tomas Bacigal

References

Nelsen, R. B. (2006): An introduction to copulas. Springer.

See Also

pCopula, generator, depfun

Examples

```
## the following gives the same definition list
copFGM()
copula("FGM")

## any list item can be modified upon function call
copPlackett(parameters=2.2,upper=10)
```

---

depfun *Dependence function of Extreme-Value copula*

Description

Produce a list containing dependence function of specified EV copula family, its derivatives and parameters bounds. Only Hussler-Reiss family is limited to two dimensions.

ldepPartition3D returns set of 5 dependence functions (see details).

Usage

```
depfun(name, ...)

dep1(...)
depGalambos(...)
depGumbel(...)
depHuslerReiss(...)
depMax(power = 10, ...)
```

```
depTawn(dim = 2, ...)

depCC(depfun = list(dep1(),depGumbel()),
  dparameters = lapply(depfun,
    function(x) rep(list(NULL),max(1,length(x$parameters)))),
  dim = 2)
depGCC(depfun=list(dep1(),depGumbel()),
  dparameters = lapply(depfun,
    function(x) rep(list(NULL),max(1,length(x$parameters)))),
  dim = 2, symmetry = FALSE)

ldepPartition3D(power = 8)
```

Arguments

`name`	character. Code name for Pickands' dependence function, identical with the part after `dep`.
`power`	numeric. Parameter of Gumbel family dependence function, which approximates the weakest dependence function in order to bring smoothness.
`dim`	numeric. Dimension (of copula) of random vector.
`depfun`	list of dependence function definition lists, also `ldepPartition3D` can be used.
`dparameters`	list of dependence function parameters; defaults to list of `NULL`s which means the parameters are to be estimated.
`symmetry`	logical. If TRUE, then GCC reduces to standard convex sum and `depCC` is used.
`...`	named arguments. Items of the dependence function definition list to be redefined.

Details

Currently implemented families of EV copula dependence functions:

family	**dependence function** $A(t) =$	**domain**	EV.case
1	1		Π
Galambos	$1-(\sum_i t_i^{-p})^{-1/p}$	[0,10]	1(Π),Inf(M)
Gumbel-Hougaard	$(\sum_i (t_i^p))^{1/p}$	[1,Inf]	1(Π),Inf(M)
Husler-Reiss	$t_1\Phi(1/p+p\log(t_1/t_2)/2)+$ $+t_2\Phi(1/p-p\log(t_1/t_2)/2)$	[0,Inf]	0(Π),Inf(M)
Max	$(\sum_i t_i^{10})^{1/10}$		M
Tawn	$1-\sum_i p_i t_i+(\sum_i (p_i t_i)^{p_0})^{1/p_0}$	[1,Inf]x[0,1]x...	{1,0,...}(W),{Inf,1,...}(M)

Since $\sum_i t_i = 1$ a dependence function accepts argument vector with the last element omitted.

Value

`parameters`	numeric vector to be used whenever parameters of depfun are not supplied to procedure that use it, or as starting values in estimation

`dep`	function of two arguments; the first is depfun argument, the another is depfun parameters
`dep.der`	function; depfun first derivative
`dep.der2`	function; depfun second derivative
`kendall,spearman`	list. Correlation coefficient as function of copula parameter (`coef`), its inverse (`icoef`) and range (`bounds`). Available only for 1-parameter families.
`lower,upper`	numeric; parameters boundary
`id`	character; identification of depfun family
`combpars,rescalepars`	function; extract the combination parameters from the set of provided parameters and rescale them if not fulfilling inner conditions of the (general) convex combination

Author(s)

Tomas Bacigal

References

Bacigál, T., Mesiar, R.: 3-dimensional Archimax copulas and their fitting to real data. In: *COMPSTAT 2012, 20th International conference on computational statistics*. Limassol,Cyprus,27.-31.8.2012. The International Statistical Institute, 81–88 (2012).

Gudendorf, G., Segers, J. (2010): Extreme-value copulas. In *Copula Theory and Its Applications*. Springer Berlin Heidelberg, 127-145.

Insightful Corp.: EVANESCE Implementation in S-PLUS FinMetrics Module (2002). https://faculty.washington.edu/ezivot/book/QuanCopula.pdf Cited 6th July 2013.

See Also

`pCopula`, `generator`, `copula`

Examples

```
## the following gives the same definition list
depGumbel()
depfun("Gumbel")

## any list item can be modified upon function call
depGumbel(parameters=2.2,upper=10)

## general convex combination of 5 basic depfuns that arise from
## partitioning method for 3 dimensions; it results in
## (3x5)-parametric Pickand's dependence function definition list
depGCC(depfun=ldepPartition3D(), dim = 3)
```

`generator` *Generator of Archimedean copula*

Description

Produce a list containing generator of specified Archimedean family, its inverse and derivatives with parameters bounds.

Usage

```
generator(name,...)

genAMH(...)
genClayton(...)
genFrank(...)
genGumbel(...)
genJoe(...)
genLog(...)
```

Arguments

`name`	character; code name for generator, identical with the part after 'gen'
`...`	named arguments; items of the generator definition list to be redefined

Details

Currently implemented families of Archimedean copula generator:

family	**generator** $\phi(t) =$	p**ar.range**	**Archimed.case**
Ali-Mikhail-Haq	$\log\left(\frac{1-(1-t)p}{t}\right)$	[-1,1[	-1(Π)
Clayton	$t^(-p)-1$	[0,Inf]	0(Π),Inf(M)
Frank	$-\log\left(\frac{\exp(-pt)-1}{\exp(-p)-1}\right)$	[-Inf,Inf]	-Inf(W),0(Π),Inf(M)
Gumbel-Hougaard	$(-\log(t))^p$	[1,Inf]	1(Π),Inf(M)
Joe	$-\log(1-(1-t)^p)$	[1,Inf]	1(Π),Inf(M)
Log	$-\log(t)$		Π

Value

`parameters`	numeric vector to be used whenever parameters of generator are not supplied to procedure that use it, or as starting values in estimation.
`gen`	function of two arguments. The first is generator argument, the another is genereator parameters.
`gen.der`	function. Generator first derivative.
`gen.der2`	function. Generator second derivative.

`gen.inv`	function. Generator inverse.
`gen.inv.der`	function. First derivative of generator inverse.
`gen.inv.der2`	function. second derivative of generator inverse.
`kendall,spearman`	list. Correlation coefficient as function of copula parameter (`coef`), its inverse (`icoef`) and range (`bounds`). Available only for 1-parameter families.
`lower,upper`	numeric; parameters boundary
`id`	character; identification of generator family

Author(s)

Tomas Bacigal

References

Nelsen, R. B.: An introduction to copulas. Springer (2006).

See Also

`pCopula`, `depfun`, `copula`

Examples

```
## the following gives the same definition list
genGumbel()
generator("Gumbel")

## any list item can be modified upon function call
genGumbel(parameters=2.2,upper=10)
```

---

nderive *Numerical derivative*

Description

Linear approximation to function partial derivatives of arbitrary order and dimension.

Usage

```
nderive(fun, point = c(0), order = c(1), difference = 1e-04, area = c(0, 1, -1))
```

Arguments

`fun`	function. Arguments can be in sequence or single vector.
`point`	numeric vector. Where to evaluate the derivative.
`order`	numeric vector of orders for each variable.
`difference`	numeric. A change in the variable that (by limit) is to approach zero.
`area`	numeric. One of {0,1,-1} representing two-sided, right-sided or left-sided limit, respectively.

Value

Numeric.

Author(s)

Tomas Bacigal

See Also

nintegrate

Examples

```
##density of a bivariate Gumbel copula evaluated in point c(0.5,0.6)
nderive(fun = function(x) pCopula(x,genGumbel(),gpar=3.5), point = c(0.5,0.6),
        order = c(1,1))
```

---

nintegrate *Numerical integration*

Description

Trapezoidal integration of an arbitrarily dimensional function.

Usage

```
nintegrate(fun, lower, upper, subdivisions=100, differences=(upper-lower)/subdivisions)
```

Arguments

fun	function. Arguments can be in sequence or single vector.
lower,upper	numeric (vector). Upper and lower bound of integration. Either one of these or differences should have length of function dimension.
subdivisions	numeric (vector). Number of bins.
differences	numeric. Width of bins.

Value

Numeric.

Author(s)

Tomas Bacigal

See Also

nderive

Examples

```
##cumulative distribution function of a bivariate normal copula
##evaluated at point c(0.5,0.6); compare pCopula(c(0.5,0.6),cop=copNormal(),par=0.5)
nintegrate(function(x) dCopula(x,cop=copNormal(),par=0.5),
  lower=0.001, upper=c(0.5,0.6), subdivisions=20)
```

---

vpartition *Vector partitioning*

Description

Split a vector to subvectors of specified lengths.

Usage

```
vpartition(x, lengths, matrixify = TRUE)
```

Arguments

x	vector to be splitted.
lengths	numeric vector. Lengths of the subvectors.
matrixify	logical. Whether to return matrix if lengths are identical.

Details

If `sum(lengths)>length(x)` holds true, the extra places are filled with `NA`s.

Value

List of numeric vectors, or a matrix if all lengths are equal.

Author(s)

Tomas Bacigal

Examples

```
vpartition(1:10,c(4,5,2))
```

xCopula *Archimax and generic copula distribution functions*

Description

Copula cumulative distribution function ('p'), probability density function ('d'), conditional probability cdf (c), quantile (q), random vector sampling ('r'), parameters estimation ('e'), goodness-of-fit test ('g'), checking copula properties ('is').

Usage

```
pCopula(data, generator=genGumbel(), depfun=dep1(), copula=NULL,
  gpars=generator$parameters, dpars=depfun$parameters,
  pars=if(is.null(copula)) list(gpars,dpars) else copula$parameters,
  subdivisions=50, quantile=NULL,probability=data[,quantile])

pCopulaEmpirical(data, base = data)

dCopula(data,generator=genGumbel(),depfun=dep1(),copula=NULL,
  gpars=generator$parameters, dpars=depfun$parameters,
  pars=if(is.null(copula)) list(gpars,dpars) else copula$parameters,
  difference=1e-4,area=c(0), shrinkdiff=FALSE)

cCopula(data, conditional.on=c(1), generator=genGumbel(), depfun=dep1(), copula=NULL,
  gpars=generator$parameters, dpars=depfun$parameters,
  pars=if(is.null(copula)) list(gpars,dpars) else copula$parameters,
  difference=1e-4,area=c(0), quantile=NULL, probability=data[,quantile])

qCopula(data, quantile=1, probability=0.95, conditional.on=NULL,
  generator=genGumbel(), depfun=dep1(), copula=NULL,
  gpars=generator$parameters, dpars=depfun$parameters,
  pars=if(is.null(copula)) list(gpars,dpars) else copula$parameters,
  difference=1e-4, area=c(0))

rCopula(n, dim=2, generator=genGumbel(), depfun=dep1(), copula=NULL,
  gpars=generator$parameters, dpars=depfun$parameters,
  pars=if(is.null(copula)) list(gpars,dpars) else copula$parameters)

rCopulaArchimax2D(n, generator=genLog(), depfun=dep1(),
  gpars=generator$parameters, dpars=depfun$parameters, pars=list(gpars,dpars))

eCopula(data, generator=genGumbel(), depfun=dep1(), copula=NULL,
 glimits=list(generator$lower,generator$upper), dlimits=list(depfun$lower,depfun$upper),
  limits=list(copula$lower,copula$upper),
  ggridparameters=if(!is.null(unlist(glimits))) do.call(
   function(...) mapply(c,...,length.out=pgrid,SIMPLIFY=FALSE), glimits) else NULL,
  dgridparameters=if(!is.null(unlist(dlimits))) do.call(
```

```
   function(...) mapply(c,...,length.out=pgrid,SIMPLIFY=FALSE), dlimits) else NULL,
  gridparameters=if(!is.null(unlist(limits))) do.call(function(...)
    mapply(c,...,length.out=pgrid,SIMPLIFY=FALSE), limits) else NULL,
  technique=c("ML","LS","icorr"), procedure=c("optim","nlminb","nls","grid"),
  method="default", corrtype = c("kendall","spearman"), control=NULL, pgrid=10)

gCopula(data, generator, depfun=dep1(), copula=NULL,
  glimits=list(generator$lower,generator$upper),
  dlimits=list(depfun$lower,depfun$upper),
  limits=list(copula$lower,copula$upper),
  etechnique=c("ML","LS","icorr"), eprocedure=c("optim","nlminb","nls"),
  emethod="default", ecorrtype=c("kendall","spearman"), econtrol=NULL,
  N=100, m=nrow(data), ncores=1)

gCopulaEmpirical(data,N=100,ncores=1)

isCopula(generator=genLog(),depfun=dep1(), copula=NULL,
  glimits=list(generator$lower,generator$upper),
  dlimits=list(depfun$lower,depfun$upper),
  limits=list(copula$lower,copula$upper),
  ggridparameters=if(!is.null(unlist(glimits))) do.call(
   function(...) mapply(c,...,length.out=pgrid,SIMPLIFY=FALSE), glimits) else NULL,
  dgridparameters=if(!is.null(unlist(dlimits))) do.call(
   function(...) mapply(c,...,length.out=pgrid,SIMPLIFY=FALSE), dlimits) else NULL,
  gridparameters=if(!is.null(unlist(limits))) do.call(
   function(...) mapply(c,...,length.out=pgrid,SIMPLIFY=FALSE), limits) else NULL,
  dagrid=10, pgrid=10, dim=3, tolerance=1e-15)


## S3 method for class 'eCopulaArchimax'
print(x,...)
## S3 method for class 'eCopulaGeneric'
print(x,...)
## S3 method for class 'gCopula'
print(x,...)
## S3 method for class 'isCopula'
print(x,...)
```

Arguments

data	vector, matrix or data frame with as many columns as variables. List of two such objects in case of gCopulaEmpirical.
generator	list containing archimedean generator, it's inverse, parameter bounds and possibly derivatives. See generator.
depfun	list containing Pickand's dependence function, parameter bounds and possibly derivatives. See depfun.
copula	list containing generic copula CDF and/or PDF, parameter values and bounds. See copula.

`pars` either numeric vector of generic copula parameters, or list of two vectors: generator parameters and dependence function parameters.

`gpars` numeric vector of `generator` parameters, `NULL` in trivial case.

`dpars` numeric vector of `depfun` parameters, `NULL` in trivial case.

`quantile` numeric. Index of the quantile variable among other variables; if `NULL` (default) then no quantile is computed.

`probability` numeric. Probability corresponding to the wanted quantile. Replicated to length of the data. By default `quantile`-th column of `data`.

`subdivisions` integer. Parameter passed to `nintegrate`.

`difference` parameter passed to `nderive`.

`base` data.frame or matrix. Data set from which an empirical copula is constructed.

`area` parameter passed to `nderive`.

`shrinkdiff` logical. Whether to shrink `difference` if [0,1] interval is exceeded. By default, `area` is changed instead.

`conditional.on` numeric. Index of variables to be conditioned on.

`n` number of random observations to be generated.

`dim` number of dimensions of copula = number of variables.

`glimits, dlimits, limits`
list of two vectors: lower and upper bound of `generator`, `depfun` and generic copula parameters, respectively.

`technique, etechnique`
copula parameters estimation method: Maximum pseudo-likelihood `"ML"`, Least square distance to empirical copula `"LS"` and Inversion of correlation coefficient relation to copula parameter `"icorr"`.

`procedure, eprocedure`
R optimization routine to estimate parameters (`"optim"`, `"nlminb"`, `"nls"`) or "brute force" search over parameter grid (`"grid"`). The last one is useful when the other methods give unsatisfactory results. `"nls"` cannot be used with ML technique. Ignored with "icorr" technique.

`method, emethod`
optimisation algorithm used by `optim` procedure.

`corrtype,ecorrtype`
character. Correlation coeficient used by `icorr` technique, either "kendall" and "spearman".

`control,econtrol`
list of control settings passed to optimization routines.

`ggridparameters, dgridparameters, gridparameters`
list of parameters values to create a grid from; any list item can be vector of elements named to match `seq()` arguments; by default the sequence is constructed from `glimits`,`dlimits`,`limits`, respectively, and `pgrid`.

`pgrid` number of grid points in each dimension of parameters space. Used when `gparameters, dparameters` are not supplied.

`N` number of bootstrap cycles.

m	number of Monte Carlo cyles needed in approximation of parametric function if there is no analytical expression available.
ncores	number of cores to be used for p-value simulation. Parallelization requires package parallel or multicore and may not work on Windows OS.
dagrid	integer. Number of data grid nodes used to check copula properties in.
tolerance	numeric. How much to tolerate departure of numeric results from theoretical values/limits.
x	an object used to select a method.
...	further arguments passed to or from other methods.

Value

Numeric vector, in case of p,d,c and qCopula.

Matrix from rCopula.

eCopula returns list containing

parameters	list of numeric vectors. Generator, depfu or generic copula parameters.
approach	character vector of length 3 describing estimation and optimization method.
fvalue	numeric. Value of the optimized function.
procedure.output	list. Full outcome of an optimization function.

gCopula and gCopulaEmpirical return list containing

statistic	numeric. GOF test statistic.
q95	numeric. 95% quantile or critical value from bootstrap simulations.
p.value	numeric. p-value from bootstrap simulations.
estimate	numeric vector. Copula parameters estimates.
data.name	character. Name of the supplied data object.
method	character. Identification of GoF test used.
fitting_method	character vector of length 3. Sumarizes estimation and optimiztion method.
copula_id	character. Generator and depfu id, or copula id.

isCopula returns list containing

is.copula	logical. FALSE if any of copula properties is violated.
issues	data frame. Consists of 1) index of variable, 2) violated property, 3) deviation from allowed range, 4) copula parameters for which the issue emerged.

Author(s)

Tomas Bacigal

References

Genest, C., Rémillard, B. and Beaudoin, D. (2009): Goodness-of-fit tests for copulas: A review and a power study. *Insurance: Mathematics and Economics* 44, 199–213.

Rémillard, B., Scaillet, O. (2009): Testing for equality between two copulas. *Journal of Multivariate Analysis* 100(3), 377–386.

See Also

generator, depfun, pCopula

Examples

```
## assign generator definition list with specific parameter
ge <- genGumbel(parameters=4)

## probability P(U<0.3,V<0.5)
pCopula(c(0.3,0.5),ge)  #0.2906142
## quantile q for which P(U<q,V<0.5)=0.2906142
pCopula(c(0.2906142,0.5),ge,quantile=1)  #0.3000175
pCopula(c(NA,0.5),ge,quantile=1,probability=0.2906142)
qCopula(c(0.5),quantile=1,probability=0.2906142,generator=ge)

## conditional probability P(U<0.3|V=0.5)
cCopula(c(0.3,0.5),ge,conditional.on=2)  #0.1025705
## quantile q for which conditional probability P(U<q|V=0.5)=0.1025705
cCopula(c(0.1025705,0.5),conditional.on=2,generator=ge,quantile=1)  #0.2999861
cCopula(c(NA,0.5),conditional.on=2,generator=ge,quantile=1,probability=0.1025705)
qCopula(c(0.5),quantile=1,probability=0.1025705,conditional.on=2,generator=ge)

## copula density
dCopula(c(0.3,0.5),ge) #1.083797
local({
x <- y <- seq(0,1,length.out=20)
persp(x,y,matrix(dCopula(expand.grid(x,y),ge),nrow=length(x)),r=2,zlab="density")
})

## simulate random vector
rge <- rCopula(100,dim=2,ge)
plot(rge)
# Observe that using rCopula(100,dim=2,cop=copGumbel(parameters=4))
# would take much more time to sample, since numerical derivative needs to be employed.

## --- fit copula to data set
# maximum likelihood (using density)
eCopula(rge,ge,technique="ML")
# some methods has no support for parameters bounds (do not mind a warning message)
eCopula(rge,ge,technique="ML",method="BFGS")
# least-square fit to empirical copula
eCopula(rge,ge,technique="LS",procedure="nlminb")
# maximizing discretized likelihood function
eCopula(rge,ge,technique="ML",procedure="grid",glimits=list(2.,6.),pgrid=20)
```

```
# specify nodes of the grid
eCopula(rge,ge,tech="ML",proc="grid",ggridparameters=list(c(2.,6.,length.out=20)))
# without naming, it won't create sequence
eCopula(rge,ge,technique="ML",procedure="grid",ggridparameters=list(c(2.,6.,20)))
# inversion of Kendall's tau
eCopula(rge,ge,technique="icorr",corrtype="kendall")

## --- GoF test, set substantially higher N to increase precision of p-value
gCopula(rge,ge,etechnique="ML",N=5)
# parallel computing takes lesser time, but the progress is not displayed
# not available on Windows OS
if(.Platform$OS.type!="windows") {
  gCopula(rge,ge,etechnique="ML",N=5,ncores=2)
}

## testing if two data sets has equal copulas
rge1 <- rCopula(80,dim=2,genClayton(),gpars=3)
gCopula(list(rge,rge1),N=5)

## check whether some hypotheticaly-copula function does not violate
## copula properties (over data and parameters grid)
isCopula(genGumbel(),dagrid=10,pgrid=10,tolerance=1e-15)

## all the above functions are ready for archimax or generic copulas
## as well as for higher dimensions
pCopula(c(0.3,0.5,1.0),genClayton(),depGumbel(),gpars=0.01,dpars=4.)  #0.2907613
pCopula(c(0.3,0.5,1.0),copula=copGumbel(),pars=4.)  #0.2906142
```

`xPareto` *4-parametric univariate Pareto distribution*

Description

Probability (cumulative) distribution function `pPareto` and quantile function `qPareto` of Pareto type IV distribution.

Usage

```
pPareto(t,pars)
qPareto(t,pars)
```

Arguments

`t`	numeric. Argument of the functions.
`pars`	numeric vector of length 4. Parameters of the Pareto distribution (see Details).

Details

Cumulative distribution function of 4-parametric Pareto is defined as

$$cdf(x) = 1 - \left(1 + \left(\frac{t - p_4}{/} p_1\right)^{1/p_3}\right)^{-p_2} \quad \text{for } t \geq p_4$$

and results to 0 otherwise.

Value

Numeric.

Author(s)

Tomas Bacigal

See Also

generator, depfun, copula

Examples

```
## probability P(X<q)=p
pPareto(t = 2.5, pars = c(10.,5.,3.,1))  # 0.8823436
qPareto(t = .Last.value, pars = c(10.,5.,3.,1))  # 2.5
```

Index

∗ **Archimax**
acopula-package, 2
∗ **Archimedean copula**
generator, 7
∗ **Archimedean**
acopula-package, 2
∗ **CDF**
xPareto, 16
∗ **EV copula**
depfun, 4
∗ **Extreme-Value copula**
depfun, 4
∗ **Extreme-Value**
acopula-package, 2
∗ **Pareto distribution**
xPareto, 16
∗ **Pickands' dependence function**
depfun, 4
∗ **conditional probability**
xCopula, 11
∗ **copula**
copula, 3
∗ **cumulative distribution function**
xCopula, 11
∗ **d-increasing**
xCopula, 11
∗ **derivative**
nderive, 8
∗ **empirical copula**
xCopula, 11
∗ **generator**
generator, 7
∗ **goodness-of-fit test**
xCopula, 11
∗ **integral**
nintegrate, 9
∗ **inverse of correlation coefficient**
xCopula, 11
∗ **linear approximation**
nderive, 8
∗ **maximum likelihood**
xCopula, 11
∗ **probability density function**
xCopula, 11
∗ **quantile**
xCopula, 11
xPareto, 16
∗ **sampling**
xCopula, 11
∗ **split vector**
vpartition, 10
∗ **trapezoid**
nintegrate, 9

acopula *(acopula-package)*, 2
acopula-package, 2

cCopula *(xCopula)*, 11
copFGM *(copula)*, 3
copGumbel *(copula)*, 3
copNormal *(copula)*, 3
copPlackett *(copula)*, 3
copProduct *(copula)*, 3
copula, 3, *6*, *8*, *12*, *17*

dCopula *(xCopula)*, 11
dep1 *(depfun)*, 4
depCC *(depfun)*, 4
depfun, *4*, 4, *8*, *12*, *15*, *17*
depGalambos *(depfun)*, 4
depGCC *(depfun)*, 4
depGumbel *(depfun)*, 4
depHuslerReiss *(depfun)*, 4
depMax *(depfun)*, 4
depTawn *(depfun)*, 4

eCopula *(xCopula)*, 11
eCopulaArchimax *(xCopula)*, 11
eCopulaGeneric *(xCopula)*, 11

gCopula *(xCopula)*, 11
gCopulaEmpirical *(xCopula)*, 11
genAMH *(generator)*, 7
genClayton *(generator)*, 7
generator, *4*, *6*, 7, *12*, *15*, *17*
genFrank *(generator)*, 7
genGumbel *(generator)*, 7
genJoe *(generator)*, 7
genLog *(generator)*, 7

isCopula *(xCopula)*, 11

ldepPartition3D *(depfun)*, 4

nderive, 8, *9*, *13*
nintegrate, *9*, 9, *13*

parallel, *14*
pCopula, *4*, *6*, *8*, *15*
pCopula *(xCopula)*, 11
pCopulaEmpirical *(xCopula)*, 11
pPareto *(xPareto)*, 16
print.eCopulaArchimax *(xCopula)*, 11
print.eCopulaGeneric *(xCopula)*, 11
print.gCopula *(xCopula)*, 11
print.isCopula *(xCopula)*, 11

qCopula *(xCopula)*, 11
qPareto *(xPareto)*, 16

rCopula *(xCopula)*, 11
rCopulaArchimax2D *(xCopula)*, 11

vpartition, 10

xCopula, 11
xPareto, 16