

Package ‘fCopulae’

October 28, 2009

Version 2110.78

Revision

Date 2009-10-27

Title Rmetrics - Dependence Structures with Copulas

Author Diethelm Wuertz and many others, see the SOURCE file

Depends R (>= 2.4.0), sn, methods, timeDate, timeSeries, fBasics (>= 2100.78)

Suggests RUnit, tcltk

Maintainer Rmetrics Core Team <Rmetrics-core@r-project.org>

Description Environment for teaching “Financial Engineering and Computational Finance”

NOTE SEVERAL PARTS ARE STILL PRELIMINARY AND MAY BE CHANGED IN THE FUTURE. THIS TYPICALLY INCLUDES FUNCTION AND ARGUMENT NAMES, AS WELL AS DEFAULTS FOR ARGUMENTS AND RETURN VALUES.

LazyLoad yes

LazyData yes

License GPL (>= 2)

URL <http://www.rmetrics.org>

Repository CRAN

Date/Publication 2009-10-28 07:56:25

R topics documented:

adapt	2
ArchimedeanCopulae	4
ArchimedeanDependency	6
ArchimedeanGenerator	8
ArchimedeanModelling	9
BivariateBinning	11
BivariateGridding	12
cauchy2d	13
CopulaeClass	14
density2d	16
elliptical2d	17
EllipticalCopulae	19
EllipticalDependency	22
EllipticalGenerator	25
EllipticalModelling	28
EmpiricalCopulae	30
ExtremeValueCopulae	32
ExtremeValueDependency	34
ExtremeValueGenerator	36
ExtremeValueModelling	37
MultivariateDistribution	39
norm2d	44
t2d	45
Index	47

adapt

*Adaptive Numerical Integration in 2–20 Dimensions***Description**

Integrates a scalar function over a multidimensional rectangle.

Usage

```
adapt(ndim, lower, upper, minpts = 100, maxpts = NULL, functn,
      eps = 0.01, ...)
```

Arguments

ndim	the dimension of the integral, and i.e. number
lower	vector of at least length ndim of the lower bounds on the integral.
upper	vector of at least length ndim of the upper bounds on the integral.
minpts	the minimum number of function evaluations.
maxpts	the maximum number of function evaluations or NULL per default, see <i>Details</i> .

<code>functn</code>	an R function which should take a single vector argument and possibly some parameters and return the function value at that point. <code>functn</code> must return a single numeric value.
<code>eps</code>	the desired accuracy for the relative error.
<code>...</code>	other parameters to be passed to <code>functn</code>

Details

The function computes

$$\int_l^u \text{functn}(t) d^n t$$

where $l = \text{lower}$, $u = \text{upper}$ and $n = \text{ndim}$. Infinite rectangles are not allowed, and `ndim` must be between 2 and 20.

This is modified from Mike Meyer's S code. The functions just call A.C. Genz's fortran ADAPT subroutine to do all of the calculations. A work array is allocated within the C/Fortran code.

The Fortran function has been modified to use double precision, for compatibility with R. It only works in two or more dimensions; for one-dimensional integrals use the [integrate](#) function in the base package.

Setting `maxpts` to NULL asks the function to keep doubling `maxpts` (starting at $\max(\text{minpts}, 500, r(\text{ndim}))$) until the desired precision is achieved or R runs out of memory. Note that the necessary number of evaluations typically grows exponentially with the dimension `ndim`, and the underlying code requires $\text{maxpts} \geq r(\text{ndim})$ where $r(d) = 2^d + 2d(d + 3) + 1$.

Value

A list of class "integration" with components

<code>value</code>	the estimated integral
<code>relerr</code>	the estimated relative error; $< \text{eps}$ argument if the algorithm converged properly.
<code>minpts</code>	the actual number of function evaluations
<code>ifail</code>	an error indicator. If <code>ifail</code> is not equal to 0, the function warns the user of the error condition.

See Also

[integrate](#)

Examples

```
## Example of p - dimensional spherical normal distribution:
ir2pi <- 1/sqrt(2*pi)
fred <- function(z) { ir2pi^length(z) * exp(-0.5 * sum(z * z)) }

adapt(2, lo = c(-5, -5), up = c(5, 5), functn = fred)
```

```

adapt(2, lo = c(-5, -5), up = c(5, 5), functn = fred, eps = 1e-4)
adapt(2, lo = c(-5, -5), up = c(5, 5), functn = fred, eps = 1e-6)

## adapt "sees" function ~= constantly 0 --> wrong result
adapt(2, lo = c(-9,-9), up = c(9,9), functn = fred)

## fix by using much finer initial grid:
adapt(2, lo = c(-9,-9), up = c(9,9), functn = fred, min = 1000)
adapt(2, lo = c(-9,-9), up = c(9,9), functn = fred, min = 1000, eps = 1e-6)
il <- print(integrate(dnorm, -2, 2))$value

## True values for the following example:
il ^ c(3, 5)
for(p in c(3, 5)) {
  cat("\np = ", p, "\n-----\n")
  f.lo <- rep(-2., p)
  f.up <- rep(+2., p)
  # not enough evaluations:
  print(adapt(p, lo=f.lo, up=f.up, max=100*p, functn = fred))
  # enough evaluations:
  print(adapt(p, lo=f.lo, up=f.up, max=10^p, functn = fred))
  # no upper limit; p=3: 7465 points, ie 5 attempts (on an Athlon/gcc/g77):
  print(adapt(p, lo=f.lo, up=f.up, functn = fred, eps = 1e-5))
}

```

ArchimedeanCopulae *Bivariate Archimedean Copulae*

Description

A collection and description of functions to investigate bivariate Archimedean copulae.

Archimedean Copulae Functions:

rarchmCopula	Generates Archimedean copula variates,
parchmCopula	computes Archimedean copula probability,
darchmCopula	computes Archimedean copula density,
rarchmSlider	displays interactive plots of variates,
parchmSlider	displays interactive plots of probability,
darchmSlider	displays interactive plots of density.

Special Copulae Functions:

rgumbelCopula	Generates Gumbel copula variates,
pgumbelCopula	computes Gumbel copula probability,
dgumbelCopula	computes Gumbel copula density.

Usage

```

rarchmCopula(n, alpha = NULL, type = archmList())
parchmCopula(u = 0.5, v = u, alpha = NULL, type = archmList(), output =
  c("vector", "list"), alternative = FALSE)
darchmCopula(u = 0.5, v = u, alpha = NULL, type = archmList(), output =
  c("vector", "list"), alternative = FALSE)

rarchmSlider(B = 10)
parchmSlider(type = c("persp", "contour"), B = 10)
darchmSlider(type = c("persp", "contour"), B = 10)

rgumbelCopula(n, alpha = 2)
pgumbelCopula(u = 0.5, v = u, alpha = 2, output = c("vector", "list"))
dgumbelCopula(u = 0.5, v = u, alpha = 2, output = c("vector", "list"))

```

Arguments

alpha	[Phi*][*archmCopula] - the parameter of the Archimedean copula. A numerical value.
alternative	[*Copula] - Should the probability be computed alternatively ...
B	[*Slider] - the maximum slider menu value when the boundary value is infinite. By default this is set to 10.
n	[rarchmCopula] - the number of random deviates to be generated, an integer value.
output	[*archmCopula] - output - a character string specifying how the output should be formatted. By default a vector of the same length as u and v. If specified as "list" then u and v are expected to span a two-dimensional grid as outputted by the function <code>grid2d</code> and the function returns a list with elements \$x, \$y, and \$z which can be directly used for example by 2D plotting functions.
type	[*archmCopula] - the type of the Archimedean copula. A character string ranging between "1" and "22". By default copula No. 1 will be chosen. [*archmSlider] - the type of the plot. A character string either specifying a perspective or contour plot.
u, v	[*archmCopula] - two numeric values or vectors of the same length at which the copula will be computed. If u is a list then the \$x and \$y elements will be used as u and v. If u is a two column matrix then the first column will be used as u and the second as v.

Value

The function `pcopula` returns a numeric matrix of probabilities computed at grid positions `xly`.

The function `parchmCopula` returns a numeric matrix with values computed for the Archimedean copula.

The function `darchmCopula` returns a numeric matrix with values computed for the density of the Archimedean copula.

The functions `Phi*` return a numeric vector with the values computed from the Archimedean generator, its derivatives, or its inverse.

The functions `cK` and `cKInv` return a numeric vector with the values of the density and inverse for Archimedean copulae.

Author(s)

Diethelm Wuertz for the Rmetrics R-port.

ArchimedeanDependency

Bivariate Archimedean Copulae

Description

A collection and description of functions to investigate bivariate Archimedean copulae.

Archimedean Copulae Functions:

<code>archmTau</code>	Computes Kendall's tau for Archimedean copulae,
<code>archmRho</code>	computes Spearman's rho for Archimedean copulae,
<code>archmTailCoeff</code>	computes tail dependence for Archimedean copulae,
<code>archmTailPlot</code>	plots tail dependence for Archimedean copulae.

Usage

```
archmTau(alpha = NULL, type = archmList(), lower = 1.0e-10)
archmRho(alpha = NULL, type = archmList(), method = c("integrate2d", "adapt"),
  error = 1.0e-5)
```

```
archmTailCoeff(alpha = NULL, type = archmList())
archmTailPlot(alpha = NULL, type = archmList(), tail = c("Upper", "Lower"))
```

Arguments

alpha	the parameter of the Archimedean copula. A numerical value.
error	[archmRho] - the error bound to be achieved by the <code>integrate2d</code> integration formula. A numeric value, by default <code>error=1.0e-5</code> .
lower	[archmTau] - a numeric value setting the lower bound for the internal integration function <code>integrate</code> .
tail	[archmTailPlot] - a character string, either "Upper" or "Lower" denoting which of the two tails should be displayed. By default the upper tail dependence will be considered.
type	the type of the Archimedean copula. A character string ranging between "1" and "22". By default copula No. 1 will be chosen.
method	[archmRho] - a character string that determines which integration method should be used, either "integrate2d" or "adapt". If the second method is selected the contributed R package "adapt" is required.

Value

The function `pcopula` returns a numeric matrix of probabilities computed at grid positions $x|y$.

The function `parchmCopula` returns a numeric matrix with values computed for the Archimedean copula.

The function `darchmCopula` returns a numeric matrix with values computed for the density of the Archimedean copula.

The functions `Phi*` return a numeric vector with the values computed from the Archimedean generator, its derivatives, or its inverse.

The functions `cK` and `cKInv` return a numeric vector with the values of the density and inverse for Archimedean copulae.

Author(s)

Diethelm Wuertz for the Rmetrics R-port.

 ArchimedeanGenerator

Bivariate Archimedean Copulae

Description

A collection and description of functions concerned with the generator function for the Archimedean copula and with functions for setting and checking the distributional parameters.

Functions:

<code>evList</code>	Returns list of implemented Archimedean copulae,
<code>archmParam</code>	Sets default parameters for an Archimedean copula,
<code>archmRange</code>	returns the range of valid rho values,
<code>archmCheck</code>	checks if rho is in the valid range,
<code>Phi</code>	Computes generator Phi, inverse and derivatives,
<code>PhiSlider</code>	displays interactively generator function,
<code>Kfunc</code>	computes copula density and its inverse,
<code>KfuncSlider</code>	displays interactively density function.

Usage

```

archmList()
archmParam(type = archmList())
archmRange(type = archmList(), B = Inf)
archmCheck(alpha, type = archmList())

Phi(x, alpha = NULL, type = archmList(), inv = FALSE, deriv = paste(0:2))
PhiSlider(B = 5)

Kfunc(x, alpha = NULL, type = archmList(), inv = FALSE, lower = 1.0e-8)
KfuncSlider(B = 5)

```

Arguments

<code>alpha</code>	<code>[Phi*][*archmCopula]</code> - the parameter of the Archimedean copula. A numerical value.
<code>B</code>	<code>[archmRange]</code> - the maximum slider menu value when the boundary value is infinite. By default this is set to <code>B=Inf</code> . <code>[*Slider]</code> - the maximum slider menu value when the boundary value is infinite. By default this is set to <code>B=5</code> .
<code>deriv</code>	<code>[Phi]</code> - an integer value. Should the function itself, <code>deriv="0"</code> , or the first <code>deriv="1"</code> , or second <code>deriv="2"</code> derivative be evaluated?

<code>inv</code>	<code>[Phi][Kfunc]</code> - a logical flag. Should the inverse function be computed?
<code>lower</code>	<code>[Kfunc]</code> - a numeric value setting the lower bound for the internal root finding function <code>uniroot</code> .
<code>type</code>	<code>[*archmCopula][Phi][Kfunc]</code> - the type of the Archimedean copula. A character string ranging between "1" and "22". By default copula No. 1 will be chosen.
<code>x</code>	<code>[Kfunc]</code> - a numeric value or vector ranging between zero and one. <code>[Phi]</code> - a numeric value or vector.

Value

The function `Phi` returns a numeric vector with the values computed from the Archimedean generator, its derivatives, or its inverse.

The function `Kfunc` returns a numeric vector with the values of the density and inverse for Archimedean copulae.

Author(s)

Diethelm Wuertz for the Rmetrics R-port.

References

RB Nelson - An Introduction to Copulas

Examples

```
## archmList -
# Return list of implemented copulae:
archmList()
```

ArchimedeanModelling

Bivariate Archimedean Copulae

Description

A collection and description of functions to investigate bivariate Archimedean copulae.

Archimedean Copulae Functions:

<code>archmCopulaSim</code>	simulates an Archimedean copula,
<code>archmCopulaFit</code>	fits the parameters of an Archimedean copula.

Usage

```
archmCopulaSim(n, alpha = NULL, type = archmList())
archmCopulaFit(u, v = NULL, type = archmList(), ...)
```

Arguments

alpha	[Phi*][*archmCopula] - the parameter of the Archimedean copula. A numerical value.
n	[rarchmCopula] - the number of random deviates to be generated, an integer value.
type	the type of the Archimedean copula. A character string ranging between "1" and "22". By default copula No. 1 will be chosen.
u, v	[*archmCopula] - two numeric values or vectors of the same length at which the copula will be computed. If u is a list then the the \$x and \$y elements will be used as u and v. If u is a two column matrix then the first column will be used as u and the the second as v.
...	[archmCopulaFit] - arguments passed to the optimization function in use, nlmminb.

Value

The function `pcopula` returns a numeric matrix of probabilities computed at grid positions `xly`.

The function `parchmCopula` returns a numeric matrix with values computed for the Archimedean copula.

The function `darchmCopula` returns a numeric matrix with values computed for the density of the Archimedean copula.

The functions `Phi*` return a numeric vector with the values computed from the Archimedean generator, its derivatives, or its inverse.

The functions `cK` and `cKInv` return a numeric vector with the values of the density and inverse for Archimedean copulae.

Author(s)

Diethelm Wuertz for the Rmetrics R-port.

 BivariateBinning *Square and Hexagonal Data Binning*

Description

A collection and description of functions which allow to create histograms due to square and hexagonal binning.

Bivariate Binning Functions:

squareBinning	does a square binning of data points,
hexBinning	does a hexagonal binning of data points

Usage

```
squareBinning(x, y = NULL, bins = 30)
hexBinning(x, y = NULL, bins = 30)

## S3 method for class 'squareBinning':
plot(x, col = heat.colors(12), addPoints = TRUE,
     addRug = TRUE, ...)
## S3 method for class 'hexBinning':
plot(x, col = heat.colors(12), addPoints = TRUE,
     addRug = TRUE, ...)
```

Arguments

addPoints	a logical flag, should the center of mass points added to the plot?
addRug	a logical flag, should a rug representation be added to the plot, for details see the function rug.
bins	an integer specifying the number of bins.
col	color map like for the image function.
x, y	[squareBinning][hexBinning] - either two numeric vectors of equal length or if y is NULL, a list with entries x, y, or named data frame with x in the first and y in the second column. Note, timeSeries objects are also allowed as input.
	[plot] - an object of class squareBinning or hexBinning.
...	arguments to be passed.

Value

A list with three entries, x, y and z, specified by an object of class squareBinning or hexBinning. Note, the returned value, can be directly used by the persp() and contour 3D plotting functions.

Author(s)

Diethelm Wuertz for the Rmetrics R-port.

Examples

```
## squareBinning -
sB = squareBinning(x = rnorm(1000), y = rnorm(1000))
plot(sB)

## hexBinning -
hB = hexBinning(x = rnorm(1000), y = rnorm(1000))
plot(hB)
```

BivariateGridding *Bivariate Gridded Data Sets*

Description

A collection and description of functions which allow to generate bivariate gridded data sets.

Grid Data Functions:

gridData	generates a grid data set of class 'gridData',
persp	generates a perspective plot from a grid data set,
contour	generates a contour plot from a grid data set.

Usage

```
gridData(x = (-10:10)/10, y = x, z = outer(x, y, function(x, y) (x^2+y^2) ) )

## S3 method for class 'gridData':
persp(x, theta = -40, phi = 30, col = "steelblue",
      ticktype = "detailed", ...)
## S3 method for class 'gridData':
contour(x, addImage = TRUE, ...)
```

Arguments

addImage	[contour] - a logical flag indicating if an image plot should be underlayed to the contour level plot.
x, y, z	[gridData] - x and y are two numeric vectors of grid points and z is a numeric matrix or any other rectangular object which can be transformed by the function <code>as.matrix</code> into a matrix object.

```
theta, phi, col, ticktype
      [persp] -
      tailored parameters passed the perspective plot function persp.
...
      [contour][persp] -
      additional arguments to be passed to the perspective and contour plot functions.
```

Value

gridData -
A list with at least three entries, x, y and z.
The returned values, can be directly used by the `persp.gridData()` and `contour.gridData` 3D plotting methods.

Author(s)

Diethelm Wuertz for the Rmetrics R-port,
H. Akima for the Fortran Code of the Akima spline interpolation routine.

Examples

```
## gridData -
# Grid Data Set
gD = gridData()
persp(gD)
contour(gD)
```

cauchy2d

Bivariate Cauchy Distribution

Description

Density, distribution function, and random generation for the bivariate Cauchy distribution.

Usage

```
pcauchy2d(x, y = x, rho = 0)
dcauchy2d(x, y = x, rho = 0)
rcauchy2d(n, rho = 0)
```

Arguments

n the number of random deviates to be generated, an integer value.
rho the correlation parameter, a numeric value ranging between minus one and one,
 by default zero.
x, y two numeric vectors defining the x and y coordinates.

Value

pcauchy2d

returns a two column matrix of probabilities for the bivariate Cauchy distribution function.

dcauchy2d

returns a two column matrix of densities for the bivariate Cauchy distribution function.

rcauchy2d

returns a two column matrix of random deviates generated from the bivariate Cauchy distribution function.

Author(s)

Adelchi Azzalini for the underlying pnorm2d function,
Diethelm Wuertz for the Rmetrics R-port.

References

Azzalini A., (2004); *The sn Package*; R Reference Guide available from www.r-project.org.
Venables W.N., Ripley B.D., (2002); *Modern Applied Statistics with S*, Fourth Edition, Springer.

Examples

```
## Bivariate Cauchy Density:
x = (-40:40)/10
X = grid2d(x)
z = dcauchy2d(X$x, X$y, rho = 0.5)
Z = list(x = x, y = x, z = matrix(z, ncol = length(x)))
persp(Z, theta = -40, phi = 30, col = "steelblue")
```

CopulaeClass

Bivariate Copulae Class

Description

A collection and description of functions to specify the copula class and to investigate bivariate Frechet copulae.

The class representation and methods are:

fcOPULA representation for an S4 object of class "fcOPULA",
show S4 print method.

Frechet Copulae:

pfrechetCopula computes Frechet copula probability.

Usage

```
## S4 method for signature 'fCOPULA':
show(object)

pfrechetCopula(u = 0.5, v = u, type = c("m", "pi", "w"),
               output = c("vector", "list"))
```

Arguments

object	[show] - an S4 object of class "fCOPULA".
output	[*frechetCopula] - output - a character string specifying how the output should be formatted. By default a vector of the same length as <i>u</i> and <i>v</i> . If specified as "list" then <i>u</i> and <i>v</i> are expected to span a two-dimensional grid as outputted by the function <code>grid2d</code> and the function returns a list with elements <i>x</i> , <i>y</i> , and <i>z</i> which can be directly used for example by 2D plotting functions.
type	[*frechetCopula] - the type of the Frechet copula. A character string selected from: "m", "pi", or "w".
<i>u</i> , <i>v</i>	two numeric values or vectors of the same length at which the copula will be computed. If <i>u</i> is a list then the <i>x</i> and <i>y</i> elements will be used as <i>u</i> and <i>v</i> . If <i>u</i> is a two column matrix then the first column will be used as <i>u</i> and the second as <i>v</i> .

Details

The function `pfrechetCopula` returns a numeric matrix of probabilities computed at grid positions *u**v*. The arguments *u* and *v* are two single values or two numeric vectors of the same length. If *v* is not specified then the same values are taken as for *u*. Alternatively, *u* may be given as a two column vector or as a list with two entries as vectors. The first column or entry is taken as *u* and the second as *v*.

Value

The print method `show` returns an S4 object of class "fCOPULA". The object contains the following slots:

@call	the function call.
@copula	the name of the copula.
@param	a list whose elements specify the model parameters of the copula.
@title	a character string with the name of the copula. This can be overwritten specifying a user defined input argument.
@description	a character string with an optional user defined description. By default just the current date will be returned.

The function `pfrechetCopula` returns a numeric vector of probabilities. An attribute named "control" is added which returns the name of the Frechet copula.

Author(s)

Diethelm Wuertz for the Rmetrics R-port.

Examples

```
## fCOPULA -
  getClass("fCOPULA")

## pcopula -
# The Frechet Copula - m:
pfrechetCopula(0.5)
pfrechetCopula(0.25, 0.75)
pfrechetCopula(runif(5))

grid2d()
pfrechetCopula(grid2d())
```

density2d

Bivariate Density Tools

Description

Grid generator, kernel density estimator, histogram counter, and integrator for bivariate distributions

Usage

```
grid2d(x = (0:10)/10, y = x)
density2d(x, y = NULL, n = 20, h = NULL, limits = c(range(x), range(y)))
hist2d(x, y = NULL, n = c(20, 20))
integrate2d(fun, error = 1.0e-5, ...)
```

Arguments

<code>error</code>	the error bound to be achieved by the integration formula. A numeric value.
<code>fun</code>	the function to be integrated. The first argument requests the x values, the second the y values, and the remaining are reserved for additional parameters. The integration is over the unit square "[0,1] ² ".
<code>h</code>	a vector of bandwidths for x and y directions. Defaults to normal reference bandwidth.
<code>limits</code>	the limits of the rectangle covered by the grid.

<code>n</code>	<code>n</code> - an integer specifying the number of grid points in each direction. The default value is 20. [hist2D] - In this case <code>n</code> may be a scalar or a two element vector. The default value is 20. [rnorm2d] - the number of random deviates to be generated, an integer value.
<code>x, y</code>	two numeric vectors defining the <code>x</code> and <code>y</code> coordinates. [density2D][hist2D] - two vectors of coordinates of data. If <code>y</code> is NULL then <code>x</code> is assumed to be a two column matrix, where the first column contains the <code>x</code> data, and the second column the <code>y</code> data.
<code>...</code>	parameters passed to the function to be integrated.

Value

`grid2d`
returns a list with two vectors named `$x` and `$y` spanning the grid defined by the coordinate vectors `x` and `y`.

`density2d`
`hist2d`
returns a list with three elements `$x`, `$y`, and `$z`. `x` and `y` are vectors spanning the two dimensional grid and `z` the corresponding matrix. The output can directly serve as input to the plotting functions `image`, `contour` and `persp`.

`integrate2d`
returns a list with the `$value` of the integral over the unit square $[0,1]^2$, an `$error` estimate and the number of grid `$points` used by the integration function.

Author(s)

W.N. Venables and B.D. Ripley for the underlying `kde2d` function,
Gregory R. Warnes for the underlying `hist2d` function,
Diethelm Wuertz for the Rmetrics R-port.

References

Azzalini A., (2004); *The sn Package*; R Reference Guide available from www.r-project.org.
Venables W.N., Ripley B.D., (2002); *Modern Applied Statistics with S*, Fourth Edition, Springer.
Warnes G.R., (2004); *The gregmisc Package*; R Reference Guide available from www.r-project.org.

Description

Density function for bivariate elliptical distributions.

Usage

```
delliptical2d(x, y = x, rho = 0, param = NULL, type = c("norm", "cauchy", "t",
  "logistic", "laplace", "kotz", "epower"), output = c("vector", "list"))
```

Arguments

output	output - a character string specifying how the output should be formatted. By default a vector of the same length as u and v . If specified as "list" then u and v are expected to span a two-dimensional grid as outputted by the function <code>grid2d</code> and the function returns a list with elements x , y , and z which can be directly used for example by 2D plotting functions.
param	additional parameters to specify the bivariate density function. Only effective for the Kotz and Exponential Power distribution. For the Kotz distribution we can specify a numeric value, by default defined as <code>param=c(r=sqrt(2))</code> , and for the Exponential Power distribution a numeric vector, by default defined as <code>param=c(r=sqrt(2), s=1/2)</code> .
rho	the correlation parameter, a numeric value ranging between minus one and one, by default zero.
type	the type of the elliptical copula. A character string selected from: "norm", "cauchy", "t", "laplace", "kotz", or "epower".
x, y	two numeric vectors defining the x and y coordinates.

Value

`delliptical2d`
returns a two column matrix of densities for the selected bivariate elliptical distribution function.

Author(s)

Diethelm Wuertz for the Rmetrics R-port.

References

Azzalini A., (2004); *The sn Package*; R Reference Guide available from www.r-project.org.
Venables W.N., Ripley B.D., (2002); *Modern Applied Statistics with S*, Fourth Edition, Springer.

Examples

```
## Kotz' Elliptical Density:
x = (-40:40)/10
X = grid2d(x)
z = delliptical2d(X$x, X$y, rho = 0.5, type = "kotz")
Z = list(x = x, y = x, z = matrix(z, ncol = length(x)))
persp(Z, theta = -40, phi = 30, col = "steelblue")
```

Description

A collection and description of functions to investigate bivariate elliptical copulae.

Elliptical Copulae Functions:

<code>rellipticalCopula</code>	Generates elliptical copula variates,
<code>pellipticalCopula</code>	computes elliptical copula probability,
<code>dellipticalCopula</code>	computes elliptical copula density,
<code>rellipticalSlider</code>	displays interactive plots of variates,
<code>pellipticalSlider</code>	displays interactive plots of probability,
<code>dellipticalSlider</code>	displays interactive plots of density.

Usage

```
rellipticalCopula(n, rho = 0.75, param = NULL, type = c("norm", "cauchy",
  "t"))
pellipticalCopula(u = 0.5, v = u, rho = 0.75, param = NULL,
  type = ellipticalList(), output = c("vector", "list"), border = TRUE)
dellipticalCopula(u = 0.5, v = u, rho = 0.75, param = NULL,
  type = ellipticalList(), output = c("vector", "list"), border = TRUE)

rellipticalSlider(B = 100)
pellipticalSlider(type = c("persp", "contour"), B = 20)
dellipticalSlider(type = c("persp", "contour"), B = 20)
```

Arguments

<code>B</code>	<code>[*Slider]</code> - the maximum slider menu value when the boundary value is infinite. By default this is set to 10.
<code>border</code>	<code>[pellipticalCopula][dellipticalCopula]</code> - a logical flag. If the argument <code>u</code> is an integer, say <code>N</code> , greater than one than all points on a square grid $[(0:N)/N]^2$ are computed. If <code>border</code> is <code>FALSE</code> then the border points are removed from the returned value, by default this is not the case.
<code>n</code>	<code>[rellipticalCopula][ellipticalCopulaSim]</code> - the number of random deviates to be generated, an integer value.
<code>output</code>	<code>[pellipticalCopula][dellipticalCopula]</code> - a character string specifying how the output should be formatted. By default a vector of the same length as <code>u</code> and <code>v</code> is returned. If specified as <code>"list"</code> then <code>u</code> and <code>v</code> are expected to span a two-dimensional grid as outputted by the function <code>grid2d</code> and the function returns a list with elements <code>\$x</code> , <code>y</code> , and <code>z</code> which can be

directly used for example by 2D plotting functions. For the grid version, when `u` is specified as an integer greater than one, always the output in form of a list will be returned.

<code>rho</code>	<code>[*ellipticalCopula]</code> - is the numeric value setting the correlation strength, ranging between minus one and one.
<code>param</code>	<code>[*ellipticalCopula][gfunc]</code> - additional distributional parameters: for the Student-t distribution this is "nu", for the Kotz distribution this is "r", and for the Exponential Power distribution these are "r" and "s". If the argument <code>param=NULL</code> then default values are taken. These are for the Student-t <code>param=c(nu=4)</code> , for the Kotz distribution <code>param=c(r=1)</code> , and for the exponential power distribution <code>param=c(r=1, s=1)</code> . Note, that the Kotz and exponential power copulae are independent of <code>r</code> , and that <code>r</code> only enters the generator, the density, the probability and the quantile functions.
<code>type</code>	<code>[*ellipticalCopula][gfunc]</code> - the type of the elliptical copula. A character string selected from: "norm", "cauchy", "t", "logistic", "laplace", "kotz", or "epower". <code>[*ellipticalSlider]</code> - a character string which indicates what kind of plot should be displayed, either a perspective plot if <code>type="persp"</code> , the default value, or a contour plot if <code>type="contour"</code> .
<code>u, v</code>	<code>[*ellipticalCopula]</code> - two numeric values or vectors of the same length at which the copula will be computed. If <code>u</code> is a list then the <code>\$x</code> and <code>\$y</code> elements will be used as <code>u</code> and <code>v</code> . If <code>u</code> is a two column matrix then the first column will be used as <code>u</code> and the second as <code>v</code> . If <code>u</code> is an integer value greater than one, say <code>N</code> , then the values for all points on the $[(0:N)/N]^2$ grid spanning the unit square will be returned.
<code>...</code>	<code>[ellipticalCopulaFit]</code> - arguments passed to the optimization function <code>nlm</code> .

Value

Copula Functions:

The functions `[rpd]ellipticalCopula` return a numeric vector of random variates, probabilities, or densities for the specified copula computed at grid coordinates `ulv`.

The functions `[rpd]ellipticalSlider` display an interactive graph of an perspective copula plot either for random variates, probabilities or densities. Alternatively, an image underlayed contour plot can be shown.

Copula Dependence Measures:

The functions `ellipticalTau` and `ellipticalRho` return a numeric value for Kendall's Tau and Spearman's Rho.

Copula Tail Coefficient:

The function `ellipticalTailCoeff` returns the coefficient of tail dependence for a specified copula. The function `ellipticalTailPlot` displays a whole plot for the upper or alternatively for the lower tail dependence as a function of u for a set of nine `rho` values.

Copula Generator Function:

The function `gfunc` computes the generator function for the specified copula, by default the normal copula. If the argument `x` is missing, then the normalization constant `lambda` will be returned, otherwise if `x` is specified the values for the function $g(x)$ will be returned. The selected type of copula is added to the output as an attribute named `"control"`. The function `gfuncSlider` allows to display interactively the generator function, the marginal density, the marginal probability, and the contours of the bivariate density.

Copula Simulation and Parameter Fitting:

The function `ellipticalCopulaSim` returns a numeric two-column matrix with randomly generated variates for the specified copula.

The function `ellipticalCopulaFit` returns a fit to empirical data for the specified copula. The returned object is a list with elements from the function `nlminb`.

Author(s)

Diethelm Wuertz for the Rmetrics R-port.

Examples

```
## [rp]ellipticalCopula -
# Default Normal Copula:
rellipticalCopula(10)
pellipticalCopula(10)

## [rp]ellipticalCopula -
# Student-t Copula Probability and Density:
u = grid2d(x = (0:25)/25)
pellipticalCopula(u, rho = 0.75, param = 4,
  type = "t", output = "list")
d = dellipticalCopula(u, rho = 0.75, param = 4,
  type = "t", output = "list")
persp(d, theta = -40, phi = 30, col = "steelblue")
```

```
## ellipticalTau -
## ellipticalRho -
  # Dependence Measures:
  ellipticalTau(rho = -0.5)
  ellipticalRho(rho = 0.75, type = "logistic", subdivisions = 100)

## ellipticalTailCoeff -
  # Student-t Tail Coefficient:
  ellipticalTailCoeff(rho = 0.25, param = 3, type = "t")

## gfunc -
  # Generator Function:
  plot(gfunc(x = 0:10), main = "Generator Function")

## ellipticalCopulaSim -
## ellipticalCopulaSim -
  # Simualtion and Parameter Fitting:
  rv = ellipticalCopulaSim(n = 100, rho = 0.75)
  ellipticalCopulaFit(rv)
```

EllipticalDependency

Bivariate Elliptical Copulae

Description

A collection and description of functions to investigate bivariate elliptical copulae.

Elliptical Copulae Functions:

ellipticalTau	Computes Kendall's tau for elliptical copulae,
ellipticalRho	computes Spearman's rho for elliptical copulae,
ellipticalTailCoeff	computes tail dependence for elliptical copulae,
ellipticalTailPlot	plots tail dependence for elliptical copulae.

Usage

```
ellipticalTau(rho)
ellipticalRho(rho, param = NULL, type = ellipticalList(), subdivisions = 500)

ellipticalTailCoeff(rho, param = NULL, type = c("norm", "cauchy", "t"))
ellipticalTailPlot(param = NULL, type = c("norm", "cauchy", "t"),
  tail = c("Lower", "Upper"))
```

Arguments

<code>rho</code>	<code>[*ellipticalCopula]</code> - is the numeric value setting the correlation strength, ranging between minus one and one.
<code>param</code>	<code>[*ellipticalCopula][gfunc]</code> - additional distributional parameters: for the Student-t distribution this is "nu", for the Kotz distribution this is "r", and for the Exponential Power distribution these are "r" and "s". If the argument <code>param=NULL</code> then default values are taken. These are for the Student-t <code>param=c(nu=4)</code> , for the Kotz distribution <code>param=c(r=1)</code> , and for the exponential power distribution <code>param=c(r=1, s=1)</code> . Note, that the Kotz and exponential power copulae are independent of r, and that r only enters the generator, the density, the probability and the quantile functions.
<code>subdivisions</code>	<code>[ellipticalRho]</code> - an integer value with the number of subdivisions in each direction on the two dimensional unit square to compute the mean value of Spearman's Rho. By default 500 subdivisions are used.
<code>tail</code>	<code>[ellipticalTailPlot]</code> - a character string, either "Upper" or "Lower" denoting which of the two tails should be displayed. By default the upper tail dependence will be considered.
<code>type</code>	<code>[*ellipticalCopula][gfunc]</code> - the type of the elliptical copula. A character string selected from: "norm", "cauchy", "t", "logistic", "laplace", "kotz", or "epower". <code>[*ellipticalSlider]</code> - a character string which indicates what kind of plot should be displayed, either a perspective plot if <code>type="persp"</code> , the default value, or a contour plot if <code>type="contour"</code> .
<code>...</code>	<code>[ellipticalCopulaFit]</code> - arguments passed to the optimization function <code>nlm</code> .

Value**Copula Functions:**

The functions `[rpd]ellipticalCopula` return a numeric vector of random variates, probabilities, or densities for the specified copula computed at grid coordinates `ulv`.

The functions `[rpd]ellipticalSlider` display an interactive graph of an perspective copula plot either for random variates, probabilities or densities. Alternatively, an image underlayed contour plot can be shown.

Copula Dependence Measures:

The functions `ellipticalTau` and `ellipticalRho` return a numeric value for Kendall's Tau and Spearman's Rho.

Copula Tail Coefficient:

The function `ellipticalTailCoeff` returns the coefficient of tail dependence for a specified copula. The function `ellipticalTailPlot` displays a whole plot for the upper or alternatively for the lower tail dependence as a function of u for a set of nine `rho` values.

Copula Generator Function:

The function `gfunc` computes the generator function for the specified copula, by default the normal copula. If the argument `x` is missing, then the normalization constant `lambda` will be returned, otherwise if `x` is specified the values for the function $g(x)$ will be returned. The selected type of copula is added to the output as an attribute named `"control"`. The function `gfuncSlider` allows to display interactively the generator function, the marginal density, the marginal probability, and the contours of the bivariate density.

Copula Simulation and Parameter Fitting:

The function `ellipticalCopulaSim` returns a numeric two-column matrix with randomly generated variates for the specified copula.

The function `ellipticalCopulaFit` returns a fit to empirical data for the specified copula. The returned object is a list with elements from the function `nlminb`.

Author(s)

Diethelm Wuertz for the Rmetrics R-port.

Examples

```
## [rp]ellipticalCopula -
# Default Normal Copula:
rellipticalCopula(10)
pellipticalCopula(10)

## [rp]ellipticalCopula -
# Student-t Copula Probability and Density:
u = grid2d(x = (0:25)/25)
pellipticalCopula(u, rho = 0.75, param = 4,
  type = "t", output = "list")
d = dellipticalCopula(u, rho = 0.75, param = 4,
  type = "t", output = "list")
persp(d, theta = -40, phi = 30, col = "steelblue")
```

```
## ellipticalTau -
## ellipticalRho -
  # Dependence Measures:
  ellipticalTau(rho = -0.5)
  ellipticalRho(rho = 0.75, type = "logistic", subdivisions = 100)

## ellipticalTailCoeff -
  # Student-t Tail Coefficient:
  ellipticalTailCoeff(rho = 0.25, param = 3, type = "t")

## gfunc -
  # Generator Function:
  plot(gfunc(x = 0:10), main = "Generator Function")

## ellipticalCopulaSim -
## ellipticalCopulaSim -
  # Simualtion and Parameter Fitting:
  rv = ellipticalCopulaSim(n = 100, rho = 0.75)
  ellipticalCopulaFit(rv)
```

EllipticalGenerator

Bivariate Elliptical Copulae

Description

A collection and description of functions concerned with the generator function for the elliptical copula and with functions for setting and checking the distributional parameters.

Functions:

ellipticalList	Returns list of implemented elliptical copulae,
ellipticalParam	Sets default parameters for an elliptical copula,
ellipticalRange	returns the range of valid rho values,
ellipticalCheck	checks if rho is in the valid range,
gfunc	Generator function for elliptical distributions,
gfuncSlider	Slider for generator, density and probability.

Usage

```
ellipticalList()
ellipticalParam(type = ellipticalList())
ellipticalRange(type = ellipticalList())
ellipticalCheck(rho = 0.75, param = NULL, type = ellipticalList())

gfunc(x, param = NULL, type = ellipticalList())
gfuncSlider(B = 10)
```

Arguments

B	[*Slider] - the maximum slider menu value when the boundary value is infinite. By default this is set to 10.
rho	[*ellipticalCopula] - is the numeric value setting the correlation strength, ranging between minus one and one.
param	[*ellipticalCopula][gfunc] - additional distributional parameters: for the Student-t distribution this is "nu", for the Kotz distribution this is "r", and for the Exponential Power distribution these are "r" and "s". If the argument <code>param=NULL</code> then default values are taken. These are for the Student-t <code>param=c(nu=4)</code> , for the Kotz distribution <code>param=c(r=1)</code> , and for the exponential power distribution <code>param=c(r=1, s=1)</code> . Note, that the Kotz and exponential power copulae are independent of r , and that r only enters the generator, the density, the probability and the quantile functions.
type	[*ellipticalCopula][gfunc] - the type of the elliptical copula. A character string selected from: "norm", "cauchy", "t", "logistic", "laplace", "kotz", or "epower". [*ellipticalSlider] - a character string which indicates what kind of plot should be displayed, either a perspective plot if <code>type="persp"</code> , the default value, or a contour plot if <code>type="contour"</code> .
x	[gfunc] - a numeric value or vector out of the range $[0, \text{Inf})$ at which the generator will be computed.
...	[ellipticalCopulaFit] - arguments passed to the optimization function <code>nlmminb</code> .

Value**Copula Functions:**

The functions `[rpd]ellipticalCopula` return a numeric vector of random variates, probabilities, or densities for the specified copula computed at grid coordinates `ulv`.

The functions `[rpd]ellipticalSlider` display an interactive graph of an perspective copula plot either for random variates, probabilities or densities. Alternatively, an image underlaid contour plot can be shown.

Copula Dependence Measures:

The functions `ellipticalTau` and `ellipticalRho` return a numeric value for Kendall's Tau and Spearman's Rho.

Copula Tail Coefficient:

The function `ellipticalTailCoeff` returns the coefficient of tail dependence for a specified copula. The function `ellipticalTailPlot` displays a whole plot for the upper or alternatively for the lower tail dependence as a function of u for a set of nine `rho` values.

Copula Generator Function:

The function `gfunc` computes the generator function for the specified copula, by default the normal copula. If the argument `x` is missing, then the normalization constant `lambda` will be returned, otherwise if `x` is specified the values for the function $g(x)$ will be returned. The selected type of copula is added to the output as an attribute named `"control"`. The function `gfuncSlider` allows to display interactively the generator function, the marginal density, the marginal probability, and the contours of the bivariate density.

Copula Simulation and Parameter Fitting:

The function `ellipticalCopulaSim` returns a numeric two-column matrix with randomly generated variates for the specified copula.

The function `ellipticalCopulaFit` returns a fit to empirical data for the specified copula. The returned object is a list with elements from the function `nlminb`.

Author(s)

Diethelm Wuertz for the Rmetrics R-port.

Examples

```
## ellipticalList -  
# List implemented copulae:  
ellipticalList()  
  
## gfunc -  
# Generator Function:  
gfunc(x = (0:10)/10, param = 2, type = "t")  
  
## gfuncSlider -  
# Try:  
if (require(tcltk)) {  
  gfuncSlider()  
}
```

 EllipticalModelling

Bivariate Elliptical Copulae

Description

A collection and description of functions to investigate bivariate elliptical copulae.

Elliptical Copulae Functions:

<code>ellipticalCopulaSim</code>	simulates an elliptical copula,
<code>ellipticalCopulaFit</code>	fits the parameters of an elliptical copula.

Usage

```
ellipticalCopulaSim(n, rho = 0.75, param = NULL, type = c("norm", "cauchy", "t"))
ellipticalCopulaFit(u, v, type = c("norm", "cauchy", "t"), ...)
```

Arguments

<code>n</code>	<code>[rellipticalCopula][ellipticalCopulaSim]</code> - the number of random deviates to be generated, an integer value.
<code>rho</code>	<code>[*ellipticalCopula]</code> - is the numeric value setting the correlation strength, ranging between minus one and one.
<code>param</code>	<code>[*ellipticalCopula][gfunc]</code> - additional distributional parameters: for the Student-t distribution this is "nu", for the Kotz distribution this is "r", and for the Exponential Power distribution these are "r" and "s". If the argument <code>param=NULL</code> then default values are taken. These are for the Student-t <code>param=c(nu=4)</code> , for the Kotz distribution <code>param=c(r=1)</code> , and for the exponential power distribution <code>param=c(r=1, s=1)</code> . Note, that the Kotz and exponential power copulae are independent of <code>r</code> , and that <code>r</code> only enters the generator, the density, the probability and the quantile functions.
<code>type</code>	<code>[*ellipticalCopula][gfunc]</code> - the type of the elliptical copula. A character string selected from: "norm", "cauchy", "t", "logistic", "laplace", "kotz", or "epower". <code>[*ellipticalSlider]</code> - a character string which indicates what kind of plot should be displayed, either a perspective plot if <code>type="persp"</code> , the default value, or a contour plot if <code>type="contour"</code> .
<code>u, v</code>	<code>[*ellipticalCopula]</code> - two numeric values or vectors of the same length at which the copula will be computed. If <code>u</code> is a list then the <code>\$x</code> and <code>\$y</code> elements will be used as <code>u</code> and <code>v</code> . If <code>u</code> is a two column matrix then the first column will be used as <code>u</code> and the second as <code>v</code> . If <code>u</code> is an integer value greater than one, say <code>N</code> , then the values for all points on the $[(0:N)/N]^2$ grid spanning the unit square will be returned.

... [ellipticalCopulaFit] -
arguments passed to the optimization function `nlsminb`.

Value

Copula Functions:

The functions `[rpd]ellipticalCopula` return a numeric vector of random variates, probabilities, or densities for the specified copula computed at grid coordinates `ulv`.

The functions `[rpd]ellipticalSlider` display an interactive graph of an perspective copula plot either for random variates, probabilities or densities. Alternatively, an image underlayed contour plot can be shown.

Copula Dependence Measures:

The functions `ellipticalTau` and `ellipticalRho` return a numeric value for Kendall's Tau and Spearman's Rho.

Copula Tail Coefficient:

The function `ellipticalTailCoeff` returns the coefficient of tail dependence for a specified copula. The function `ellipticalTailPlot` displays a whole plot for the upper or alternatively for the lower tail dependence as a function of `u` for a set of nine `rho` values.

Copula Generator Function:

The function `gfunc` computes the generator function for the specified copula, by default the normal copula. If the argument `x` is missing, then the normalization constant `lambda` will be returned, otherwise if `x` is specified the values for the function $g(x)$ will be returned. The selected type of copula is added to the output as an attribute named "control". The function `gfuncSlider` allows to display interactively the generator function, the marginal density, the marginal probability, and the contours of the bivariate density.

Copula Simulation and Parameter Fitting:

The function `ellipticalCopulaSim` returns a numeric two-column matrix with randomly generated variates for the specified copula.

The function `ellipticalCopulaFit` returns a fit to empirical data for the specified copula. The returned object is a list with elements from the function `nlsminb`.

Author(s)

Diethelm Wuertz for the Rmetrics R-port.

Examples

```
## [rp]ellipticalCopula -
# Default Normal Copula:
rellipticalCopula(10)
pellipticalCopula(10)

## [rp]ellipticalCopula -
# Student-t Copula Probability and Density:
u = grid2d(x = (0:25)/25)
pellipticalCopula(u, rho = 0.75, param = 4,
  type = "t", output = "list")
d = dellipticalCopula(u, rho = 0.75, param = 4,
  type = "t", output = "list")
persp(d, theta = -40, phi = 30, col = "steelblue")

## ellipticalTau -
## ellipticalRho -
# Dependence Measures:
ellipticalTau(rho = -0.5)
ellipticalRho(rho = 0.75, type = "logistic", subdivisions = 100)

## ellipticalTailCoeff -
# Student-t Tail Coefficient:
ellipticalTailCoeff(rho = 0.25, param = 3, type = "t")

## gfunc -
# Generator Function:
plot(gfunc(x = 0:10), main = "Generator Function")

## ellipticalCopulaSim -
## ellipticalCopulaSim -
# Simualtion and Parameter Fitting:
rv = ellipticalCopulaSim(n = 100, rho = 0.75)
ellipticalCopulaFit(rv)
```

EmpiricalCopulae *Bivariate Empirical Copulae*

Description

A collection and description of functions to investigate bivariate empirical copulae.

Empirical Copulae Functions:

pempiricalCopula	computes empirical copula probability,
dempiricalCopula	computes empirical copula density.

Usage

```
pempiricalCopula(u, v, N = 10)
dempiricalCopula(u, v, N = 10)
```

Arguments

`N` [empiricalCopula] -
... .

`u, v` [*evCopula][*archmCopula] -
two numeric values or vectors of the same length at which the copula will be computed. If `u` is a list then the `$x` and `$y` elements will be used as `u` and `v`. If `u` is a two column matrix then the first column will be used as `u` and the second as `v`.

Value

The functions `*Spec` return an S4 object of class "fCOPULA". The object contains the following slots:

`@call` the function call.

`@copula` the name of the copula.

`@param` a list whose elements specify the model parameters.

`@title` a character string with the name of the copula. This can be overwritten specifying a user defined input argument.

`@description` a character string with an optional user defined description. By default just the current date when the test was applied will be returned.

The function `pcopula` returns a numeric matrix of probabilities computed at grid positions `xly`.

The function `parchmCopula` returns a numeric matrix with values computed for the Archimedean copula.

The function `darchmCopula` returns a numeric matrix with values computed for the density of the Archimedean copula.

The functions `Phi*` return a numeric vector with the values computed from the Archimedean generator, its derivatives, or its inverse.

The functions `cK` and `cKInv` return a numeric vector with the values of the density and inverse for Archimedean copulae.

Author(s)

Diethelm Wuertz for the Rmetrics R-port.

 ExtremeValueCopulae

Bivariate Extreme Value Copulae

Description

A collection and description of functions to investigate bivariate extreme value copulae.

Extreme Value Copulae Functions:

revCopula	Generates extreme value copula random variates,
pevCopula	computes extreme value copula probability,
devCopula	computes extreme value copula density,
revSlider	displays interactive plots of extreme value random variates,
pevSlider	displays interactive plots of extreme value probability,
devSlider	displays interactive plots of extreme value density.

Usage

```

revCopula(n, param = NULL, type = evList())
pevCopula(u = 0.5, v = u, param = NULL, type = evList(),
  output = c("vector", "list"), alternative = FALSE )
devCopula(u = 0.5, v = u, param = NULL, type = evList(),
  output = c("vector", "list"), alternative = FALSE )

revSlider(B = 10)
pevSlider(type = c("persp", "contour"), B = 10)
devSlider(type = c("persp", "contour"), B = 10)

```

Arguments

alternative	[evRho][evTau][*evCopula] - Should the probability be computed alternatively in a direct way from the probability formula or by default via the dependency function?
B	[*Slider] - the maximum slider menu value when the boundary value is infinite. By default this is set to 10.
n	[revCopula][evCopulaSim] - the number of random deviates to be generated, an integer value.
output	[*evCopula] - output - a character string specifying how the output should be formatted. By default a vector of the same length as <i>u</i> and <i>v</i> . If specified as "list" then <i>u</i> and <i>v</i> are expected to span a two-dimensional grid as outputted by the function <code>grid2d</code> and the function returns a list with elements <i>x</i> , <i>y</i> , and <i>z</i> which can be directly used for example by 2D plotting functions.

<code>param</code>	<code>[*ev*][A*]</code> - distribution and copulae parameters. A numeric value or vector of named parameters as required by the copula specified by the variable <code>type</code> . If set to <code>NULL</code> , then the default parameters will be taken.
<code>type</code>	<code>[*ev*][Afunc]</code> - the type of the extreme value copula. A character string selected from: "gumbel", "galambos", "husler.reiss", "tawn", or "bb5". <code>[evSlider]</code> - a character string specifying the plot type. Either a perspective plot which is the default or a contour plot with an underlying image plot will be created.
<code>u, v</code>	<code>[*evCopula][*archmCopula]</code> - two numeric values or vectors of the same length at which the copula will be computed. If <code>u</code> is a list then the <code>\$x</code> and <code>\$y</code> elements will be used as <code>u</code> and <code>v</code> . If <code>u</code> is a two column matrix then the first column will be used as <code>u</code> and the second as <code>v</code> .
<code>...</code>	<code>[evCopulaFit]</code> - arguments passed to the optimization function <code>nlm</code> .

Value

The function `pcopula` returns a numeric matrix of probabilities computed at grid positions `xly`.

The function `parchmCopula` returns a numeric matrix with values computed for the Archimedean copula.

The function `darchmCopula` returns a numeric matrix with values computed for the density of the Archimedean copula.

The functions `Phi*` return a numeric vector with the values computed from the Archimedean generator, its derivatives, or its inverse.

The functions `cK` and `cKInv` return a numeric vector with the values of the density and inverse for Archimedean copulae.

Author(s)

Diethelm Wuertz for the Rmetrics R-port.

Examples

```
## fCOPULA -
  getClass("fCOPULA")
  getSlots("fCOPULA")

## revCopula -
  # Not yet implemented
  # revCopula(n = 10, type = "galambos")
```

```
## pevCopula -
  pevCopula(u = grid2d(), type = "galambos", output = "list")

## devCopula -
  devCopula(u = grid2d(), type = "galambos", output = "list")

## AfuncSlider -
  # Generator, try:
  if (require(tcltk)) {
    AfuncSlider()
  }
```

ExtremeValueDependency

Bivariate Extreme Value Copulae

Description

A collection and description of functions to investigate bivariate extreme value copulae.

Extreme Value Copulae Functions:

evTau	Computes Kendall's tau for extreme value copulae,
evRho	computes Spearman's rho for extreme value copulae,
evTailCoeff	computes tail dependence for extreme value copulae,
evTailCoeffSlider	plots tail dependence for extreme value copulae.

Usage

```
evTau(param = NULL, type = evList(), alternative = FALSE)
evRho(param = NULL, type = evList(), alternative = FALSE)

evTailCoeff(param = NULL, type = evList())
evTailCoeffSlider(B = 10)
```

Arguments

alternative	[evRho][evTau][*evCopula] - Should the probability be computed alternatively in a direct way from the probability formula or by default via the dependency function?
B	[*Slider] - the maximum slider menu value when the boundary value is infinite. By default this is set to 10.
param	[*ev*][A*] - distribution and copulae parameters. A numeric value or vector of named parameters as required by the copula specified by the variable <code>type</code> . If set to <code>NULL</code> , then the default parameters will be taken.

```

type      [*ev*][Afunc] -
          the type of the extreme value copula. A character string selected from: "gumbel", "galambos", "husler.reiss", "tawn", or "bb5".
[evSlider] -
          a character string specifying the plot type. Either a perspective plot which is the default or a contour plot with an underlying image plot will be created.
...      [evCopulaFit] -
          arguments passed to the optimization function nlmminb.

```

Value

The function `pcopula` returns a numeric matrix of probabilities computed at grid positions `xly`.

The function `parchmCopula` returns a numeric matrix with values computed for the Archimedean copula.

The function `darchmCopula` returns a numeric matrix with values computed for the density of the Archimedean copula.

The functions `Phi*` return a numeric vector with the values computed from the Archimedean generator, its derivatives, or its inverse.

The functions `cK` and `cKInv` return a numeric vector with the values of the density and inverse for Archimedean copulae.

Author(s)

Diethelm Wuertz for the Rmetrics R-port.

Examples

```

## fCOPULA -
  getClass("fCOPULA")
  getSlots("fCOPULA")

## revCopula -
  # Not yet implemented
  # revCopula(n = 10, type = "galambos")

## pevCopula -
  pevCopula(u = grid2d(), type = "galambos", output = "list")

## devCopula -
  devCopula(u = grid2d(), type = "galambos", output = "list")

## AfuncSlider -
  # Generator, try:
  if (require(tcltk)) {
    AfuncSlider()
  }

```

 ExtremeValueGenerator

Bivariate Extreme Value Copulae

Description

A collection and description of functions concerned with the generator function for the extreme value copula and with functions for setting and checking the distributional parameters.

Functions:

<code>evList</code>	Returns list of implemented extreme value copulae,
<code>evParam</code>	sets default parameters for an extreme value copula,
<code>evRange</code>	returns the range of valid rho values,
<code>evCheck</code>	checks if rho is in the valid range,
<code>Afunc</code>	computes dependence function,
<code>AfuncSlider</code>	displays interactively dependence function.

Usage

```
evList()
evParam(type = evList())
evRange(type = evList())
evCheck(param, type = evList())

Afunc(x, param = NULL, type = evList())
AfuncSlider()
```

Arguments

<code>param</code>	distribution and copulae parameters. A numeric value or vector of named parameters as required by the copula specified by the variable <code>type</code> . If set to <code>NULL</code> , then the default parameters will be taken.
<code>type</code>	the type of the extreme value copula. A character string selected from: "gumbel", "galambos", "husler.reiss", "tawn", or "bb5".
<code>x</code>	a numeric value or vector ranging between zero and one.

Value

The function `pcopula` returns a numeric matrix of probabilities computed at grid positions `xly`.

The function `parchmCopula` returns a numeric matrix with values computed for the Archimedean copula.

The function `darchmCopula` returns a numeric matrix with values computed for the density of the Archimedean copula.

The functions `Phi*` return a numeric vector with the values computed from the Archimedean generator, its derivatives, or its inverse.

The functions `cK` and `cKInv` return a numeric vector with the values of the density and inverse for Archimedean copulae.

Author(s)

Diethelm Wuertz for the Rmetrics R-port.

Examples

```
## fCOPULA -
  getClass("fCOPULA")
  getSlots("fCOPULA")

## revCopula -
  # Not yet implemented
  # revCopula(n = 10, type = "galambos")

## pevCopula -
  pevCopula(u = grid2d(), type = "galambos", output = "list")

## devCopula -
  devCopula(u = grid2d(), type = "galambos", output = "list")

## AfuncSlider -
  # Generator, try:
  if (require(tcltk)) {
    AfuncSlider()
  }
```

ExtremeValueModelling

Bivariate Extreme Value Copulae

Description

A collection and description of functions to investigate bivariate extreme value copulae.

Extreme Value Copulae Functions:

<code>evCopulaSim</code>	simulates an extreme value copula,
<code>evCopulaFit</code>	fits the parameters of an extreme value copula.

Usage

```
evCopulaSim(n, param = NULL, type = evList())
evCopulaFit(u, v = NULL, type = evList(), ...)
```

Arguments

<code>n</code>	[revCopula][evCopulaSim] - the number of random deviates to be generated, an integer value.
<code>param</code>	[*ev*][A*] - distribution and copulae parameters. A numeric value or vector of named parameters as required by the copula specified by the variable <code>type</code> . If set to <code>NULL</code> , then the default parameters will be taken.
<code>type</code>	[*ev*][Afunc] - the type of the extreme value copula. A character string selected from: "gumbel", "galambos", "husler.reiss", "tawn", or "bb5". [evSlider] - a character string specifying the plot type. Either a perspective plot which is the default or a contour plot with an underlying image plot will be created.
<code>u, v</code>	[*evCopula][*archmCopula] - two numeric values or vectors of the same length at which the copula will be computed. If <code>u</code> is a list then the <code>\$x</code> and <code>\$y</code> elements will be used as <code>u</code> and <code>v</code> . If <code>u</code> is a two column matrix then the first column will be used as <code>u</code> and the second as <code>v</code> .
<code>...</code>	[evCopulaFit] - arguments passed to the optimization function <code>nlm</code> .

Value

The function `pcopula` returns a numeric matrix of probabilities computed at grid positions `x|y`.

The function `parchmCopula` returns a numeric matrix with values computed for the Archimedean copula.

The function `darchmCopula` returns a numeric matrix with values computed for the density of the Archimedean copula.

The functions `Phi*` return a numeric vector with the values computed from the Archimedean generator, its derivatives, or its inverse.

The functions `cK` and `cKInv` return a numeric vector with the values of the density and inverse for Archimedean copulae.

Author(s)

Diethelm Wuertz for the Rmetrics R-port.

Examples

```
## fCOPULA -
  getClass("fCOPULA")
  getSlots("fCOPULA")

## revCopula -
  # Not yet implemented
  # revCopula(n = 10, type = "galambos")

## pevCopula -
  pevCopula(u = grid2d(), type = "galambos", output = "list")

## devCopula -
  devCopula(u = grid2d(), type = "galambos", output = "list")

## AfuncSlider -
  # Generator, try:
  if (require(tcltk)) {
    AfuncSlider()
  }
```

 MultivariateDistribution

Multivariate Skew Normal and Student-t Distributions

Description

A collection and description of functions to compute multivariate densities and probabilities from skew normal and skew Student-t distribution functions. Furthermore, multivariate random deviates can be generated, and for multivariate data, the parameters of the underlying distribution can be estimated by the maximum log-likelihood estimation.

The functions are:

<code>dmvsnorm</code>	Multivariate Skew Normal Density,
<code>pmvsnorm</code>	Multivariate Skew Normal Probability,
<code>rmvsnorm</code>	Random Deviates from MV Skew Normal Distribution,
<code>dmvst</code>	Multivariate Skew Student Density,
<code>pmvst</code>	Multivariate Skew Student Probability,
<code>rmvst</code>	Random Deviates from MV Skew Student Distribution,
<code>mvFit</code>	Fits a MV Skew Normal or Student-t Distribution,
<code>print</code>	S3 print method for an object of class 'fMV',
<code>plot</code>	S3 Plot method for an object of class 'fMV',
<code>summary</code>	S3 summary method for an object of class 'fMV'.

These functions are useful for portfolio selection and optimization if one likes to model the data by multivariate normal, skew normal, or skew Student-t distribution functions.

Usage

```

dmvsnorm(x, dim = 2, mu = rep(0, dim), Omega = diag(dim), alpha = rep(0, dim))
pmvsnorm(q, dim = 2, mu = rep(0, dim), Omega = diag(dim), alpha = rep(0, dim))
rmvsnorm(n, dim = 2, mu = rep(0, dim), Omega = diag(dim), alpha = rep(0, dim))

dmvst(x, dim = 2, mu = rep(0, dim), Omega = diag(dim), alpha = rep(0, dim), df = 4)
pmvst(q, dim = 2, mu = rep(0, dim), Omega = diag(dim), alpha = rep(0, dim), df = 4)
rmvst(n, dim = 2, mu = rep(0, dim), Omega = diag(dim), alpha = rep(0, dim), df = 4)

mvFit(x, method = c("snorm", "st"), fixed.df = NA, title = NULL,
      description = NULL, trace = FALSE, ...)

## S4 method for signature 'fMV':
show(object)

## S3 method for class 'fMV':
plot(x, which = "ask", ...)
## S3 method for class 'fMV':
summary(object, which = "ask", doplot = TRUE, ...)

```

Arguments

description	[mvFit] - a character string, assigning a brief description to an "fMV" object.
doplot	a logical value, by default TRUE. Should a plot be generated and displayed?
dim	[*mvsnorm][*mvst] - the column dimension of the matrix x. If x is specified as a vector, dim=1 must be set to one.
fixed.df	either NA, the default, or a numeric value assigning the number of degrees of freedom to the model. In the case that fixed.df=NA the value of df will be included in the optimization process, otherwise not.
method	[mvFit] - a string value specifying the method applied in the optimizing process. This can be either method="snorm" or method="st", in the first case the parameters for a skew normal distribution will be fitted and in the second case the parameters for a skew Student-t distribution.
mu, Omega, alpha, df	[*mvsnorm][*mvst] - the model parameters: mu a vector of mean values, one for each column, Omega the covariance matrix, alpha the skewness vector, and df the number of degrees of freedom which is a measure for the fatness of the tails (excess kurtosis). For a symmetric distribution alpha is a vector of zeros. For the normal distributions df is not used and set to infinity, Inf. Note that all columns assume the same value for df.

<code>n</code>	<code>[rmvsnorm][rmvst]</code> - number of data records to be simulated, an integer value.
<code>object</code>	<code>[summary]</code> - an object of class <code>fMV</code> .
<code>title</code>	<code>[mvFit]</code> - a character string, assigning a title to an "fMV" object.
<code>trace</code>	a logical, if set to <code>TRUE</code> the optimization process will be traced, otherwise not. The default setting is <code>FALSE</code> .
<code>which</code>	which of the five plots should be displayed? <code>which</code> can be either a character string, "all" (displays all plots) or "ask" (interactively asks which one to display), or a vector of 5 logical values, for those elements which are set <code>TRUE</code> the corresponding plot will be displayed.
<code>x, q</code>	<code>[*mvsnorm][*mvst][mvFit]</code> - a numeric matrix of quantiles (returns) or any other rectangular object like a <code>data.frame</code> or a multivariate time series objects which can be transformed by the function <code>as.matrix</code> to an object of class <code>matrix</code> . If <code>x</code> is a vector, it will be transformed into a matrix object with one column. <code>[plot][print]</code> - An object of class <code>fMV</code> .
<code>...</code>	optional arguments to be passed to the optimization or plotting functions.

Details

These are "easy-to-use" functions which allow quickly to simulate multivariate data sets and to fit their parameters assuming a multivariate skew normal or skew Student-t distribution. The functions make use of the contributed R packages `sn` and `mtvnorm`.

For an extended functionality in modelling multivariate skew normal and Student-t distributions we recommend to download and use the functions from the original package `sn` which requires also the package `mtvnorm`.

The algorithm for the computation of the normal and Student-t distribution functions is described by Genz (1992) and (1993), and its implementation by Hothorn, Bretz, and Genz (2001).

The parameter estimation is done by the maximum log-likelihood estimation. The algorithm and the implementation was done by Azzalini (1985-2003).

The multivariate skew-normal distribution is discussed in detail by Azzalini and Dalla Valle (1996); the (Ω, α) parametrization adopted here is the one of Azzalini and Capitanio (1999).

The family of multivariate skew-t distributions is an extension of the multivariate Student's t family, via the introduction of a shape parameter which regulates skewness; for a zero shape parameter the skew Student-t distribution reduces to the usual t distribution. When $df = \text{Inf}$ the distribution reduces to the multivariate skew-normal one.

The plot facilities have been completely reimplemented. The S3 plot method allows for selective batch and interactive plots. The argument `which` takes care for the desired operation.

The contributed R package `mtvnorm` is required, the contributed R package `sn` is builtin, since it is not available on the Debian Software Server.

Value

[dp]mvsnorm

[dp]mvst

return a vector of density and probability values computed from the matrix x .

mvFit

returns a S4 object class of class "fASSETS", with the following slots:

@call the matched function call.

@data the input data in form of a data.frame.

@description allows for a brief project description.

@fit the results as a list returned from the underlying fitting function.

@method the selected method to fit the distribution, either "snorm", or "st".

@model the model parameters describing the fitted parameters in form of a list, model=list(mu, Omega, alpha, df).

@title a title string.

@fit\$dp a list containing the direct parameters beta, Omega, alpha. Here, beta is a matrix of regression coefficients with $\dim(\text{beta}) = c(\text{nrow}(X), \text{ncol}(Y))$, Omega is a covariance matrix of order dim, alpha is a vector of shape parameters of length dim.

@fit\$se a list containing the components beta, alpha, info. Here, beta and alpha are the standard errors for the corresponding point estimates; info is the observed information matrix for the working parameter, as explained below.

@fit\$optim the list returned by the optimizer optim; see the documentation of this function for explanation of its components.

print

is the S3 print method for objects of class "fMV" returned from the function mvFit. It shows a summary report of the parameter fit.

plot

is the S3 plot method for objects of class "fMV" returned from the function mvFit. Five plots are produced. The first plot produces a scatterplot and in one dimension an histogram plot with the fitted distribution superimposed. The second and third plot represent a QQ-plots of Mahalanobis distances. The first of these refers to the fitting of a multivariate normal distribution, a standard statistical procedure; the second gives the corresponding QQ-plot of suitable Mahalanobis distances for the multivariate skew-normal fit. The fourth and fifth plots are similar to the previous ones, except that PP-plots are produced. The plots can be displayed in several ways, depending on the argument which, for details we refer to the arguments list above.

summary

is the S3 summary method for objects of class "fMV" returned from the function mvFit. The summary method prints and plots in one step the results as done by the print and plot methods.

Author(s)

Torsten Hothorn for R's `mvtnorm` package,
 Alan Ganz and Frank Bretz for the underlying Fortran Code,
 Adelchi Azzalini for R's `sn` package,
 Diethelm Wuertz for the Rmetrics port.

References

- Azzalini A. (1985); *A Class of Distributions Which Includes the Normal Ones*, Scandinavian Journal of Statistics 12, 171–178.
- Azzalini A. (1986); *Further Results on a Class of Distributions Which Includes the Normal Ones*, Statistica 46, 199–208.
- Azzalini A., Dalla Valle A. (1996); *The Multivariate Skew-normal Distribution*, Biometrika 83, 715–726.
- Azzalini A., Capitanio A. (1999); *Statistical Applications of the Multivariate Skew-normal Distribution*, Journal Roy. Statist. Soc. B61, 579–602.
- Azzalini A., Capitanio A. (2003); *Distributions Generated by Perturbation of Symmetry with Emphasis on a Multivariate Skew-t Distribution*, Journal Roy. Statist. Soc. B65, 367–389.
- Genz A., Bretz F. (1999); *Numerical Computation of Multivariate t-Probabilities with Application to Power Calculation of Multiple Contrasts*, Journal of Statistical Computation and Simulation 63, 361–378.
- Genz A. (1992); *Numerical Computation of Multivariate Normal Probabilities*, Journal of Computational and Graphical Statistics 1, 141–149.
- Genz A. (1993); *Comparison of Methods for the Computation of Multivariate Normal Probabilities*, Computing Science and Statistics 25, 400–405.
- Hothorn T., Bretz F., Genz A. (2001); *On Multivariate t and Gauss Probabilities in R*, R News 1/2, 27–29.

Examples

```
## rmvst -
par(mfcol = c(3, 1), cex = 0.7)
r1 = rmvst(200, dim = 1)
ts.plot(as.ts(r1), xlab = "r", main = "Student-t 1d")
r2 = rmvst(200, dim = 2, Omega = matrix(c(1, 0.5, 0.5, 1), 2))
ts.plot(as.ts(r2), xlab = "r", col = 2:3, main = "Student-t 2d")
r3 = rmvst(200, dim = 3, mu = c(-1, 0, 1), alpha = c(1, -1, 1), df = 5)
ts.plot(as.ts(r3), xlab = "r", col = 2:4, main = "Skew Student-t 3d")

## mvFit -
# Generate Grid Points:
n = 51
x = seq(-3, 3, length = n)
xoy = cbind(rep(x, n), as.vector(matrix(x, n, n, byrow = TRUE)))
X = matrix(xoy, n * n, 2, byrow = FALSE)
head(X)
# The Bivariate Normal Case:
```

```

Z = matrix(dmvsnorm(X, dim = 2), length(x))
par (mfrow = c(2, 2), cex = 0.7)
persp(x, x, Z, theta = -40, phi = 30, col = "steelblue")
title(main = "Bivariate Normal Plot")
image(x, x, Z)
title(main = "Bivariate Normal Contours")
contour(x, x, Z, add = TRUE)
# The Bivariate Skew-Student-t Case:
mu = c(-0.1, 0.1)
Omega = matrix(c(1, 0.5, 0.5, 1), 2)
alpha = c(-1, 1)
Z = matrix(dmvst(X, 2, mu, Omega, alpha, df = 3), length(x))
persp(x, x, Z, theta = -40, phi = 30, col = "steelblue")
title(main = "Bivariate Student-t Plot")
image(x, x, Z)
contour(x, x, Z, add = TRUE)
title(main = "Bivariate Student-t Contours")

```

norm2d

*Bivariate Normal Distribution***Description**

Density, distribution function, and random generation for the bivariate normal distribution.

Usage

```

pnorm2d(x, y = x, rho = 0)
dnorm2d(x, y = x, rho = 0)
rnorm2d(n, rho = 0)

```

Arguments

n	the number of random deviates to be generated, an integer value.
rho	the correlation parameter, a numeric value ranging between minus one and one, by default zero.
x, y	two numeric vectors defining the x and y coordinates.

Value

pnorm2d
returns a two column matrix of probabilities for the bivariate normal distribution function.

dnorm2d
returns a two column matrix of densities for the bivariate normal distribution function.

rnorm2d
returns a two column matrix of random deviates generated from the bivariate normal distribution function.

Author(s)

Adelchi Azzalini for the underlying pnorm2d function,
Diethelm Wuertz for the Rmetrics R-port.

References

Azzalini A., (2004); *The sn Package*; R Reference Guide available from www.r-project.org.
Venables W.N., Ripley B.D., (2002); *Modern Applied Statistics with S*, Fourth Edition, Springer.

Examples

```
## Bivariate Normal Density:
x = (-40:40)/10
X = grid2d(x)
z = dnorm2d(X$x, X$y, rho = 0.5)
Z = list(x = x, y = x, z = matrix(z, ncol = length(x)))
persp(Z, theta = -40, phi = 30, col = "steelblue")
```

t2d

Bivariate Student-t Distribution

Description

Density, distribution function, and random generation for the bivariate Student-t distribution.

Usage

```
pt2d(x, y = x, rho = 0, nu = 4)
dt2d(x, y = x, rho = 0, nu = 4)
rt2d(n, rho = 0, nu = 4)
```

Arguments

n	the number of random deviates to be generated, an integer value.
nu	the number of degrees of freedom, a numeric value ranging between two and infinity, by default four.
rho	the correlation parameter, a numeric value ranging between minus one and one, by default zero.
x, y	two numeric vectors defining the x and y coordinates.

Value

pt2d

returns a two column matrix of probabilities for the bivariate Student-t distribution function.

dt2d

returns a two column matrix of densities for the bivariate Student-t distribution function.

rt2d

returns a two column matrix of random deviates generated from the bivariate Student-t distribution function.

Author(s)

Adelchi Azzalini for the underlying pnorm2d function,
Diethelm Wuertz for the Rmetrics R-port.

References

Azzalini A., (2004); *The sn Package*; R Reference Guide available from www.r-project.org.
Venables W.N., Ripley B.D., (2002); *Modern Applied Statistics with S*, Fourth Edition, Springer.

Examples

```
## Bivariate Student-t Density:
x = (-40:40)/10
X = grid2d(x)
z = dt2d(X$x, X$y, rho = 0.5, nu = 6)
Z = list(x = x, y = x, z = matrix(z, ncol = length(x)))
persp(Z, theta = -40, phi = 30, col = "steelblue")
```

Index

*Topic **distribution**

MultivariateDistribution, 39

*Topic **math**

adapt, 2

cauchy2d, 12

density2d, 16

elliptical2d, 17

norm2d, 44

t2d, 45

*Topic **models**

ArchimedeanCopulae, 4

ArchimedeanDependency, 6

ArchimedeanGenerator, 7

ArchimedeanModelling, 9

CopulaeClass, 14

EllipticalCopulae, 18

EllipticalDependency, 21

EllipticalGenerator, 24

EllipticalModelling, 27

EmpiricalCopulae, 30

ExtremeValueCopulae, 31

ExtremeValueDependency, 33

ExtremeValueGenerator, 35

ExtremeValueModelling, 37

*Topic **programming**

BivariateBinning, 10

BivariateGridding, 11

*Topic **utilities**

adapt, 2

adapt, 2

Afunc (ExtremeValueGenerator), 35

AfuncSlider

(ExtremeValueGenerator), 35

ArchimedeanCopulae, 4

ArchimedeanDependency, 6

ArchimedeanGenerator, 7

ArchimedeanModelling, 9

archmCheck

(ArchimedeanGenerator), 7

archmCopulaFit

(ArchimedeanModelling), 9

archmCopulaSim

(ArchimedeanModelling), 9

archmList (ArchimedeanGenerator),

7

archmParam

(ArchimedeanGenerator), 7

archmRange

(ArchimedeanGenerator), 7

archmRho (ArchimedeanDependency),

6

archmTailCoeff

(ArchimedeanDependency), 6

archmTailPlot

(ArchimedeanDependency), 6

archmTau (ArchimedeanDependency),

6

BivariateBinning, 10

BivariateGridding, 11

cauchy2d, 12

class, 3

contour.gridData

(BivariateGridding), 11

CopulaeClass, 14

darchmCopula

(ArchimedeanCopulae), 4

darchmSlider

(ArchimedeanCopulae), 4

dcauchy2d (cauchy2d), 12

delliptical2d (elliptical2d), 17

dellipticalCopula

(EllipticalCopulae), 18

dellipticalSlider

(EllipticalCopulae), 18

dempiricalCopula

(EmpiricalCopulae), 30

- density2d, 16
 devCopula (*ExtremeValueCopulae*), 31
 devSlider (*ExtremeValueCopulae*), 31
 dgumbelCopula
 (*ArchimedeanCopulae*), 4
 dmvsnorm
 (*MultivariateDistribution*), 39
 dmvt (*MultivariateDistribution*), 39
 dnorm2d (*norm2d*), 44
 dt2d (*t2d*), 45
- elliptical2d, 17
 ellipticalCheck
 (*EllipticalGenerator*), 24
 EllipticalCopulae, 18
 ellipticalCopulaFit
 (*EllipticalModelling*), 27
 ellipticalCopulaSim
 (*EllipticalModelling*), 27
 EllipticalDependency, 21
 EllipticalGenerator, 24
 ellipticalList
 (*EllipticalGenerator*), 24
 EllipticalModelling, 27
 ellipticalParam
 (*EllipticalGenerator*), 24
 ellipticalRange
 (*EllipticalGenerator*), 24
 ellipticalRho
 (*EllipticalDependency*), 21
 ellipticalTailCoeff
 (*EllipticalDependency*), 21
 ellipticalTailPlot
 (*EllipticalDependency*), 21
 ellipticalTau
 (*EllipticalDependency*), 21
 EmpiricalCopulae, 30
 evCheck (*ExtremeValueGenerator*), 35
 evCopulaFit
 (*ExtremeValueModelling*), 37
 evCopulaSim
 (*ExtremeValueModelling*), 37
 evList (*ExtremeValueGenerator*), 35
 evParam (*ExtremeValueGenerator*), 35
 evRange (*ExtremeValueGenerator*), 35
 evRho (*ExtremeValueDependency*), 33
 evTailCoeff
 (*ExtremeValueDependency*), 33
 evTailCoeffSlider
 (*ExtremeValueDependency*), 33
 evTau (*ExtremeValueDependency*), 33
 ExtremeValueCopulae, 31
 ExtremeValueDependency, 33
 ExtremeValueGenerator, 35
 ExtremeValueModelling, 37
- fCOPULA (*CopulaeClass*), 14
 fCOPULA-class (*CopulaeClass*), 14
 fMV (*MultivariateDistribution*), 39
 fMV-class
 (*MultivariateDistribution*), 39
- gfunc (*EllipticalGenerator*), 24
 gfuncSlider
 (*EllipticalGenerator*), 24
 grid2d (*density2d*), 16
 gridData (*BivariateGridding*), 11
- hexBinning (*BivariateBinning*), 10
 hist2d (*density2d*), 16
- integrate, 2, 3
 integrate2d (*density2d*), 16
- Kfunc (*ArchimedeanGenerator*), 7
 KfuncSlider
 (*ArchimedeanGenerator*), 7
- MultivariateDistribution, 39
 mvFit (*MultivariateDistribution*), 39
- norm2d, 44
- parchmCopula
 (*ArchimedeanCopulae*), 4
 parchmSlider
 (*ArchimedeanCopulae*), 4

- pcauchy2d (*cauchy2d*), 12
- pellipticalCopula
 - (*EllipticalCopulae*), 18
- pellipticalSlider
 - (*EllipticalCopulae*), 18
- pempiricalCopula
 - (*EmpiricalCopulae*), 30
- persp.gridData
 - (*BivariateGridding*), 11
- pevCopula (*ExtremeValueCopulae*), 31
- pevSlider (*ExtremeValueCopulae*), 31
- pfrechetCopula (*CopulaeClass*), 14
- pgumbelCopula
 - (*ArchimedeanCopulae*), 4
- Phi (*ArchimedeanGenerator*), 7
- PhiSlider (*ArchimedeanGenerator*), 7
- plot.fMV
 - (*MultivariateDistribution*), 39
- plot.hexBinning
 - (*BivariateBinning*), 10
- plot.squareBinning
 - (*BivariateBinning*), 10
- pmvsnorm
 - (*MultivariateDistribution*), 39
- pmvst (*MultivariateDistribution*), 39
- pnorm2d (*norm2d*), 44
- print.fMV
 - (*MultivariateDistribution*), 39
- print.integration (*adapt*), 2
- pt2d (*t2d*), 45

- rarchmCopula
 - (*ArchimedeanCopulae*), 4
- rarchmSlider
 - (*ArchimedeanCopulae*), 4
- rcauchy2d (*cauchy2d*), 12
- rellipticalCopula
 - (*EllipticalCopulae*), 18
- rellipticalSlider
 - (*EllipticalCopulae*), 18
- revCopula (*ExtremeValueCopulae*), 31
- revSlider (*ExtremeValueCopulae*), 31
- rgumbelCopula
 - (*ArchimedeanCopulae*), 4
- rmvsnorm
 - (*MultivariateDistribution*), 39
- rmvst (*MultivariateDistribution*), 39
- rnorm2d (*norm2d*), 44
- rt2d (*t2d*), 45

- show, fCOPULA-method
 - (*CopulaeClass*), 14
- show, fMV-method
 - (*MultivariateDistribution*), 39
- squareBinning (*BivariateBinning*), 10
- summary.fMV
 - (*MultivariateDistribution*), 39

- t2d, 45