Package ‘QRM’
February 15, 2013

Version 0.4-8
Date 2012-07-22
Title Provides R-language code to examine Quantitative Risk Management concepts
Depends R(>= 2.10.0), gsl, Matrix, mvtnorm, numDeriv, timeSeries
Description This package is designed to accompany the book
Quantitative Risk Management: Concepts, Techniques and Tools by
LazyData Yes
License GPL (>= 2)
Author Bernhard Pfaff [aut, cre], Alexander McNeil [aut] (S-Plus
original (QRMlib)), Scott Ulmann [trl] (First R port as package QRMlib)
Maintainer Bernhard Pfaff <bernhard@pfaffikus.de>
Repository CRAN
Repository/R-Forge/Project qrm
Repository/R-Forge/Revision 9
Date/Publication 2012-07-26 21:32:16
NeedsCompilation yes

R topics documented:

QRM-package .......................................................... 2
BiDensPlot ............................................................ 3
cac40 ................................................................. 4
CopulaAC ............................................................. 4
CopulaGauss .......................................................... 7
CopulaStudent ....................................................... 8
Credit ................................................................. 9
QRM-package

Description

This package is designed to accompany the book Quantitative Risk Management: Concepts, Techniques and Tools by Alexander J. McNeil, Rudiger Frey and Paul Embrechts.
Overview

This package provides functions for quantitative risk management as introduced in the book “Quantitative Risk Management: Concepts, Techniques and Tools” (henceforth: QRM). The S-Plus package “QRMlib” has been made available by the first author of the book and can be obtained by following the instructions on [http://www.ma.hw.ac.uk/~mcneil/book/QRMlib.html](http://www.ma.hw.ac.uk/~mcneil/book/QRMlib.html). A R port of this package has been made available on CRAN by Scott Ulmann. However, the package failed the checks and hence has been moved to the CRAN archive (QRMlib, version 1.4.5.1 as of 04/25/2011). This package is based on QRMlib, but (i) not all functions have been ported from QRMlib to QRM, (ii) the arguments of some functions have been modified, and (iii) the manual pages have been re-ordered by topic.

A list of the not ported functions is provided in QRM-defunct with pointers to their replacements. This was achieved by the inclusion of dependencies to the packages gsl, numDeriv and timeSeries and/or resorting to functions contained in the base installation of R. Second, in particular with respect to passing down arguments to the routines used in optimizations and/or argument matching, modifications to the functions’ closures were necessary. In addition, the names of arguments in similar functions have been unified. Third, to provide the user a faster access to the manual pages of certain risk concepts, the functions’ documentation are now ordered by concept rather than by the name of the functions.

Without modifying the existing functions of QRMlib too much, neither S3- nor S4-classes and methods have been included completely by now in QRM, but the characteristic of the former package as a collection of functions pertinent to quantitative risk modelling have been kept intact. However, this might change in future releases of QRM. By now, the current package can be used almost alike QRMlib, but with the stated modifications.

References


---

**BiDensPlot**

*Bivariate Density Plot*

Description

Generates either a perspective or a contour plot of a bivariate density.

Usage

BiDensPlot(func, xpts = c(-2, 2), ypts = c(-2, 2), npts = 50, type = c("persp", "contour"), ...)

Arguments

- **func** function, the name of a bivariate density function.
- **xpts** vector, interval of x.
- **ypts** vector, interval of y.
CopulaAC

Description

Functions for evaluating densities of Archimedean copulae, generating random variates and fitting data to AC.

Examples

BiDensPlot(func = dmnorm, mu = c(0, 0), Sigma = equicorr(2, -0.7))

cac40

CAC 40 Stock Market Index (France)

Description

This timeSeries data set provides the daily closing values of the French CAC 40 stock index for the period 1994 to March 2004. In addition, the data set is also made available as a data.frame.

Usage

data(cac40)
data(cac40.df)

Examples

data(cac40)
head(cac40)
Usage

dcopula.AC(u, theta, name = c("clayton", "gumbel"), log = TRUE)
dcopula.clayton(u, theta, log = FALSE)
dcopula.gumbel(u, theta, log = FALSE)
rAC(name = c("clayton", "gumbel", "frank", "BB9", "GIG"), n, d, theta)
rACp(name = c("clayton", "gumbel", "frank", "BB9", "GIG"), n, d, theta, A)
rcopula.gumbel(n, theta, d)
rcopula.clayton(n, theta, d)
rcopula.frank(n, theta, d)
rstable(n, alpha, beta = 1)
rFrankMix(n, theta)
rbBMix(n, theta)
rcopula.Gumbel2Gp(n = 1000, gpsizes = c(2, 2), theta = c(2, 3, 5))
rcopula.GumbelNested(n, theta)
fit.AC(Udata, name = c("clayton", "gumbel"), initial = 2, ...)

Arguments

A  
matrix, dimension $d \times p$ containing asymmetry parameters. Rowsums must be equal to one.

alpha  
numeric, parameter $0 < \alpha \leq 2$, but $\alpha \neq 1$.

beta  
numeric, parameter $-1 \leq \beta \leq 1$.

d  
integer, dimension of copula.

gpsizes  
vector, length of two, containing the group sizes.

initial  
numeric, initial value used by fit.AC() in the call to nlminb().

log  
logical, whether log density values should be returned

n  
integer, count of random variates.

name  
character, name of copula.

theta  
vector, copula parameter(s).

u  
matrix, dimension $n \times d$, where $d$ is the dimension of the copula and $n$ is the number of vector values at which to evaluate density.

Udata  
matrix, pseudo-uniform observations.

...  
ellipsis, arguments are passed down to nlminb().

Details

The function dcopula.AC() is a generic function, designed such that additional copulae, or expressions for densities of higher-dimensional copulae may be added. Clayton copula works in any dimension at present but Gumbel is only implemented for $d = 2$. To extend, one must calculate the $d$-th derivative of the generator inverse and take the logarithm of absolute value; this is the term called loggfunc. In addition, for other copulae, one needs the generator $\phi$ and the log of the negative value of its first derivative $\text{lnegphidash}$.

The random variates from rAC() with arbitrary dimension are generated by using the mixture construction of Marshall and Olkin. It may be used in place of the other functions rcopula.clayton(),
rcopula.gumbel(), and rcopula.frank(). In addition, it allows simulation of BB9 and GIG copulas which don’t have individual simulation routines.

For the Clayton and Gumbel copulae, see page 192 and 222–224 in QRM. The random variates for the BB9 and Frank copula are obtained from a mixing distribution using a Laplace transform method (see page 224 of QRM). The function rcopula.Gumbel2Gp() generates sample from a Gumbel copula with two-group structure constructed using three Gumbel generators (see pages 222-224 and 227 of QRM). The function rcopula.gumbelNested() generates sample from a d-dimensional Gumbel copula with nested structure constructed using (d – 1) Gumbel generators.

For the random variates of the Stable distribution, a default value $\beta = 1$ is used; combined with a value for $\alpha < 1$ yields a positive stable distribution, which is required for Gumbel copula generation; the case $\alpha = 1$ has not been implemented.

**Value**

vector or matrix in case of the density and random-generator related functions and a list object for the fitting function.

**See Also**

rlminb

**Examples**

```r
## Gumbel
r1 <- rAC("gumbel", n = 50, d = 7, theta = 3)
head(r1)
## Weighted Gumbel
alpha <- c(0.95, 0.7)
wtmatrix <- cbind(alpha, 1 - alpha)
r2 <- rACp(name = "gumbel", n = 1000, d = 2, theta = c(4, 1),
A = wtmatrix)
head(r2)
## Gumbel with two-group structure
r3 <- rcopula.Gumbel2Gp(n = 3000, gpsizes = c(3, 4),
theta = c(2, 3, 5))
pairs(r3)
## Nested Gumbel
r4 <- rcopula.GumbelNested(n=3000,theta=1:6)
pairs(r4)
## Frank
r5 <- rcopula.frank(1000, 2, 4)
pairs(r5)
## Fitting of Gumbel and Clayton
data(smi)
data(ftse1)
s1 <- window(ftse1, "1990-11-09", "2004-03-25")
s1a <- alignDailySeries(s1)
s2a <- alignDailySeries(smi)
idx <- merge(s1a, s2a)
r <- returns(idx)
rp <- series(window(r, "1994-01-01", "2003-12-31"))
rp <- rp[(!rp[, 1] == 0) & (!rp[, 2] == 0),]
```
CopulaGauss

Udata <- apply(rp, 2, edf, adjust = 1)
mod.gumbel <- fit.AC(Udata, "gumbel")
mod.clayton <- fit.AC(Udata, "clayton")
mod.clayton

CopulaGauss

Gauss Copula

Description

Functions for evaluating the Gauss copula, generating random variates and fitting.

Usage

dcopula.gauss(Udata, Sigma, log = FALSE)
rcopula.gauss(n, Sigma)
fit.gausscopula(Udata, ...)

Arguments

log logical, whether log density values should be returned.
n integer, count of random variates
Sigma matrix, correlation matrix.
Udata matrix, pseudo-uniform data where rows are vector observations with all values in unit interval.
... ellipsis argument, passed down to nlminb() used in optimization.

Value

For dcopula.gauss() a vector of density values of length n. For rcopula.gauss() a n x d matrix of random variates and for fit.gausscopula() a list with the optimization results.

See Also

nlminb

Examples

ll <- c(0.01, 0.99)
BiDensPlot(func = dcopula.gauss, xpts = ll, ypts = ll, Sigma = equicorr(2, 0.5))
data <- rcopula.gauss(2000, Sigma = equicorr(d = 6, rho = 0.7))
pairs(data)
## Fitting Gauss Copula
data(smi)
data(ftse100)
s1 <- window(ftse100, "1990-11-09", "2004-03-25")
s1a <- alignDailySeries(s1)
s2a <- alignDailySeries(smi)
idx <- merge(s1a, s2a)
r <- returns(idx)
rp <- series(window(r, "1994-01-01", "2003-12-31"))
rp <- rp[(rp[, 1] != 0) & (rp[, 2] != 0), ]
Udata <- apply(rp, 2, edf, adjust = 1)
copgauss <- fit.gausscopula(Udata)

### CopulaStudent

#### Student's t Copula

**Description**

Functions for copula density, generating random variates and fitting

**Usage**

```
dcopula.t(Udata, df, Sigma, log = FALSE)
rcopula.t(n, df, Sigma)
fit.tcopula(Udata, method = c("all", "Kendall", "Spearman"),
            startdf = 5, ...)
```

**Arguments**

- `df` numeric, degrees of freedom.
- `log` logical, whether log density values should be returned.
- `method` character, method for fitting.
- `n` integer, count of random variates
- `Sigma` matrix, correlation matrix
- `startdf` numeric, initial DF value.
- `Udata` matrix, dimension $n \times d$, where $d$ is the dimension of the copula and $n$ is the number of pseudo-uniform values.

**Details**

If in the call to `fit.tcopula()`, `method = "all"`, then all parameters are estimated, *i.e.*, the degrees of freedom and the dispersion parameters (initial values from Spearman correlations). In case of either `method = "Kendall"` or `method = "Spearman"`, the corresponding rank correlations are used and the optimization is only carried out with respect to the degrees of freedom parameter. The initial value for the DF is given by `startdf`. See pages 197 and 229–236 of QRM.

**Value**

A vector of density values of length $n$ for `dcopula.t()`. A matrix of random variates for `rcopula.t()`. A list object containing parameter estimates and details of fit for function `fit.tcopula()`.
See Also

nlminb

Examples

```r
ll <- c(0.01, 0.99)
# create perspective plot for bivariate density:
BiDensPlot(func = dcopula.t, xpts = ll, ypts = ll, df = 4,
Sigma = equicorr(2, 0.5))
S <- equicorr(d = 6, rho = 0.7)
data <- rcopula.t(2000, df = 4, Sigma = S)
pairs(data)
## Fitting Student's Copula
data(smi)
data(ftse100)
s1 <- window(ftse100, "1990-11-09", "2004-03-25")
s1a <- alignDailySeries(s1)
s2a <- alignDailySeries(smi)
idx <- merge(s1a, s2a)
r <- returns(idx)
rp <- series(window(r, "1994-01-01", "2003-12-31"))
rp <- rp[(rp[, 1] != 0) & (rp[, 2] != 0),]
Udata <- apply(rp, 2, edf, adjust = 1)
copt1 <- fit.tcopula(Udata)
copt2 <- fit.tcopula(Udata, method = "Kendall")
```

Description

Functions for modelling credit risk:

- Bernoulli mixture model with prescribed default and joint default probabilities
- Bernoulli mixture model with Clayton copula dependencies of default.
- Probitnormal Mixture of Bernoullis
- Beta-Binomial Distribution
- Logitnormal-Binomial Distribution
- Probitnormal-Binomial Distribution

Usage

```r
cal.beta(pi1, pi2)
cal.claytonmix(pi1, pi2)
cal.probitnorm(pi1, pi2)
dclaytonmix(x, pi, theta)
```
p Claytonmix(q, pi, theta)
rc Claytonmix(n, pi, theta)
rtcopulamix(n, pi, rho.asset, df)
dprobitnorm(x, mu, sigma)
pprobitnorm(q, mu, sigma)
rprobitnorm(n, mu, sigma)
rbinomial.mixture(n = 1000, m = 100, 
                   model = c("probitnorm", "logitnorm", "beta"), ...)
r logitnorm(n, mu, sigma)
fit.binomial(M, m)
fit.binomialBeta(M, m, startvals = c(2, 2), ses = FALSE, ...)
fit.binomialLogitnorm(M, m, startvals = c(-1, 0.5), ...)
fit.binomialProbitnorm(M, m, startvals = c(-1, 0.5), ...)
momest(data, trials, limit = 10)

Arguments

data vector, numbers of defaults in each time period.
df numeric, degree of freedom.
limit integer, maximum order of joint default probability to estimate.
M vector, count of successes.
m vector, count of trials.
model character, name of mixing distribution.
mu numeric, location parameter.
n integer, count of random variates.
pi numeric, default probability.
pi1 numeric, default probability.
pi2 numeric, joint default probability.
q numeric, values at which CDF should be evaluated.
sigma numeric, scale parameter.
startvals numeric, starting values.
theta numeric, parameter of distribution.
trials vector, group sizes in each time period.
x numeric, values at which density should be evaluated.
rho.asset numeric, asset correlation parameter.
... ellipsis, arguments are passed down to either mixing distribution or nlminb().

Details

cal.beta(): calibrates a beta mixture distribution on unit interval to give an exchangeable Bernoulli mixture model with prescribed default and joint default probabilities (see pages 354-355 in QRM).
cal.claytonmix(): calibrates a mixture distribution on unit interval to give an exchangeable
Bernoulli mixture model with prescribed default and joint default probabilities. The mixture distribution is the one implied by a Clayton copula model of default (see page 362 in QRM).

cal.probitnorm(): calibrates a probitnormal mixture distribution on unit interval to give an exchangeable Bernoulli mixture model with prescribed default and joint default probabilities (see page 354 in QRM).

dclaytonmix(), pclaytonmix(), rclaytonmix(): density, cumulative probability, and random generation for a mixture distribution on the unit interval which gives an exchangeable Bernoulli mixture model equivalent to a Clayton copula model (see page 362 in QRM).

fit.binomial(): fits binomial distribution by maximum likelihood.

dprobitnorm(), pprobitnorm(), rprobitnorm(): density, cumulative probability and random number generation for distribution of random variable Q on unit interval such that the probit transform of Q has a normal distribution with parameters \( \mu \) and \( \sigma \) (see pages 353-354 in QRM).

fit.binomialBeta(): fits a beta-binomial distribution by maximum likelihood.

fit.binomialLogitnorm(): fits a mixed binomial distribution where success probability has a logitnormal distribution. Lower and upper bounds for the input parameters M and m can be specified by means of the arguments lower and upper, which are passed to nlminb(). If convergence occurs at an endpoint of either limit, one needs to reset lower and upper parameter estimators and run the function again.

fit.binomialProbitnorm(): Fits a mixed binomial distribution where success probability has a probitnormal distribution. Lower and upper bounds for the input parameters M and m can be specified by means of the arguments lower and upper, which are passed to nlminb(). If convergence occurs at an endpoint of either limit, one needs to reset lower and upper parameter estimators and run the function again.

momest(): calculates moment estimator of default probabilities and joint default probabilities for a homogeneous group. First returned value is default probability estimate; second value is estimate of joint default probability for two firms; and so on (see pages 375-376 in QRM).

rbinomial.mixture(): random variates from mixed binomial distribution (see pages 354-355 and pages 375-377 of QRM).

rlogitnorm(): Random number generation for distribution of random variable Q on unit interval such that the probit transform of Q has a normal distribution with parameters \( \mu \) and \( \sigma \) (see pages 353-354 in QRM).

rtcopulamix(): random generation for mixing distribution on unit interval yielding Student’s t copula model (see page 361 in QRM, exchangeable case of this model is considered).

See Also

link[stats]{nlminb}

Examples

```r
## calibrating models
pi.B <- 0.2
pi2.B <- 0.05
probitnorm.pars <- cal.probitnorm(pi.B, pi2.B)
probitnorm.pars
beta.pars <- cal.beta(pi.B, pi2.B)
beta.pars
claytonmix.pars <- cal.claytonmix(pi.B, pi2.B)
claytonmix.pars
q <- (1:1000) / 1001
```
q <- q[q < 0.25]
p.probitnorm <- pprobitnorm(q, probitnorm.pars[1], probitnorm.pars[2])
p.beta <- pbeta(q, beta.pars[1], beta.pars[2])
p.claytonmix <- pclaytonmix(q, claytonmix.pars[1], claytonmix.pars[2])
scale <- range((1 - p.probitnorm), (1 - p.beta), (1 - p.claytonmix))
plot(q, (1 - p.probitnorm), type = "l", log = "y", xlab = "q", ylab = "P(Q > q)", ylim=scale)
lines(q, (1 - p.beta), col = 2)
lines(q, (1 - p.claytonmix), col = 3)
legend("topleft", c("Probit-normal", "Beta", "Clayton-Mixture"), lty=rep(1,3),col = (1:3))

## Clayton Mix
pi.B <- 0.0489603
pi2.B <- 0.003126529
claytonmix.pars <- cal.claytonmix(pi.B, pi2.B)
claytonmix.pars
q <- (1:1000) / 1001
q <- q[q < 0.25]
d.claytonmix <- dclaytonmix(q, claytonmix.pars[1], claytonmix.pars[2])
head(d.claytonmix)

## SP Data
data(spdata.raw)
attach(spdata.raw)
BdefaultRate <- Bdefaults / Bobligors

## Binomial Model
# Binomial Logitnorm Model
mod1b <- fit.binomialLogitnorm(Bdefaults, Bobligors)
# Binomial Probitnorm Model
mod1c <- fit.binomialProbitnorm(Bdefaults, Bobligors)
# Binomial Beta Model
mod1d <- fit.binomialBeta(Bdefaults, Bobligors)

## Moment estimates for default probabilities
momest(Bdefaults, Bobligors)
pi.B <- momest(Bdefaults, Bobligors)[1]
pi2.B <- momest(Bdefaults, Bobligors)[2]

## Probitnorm
probitnorm.pars <- cal.probitnorm(pi.B, pi2.B)
q <- (1:1000)/1001
q <- q[q < 0.25]
d.probitnorm <- dprobitnorm(q, probitnorm.pars[1], probitnorm.pars[2])
p <- c(0.90, 0.95, 0.975, 0.99, 0.995, 0.999, 0.9999, 0.99999, 0.999999)
sigma <- 0.2 * 10000 / sqrt(250)
VaR.t4 <- qst(p, df = 4, sd = sigma, scale = TRUE)
VaR.t4
detach(spdata.raw)

## Binomial Mixture Models
pi <- 0.04896
pi2 <- 0.00321
beta.pars <- cal.beta(pi, pi2)
probitnorm.pars <- cal.probitnorm(pi, pi2)
Danish Fire Losses

**Description**

The danish timeSeries dataset provides the daily closing value for the Danish fire losses measured from January 1980 through December 1990. In addition, the data set is also made available as a data.frame.

**Usage**

```r
data(danish)
data(danish.df)
```

**Examples**

```r
data(danish)
head(danish)
```

Dow Jones 30 Stock Prices

**Description**

The DJ timeSeries data set provides the closing values of the Dow Jones 30 Stocks from 1991-2000. In addition, the data set is also made available as a data.frame.

**Usage**

```r
data(DJ)
data(DJ.df)
```

**Examples**

```r
data(DJ)
head(DJ)
```
Description

The dji timeSeries dataset provides the daily closing value for the Dow Jones index from January 1980 to March 2004. In addition, the data set is also made available as a data.frame.

Usage

data(dji)
data(dji.df)

Examples

data(dji)
head(dji)

Description

This function calculates the empirical distribution function at each element of a vector of observations.

Usage

edf(v, adjust = FALSE)

Arguments

v vector, observations of length n.
adjust logical, adjustment of denominator to be (n + 1).

Value

vector
**Examples**

```r
data(smi)
data(ftse100)
s1 <- window(ftse100, "1990-11-09", "2004-03-25")
s1a <- alignDailySeries(s1)
s2a <- alignDailySeries(smi)
idx <- merge(s1a, s2a)
r <- returns(idx)
rp <- series(window(r, "1994-01-01", "2003-12-31"))
Udata <- apply(rp, 2, edf, adjust = 1)
plot(Udata)
```

---

**eigenmeth**

*Make Matrix Positive Definite*

**Description**

The function adjusts a negative definite symmetric matrix to make it positive definite.

**Usage**

```r
eigenmeth(mat, delta = 0.001)
```

**Arguments**

- `mat`: matrix, a symmetric matrix
- `delta`: numeric, new size of smallest eigenvalues

**Details**

See page 231 of QRM.

**Value**

a positive-definite matrix
**equicorr**  
*Equal Correlation Matrix*

**Description**

Construction of an equal correlation matrix

**Usage**

`equicorr(d, rho)`

**Arguments**

- `d`: integer, dimension of matrix
- `rho`: numeric, value of correlation

**Value**

matrix

**Examples**

```r
equicorr(7, 0.5)
ll <- c(0.01, 0.99)
BiDensPlot(func = dcopula.gauss, xpts = ll, ypts = ll,
           Sigma = equicorr(2, 0.5))
BiDensPlot(func = dcopula.t, xpts = ll, ypts = ll, df = 4,
           Sigma = equicorr(2, 0.5))
```

**ES**  
*Expected Shortfall*

**Description**

Functions for computing the expected shortfall derived from the Normal or Student’s t distribution (see page 45 of QRM).

**Usage**

- `ESnorm(p, mu = 0, sd = 1)`
- `ESst(p, mu = 0, sd = 1, df, scale = FALSE)`
Arguments

- **p** numeric, probability
- **mu** numeric, location parameter
- **sd** numeric, scale parameter
- **df** numeric, degrees of freedom
- **scale** logical, scaling Student’s t distribution to have variance one

Value

numeric

Examples

```r
p <- c(0.95, 0.99)
s <- 0.2 * 10000 / sqrt(250)
ESnorm(p)
ESst(p, sd = s, df = 4, scale = TRUE)
ESst(p, df = 4)
```

---

**ftse100**  
*FTSE 100 Stock Market Index*

Description

The *ftse100* timeSeries dataset provides the daily closing value for the FTSE index from January 1980 to March 2004. In addition, the data set is also made available as a *data.frame*.

Usage

```r
data(ftse100)
data(ftse100.df)
```

Examples

```r
data(ftse100)
head(ftse100)
```
FXGBP.RAW  

*Sterling Exchange Rates*

**Description**
The FXGBP timeSeries dataset provides daily exchange rates for major currencies (US Dollar, Japanese Yen, Euro, Swiss franc) against the British Pound for the period January 1987 through March 2004. In addition, the data set is also made available as a data.frame.

**Usage**
```r
data(FXGBP)
data(FXGBP.df)
```

**Examples**
```r
data(FXGBP)
tail(FXGBP)
```

---

**Gauss**  

*Multivariate Gauss Distribution*

**Description**
Functions for evaluating multivariate normal density, generating random variates, fitting and testing.

**Usage**
```r
dmnorm(x, mu, Sigma, log = FALSE)
fit.norm(data)
rmnorm(n, mu = 0, Sigma)
MardiaTest(data)
jointnormalTest(data, dist = c("chisquare", "beta"), plot = TRUE)
```

**Arguments**
- **data**  
  matrix, data set.
- **dist**  
  character, “chisquare” performs test against $\chi^2$ distribution, which is an approximation; “beta” performs a test against a scaled beta distribution.
- **log**  
  logical, whether log density values shall be returned.
- **n**  
  integer, count of random variates.
- **mu**  
  numeric, location parameters.
- **plot**  
  logical, whether test result shall be plotted.
- **Sigma**  
  matrix, covariance matrix.
- **x**  
  matrix, density is evaluated per row.
Examples

BiDensPlot(func = dmnorm, mu = c(0, 0), Sigma = equicorr(2, -0.7))
S <- equicorr(d = 3, rho = 0.7)
data <- rmnorm(1000, Sigma = S)
fit.norm(data)
S <- equicorr(d = 10, rho = 0.6)
data <- rmnorm(1000, Sigma = S)
MardiaTest(data)
## Dow Jones Data
data(DJ)
r <- returns(DJ)
stocks <- c("AXP", "EK", "BA", "C", "KO", "MSFT",
           "HP", "INTC", "JPM", "DIS")
ss <- window(r[, stocks], "1993-01-01", "2000-12-31")
jointnormalTest(ss)

GEV  Generalized Extreme Value Distribution

Description

Density, quantiles, cumulative probability, and fitting of the Generalized Extreme Value distribution.

Usage

pGEV(q, xi, mu = 0, sigma = 1)
qGEV(p, xi, mu = 0, sigma = 1)
dGEV(x, xi, mu = 0, sigma = 1, log = FALSE)
rGEV(n, xi, mu = 0, sigma = 1)
fit.GEV(maxima, ...)

Arguments

log    logical, whether log values of density should be returned, default is FALSE.
maxima vector, block maxima data
mu     numeric, location parameter.
n      integer, count of random variates.
p      vector, probabilities.
q      vector, quantiles.
sigma  numeric, scale parameter.
x      vector, values to evaluate density.
xi     numeric, shape parameter.
...    ellipsis, arguments are passed down to optim().
Value

numeric, probability (pGEV), quantile (qGEV), density (dGEV) or random variates (rGEV) for the
GEV distribution with shape parameter ξ, location parameter μ and scale parameter σ. A list object
in case of fit.GEV().

See Also

GPD

Examples

quantValue <- 4.5
pGEV(q = quantValue, xi = 0, mu = 1.0, sigma = 2.5)
pGumbel(q = quantValue, mu = 1.0, sigma = 2.5)
## Fitting to monthly block-maxima
data(nasdaq)
l <- -returns(nasdaq)
em <- timeLastDayInMonth(time(l))
monmax <- aggregate(l, by = em, FUN = max)
mod1 <- fit.GEV(monmax)
gamma numeric, skewness parameter(s).
lambda numeric, mixing parameter(s).
log logical, should log density be returned; default is FALSE.
mu numeric, location parameter(s).
n integer, count of random variates.
psi numeric, mixing parameter(s).
Sigma matrix, dispersion matrix for multivariate GHYP.
x vector, values to evaluate density.

Details

The univariate QRM parameterization is defined in terms of parameters $\chi, \psi, \gamma$ instead of the $\alpha, \beta, \delta$ model used by Blaesild (1981). If $\gamma = 0$, a normal variance mixture where the mixing variable $W$ has a Generalized Inverse Gaussian distribution (GIG) with parameters $\lambda, \chi, \psi$ is given, with heavier tails. If $\gamma > 0$, a normal mean-variance mixture where the mean is also perturbed to equal $\mu + (W \ast \gamma)$ which introduces asymmetry as well, is obtained. Values for $\lambda$ and $\mu$ are identical in both QRM and B parameterizations. The dispersion matrix $\Sigma$ does not appear as argument in the univariate case since its value is identically one.

Value

numeric, value(s) of density or log-density (dghyp, dmghyp, dsmghyp and dghypB) or random sample (rghyp, rmghyp, rghypB)

Note

Density values from dgyhp() should be identical to those from dghypB() if the $\alpha, \beta, \delta$ parameters of the B type are translated to the corresponding $\gamma, \chi, \psi$ parameters of the QRM type by formulas on pp 79–80 in QRM.

If $\gamma$ is a vector of zeros, the distribution is elliptical and dsmghyp() is utilised in dmghyp(). If $\lambda = (d + 1)/2$, a $d$-dimensional hyperbolic density results. If $\lambda = 1$, the univariate marginals are one-dimensional hyperbolics. If $\lambda = -1/2$, the distribution is Normal Inverse Gaussian (NIG). If $\lambda > 0$ and $\chi = 0$, one obtains a Variance Gamma distribution (VG). If one can define a constant $\nu$ such that $\lambda = (-1/2) * \nu$ and $\chi = \nu$ then one obtains a multivariate skewed-t distribution. See p. 80 of QRM for details.

Examples

old.par <- par(no.readonly = TRUE)
par(mfrow = c(2, 2))
ll <- c(-4, 4)
BiDensPlot(func = dmghyp, xpts = ll, ypts = ll, mu = c(0, 0),
Sigma = equicorr(2, -0.7), lambda = 1, chi = 1, psi = 1,
gamma = c(0, 0))
BiDensPlot(func = dmghyp, type = "contour", xpts = ll, ypts = ll,
mu = c(0, 0), Sigma = equicorr(2, -0.7), lambda = 1,
chi = 1, psi = 1, gamma = c(0, 0))
BiDensPlot(func = dmghyp, xpts = ll, ypts = ll, mu = c(0, 0),
Sigma = equicorr(2, -0.7), lambda = 1, chi = 1, psi = 1, 
gamma = c(0.5, -0.5))

BiDensPlot(func = dmghyp, type = "contour", xpts = 11, ypts = 11, 
mu = c(0, 0), Sigma = equicorr(2, -0.7), lambda = 1, 
chi = 1, psi = 1, gamma = c(0.5, -0.5))

dpar(old.par)

## GIG

### Generalized Inverse Gaussian Distribution

**Description**

Calculates (log) moments of univariate generalized inverse Gaussian (GIG) distribution and generating random variates.

**Usage**

EGIG(lambda, chi, psi, k = 1)
ElogGIG(lambda, chi, psi)
rGIG(n, lambda, chi, psi, envplot = FALSE, messages = FALSE)

**Arguments**

- `chi` numeric, chi parameter.
- `envplot` logical, whether plot of rejection envelope should be created.
- `k` integer, order of moments.
- `lambda` numeric, lambda parameter.
- `messages` logical, whether a message about rejection rate should be returned.
- `n` integer, count of random variates.
- `psi` numeric, psi parameter.

**Details**

Normal variance mixtures are frequently obtained by perturbing the variance component of a normal distribution; here this is done by multiplying the square root of a mixing variable assumed to have a GIG distribution depending upon three parameters \((\lambda, \chi, \psi)\). See p.77 in QRM.

Normal mean-variance mixtures are created from normal variance mixtures by applying another perturbation of the same mixing variable to the mean component of a normal distribution. These perturbations create Generalized Hyperbolic Distributions. See pp. 78–81 in QRM. A description of the GIG is given on page 497 in QRM Book.

**Value**

(log) mean of distribution or vector random variates in case of rGIG().
Generalized Pareto Distribution

Description

Density, quantiles, and cumulative probability of the Generalized Pareto distribution.

Usage

\begin{align*}
pGPD(q, xi, beta = 1) \\
qGPD(p, xi, beta = 1) \\
dGPD(x, xi, beta = 1, log = FALSE) \\
rGPD(n, xi, beta = 1)
\end{align*}

Arguments

- \texttt{beta} numeric, scale parameter.
- \texttt{log} logical, whether log values of density should be returned.
- \texttt{n} integer, count of random variates.
- \texttt{p} vector, probabilities.
- \texttt{q} vector, quantiles.
- \texttt{x} vector, values to evaluate density.
- \texttt{xi} numeric, shape parameter.

Value

numeric, probability (pGPD), quantile (qGPD), density (dGPD) or random variates (rGPD) for the GPD with scale parameter \( \beta \) and shape parameter \( \xi \).

See Also

GEV, POT

Gumbel Distribution

Description

Density, quantiles, and cumulative probability of the Gumbel distribution. The standard Gumbel has \( \mu \) value of 0 and \( \sigma \) value of one.
Usage

\begin{align*}
\text{dGumbel}(x, \mu = 0, \sigma = 1, \text{log} = \text{FALSE}) \\
\text{qGumbel}(p, \mu = 0, \sigma = 1) \\
\text{pGumbel}(q, \mu = 0, \sigma = 1) \\
\text{rGumbel}(n, \mu = 0, \sigma = 1)
\end{align*}

Arguments

<table>
<thead>
<tr>
<th>log</th>
<th>logical, whether log values of density should be returned.</th>
</tr>
</thead>
<tbody>
<tr>
<td>mu</td>
<td>numeric, location parameter.</td>
</tr>
<tr>
<td>n</td>
<td>integer, count of random variates.</td>
</tr>
<tr>
<td>p</td>
<td>vector, probabilities.</td>
</tr>
<tr>
<td>q</td>
<td>vector, quantiles.</td>
</tr>
<tr>
<td>sigma</td>
<td>numeric, scale parameter.</td>
</tr>
<tr>
<td>x</td>
<td>vector, values to evaluate density.</td>
</tr>
</tbody>
</table>

Value

numeric, probability (pGumbel()), quantile (qGumbel()), density (dGumbel()) or random variates (rGumbel()) for the Gumbel distribution with location parameter $\mu$ and scale parameter $\sigma$.

Examples

\begin{align*}
r\text{GumbelSim} &\leftarrow \text{rGumbel}(1000, 1.0, 2.5) \\
\text{quantValue} &\leftarrow 4.5 \\
\text{pGEV}(q = \text{quantValue}, xi = 0, \mu = 1.0, \sigma = 2.5) \\
\text{pGumbel}(q = \text{quantValue}, \mu = 1.0, \sigma = 2.5)
\end{align*}

hsi

\text{Hang Seng Stock Market Index}

Description

The hsi timeSeries dataset provides the daily closing value for the Hang Seng Index from January 1994 to March 2004. In addition, the data set is also made available as a data.frame.

Usage

\begin{align*}
data(hsi) \\
data(hsi.df)
\end{align*}

Examples

\begin{align*}
data(hsi) \\
head(hsi)
\end{align*}
Kendall

Kendall’s Rank Correlation

Description
Calculates Kendall’s rank correlations. The function is a wrapper to cor().

Usage
Kendall(data, ...)

Arguments
data matrix or data.frame.
... ellipsis, arguments are passed down to cor()

Value
matrix

See Also
cor, Spearman

Examples
S <- equicorr(d = 3, rho = 0.5)
data <- rmnorm(1000, Sigma = S)
Kendall(data)

nasdaq

NASDAQ Stock Market Index

Description
The nasdaq timeSeries dataset provides the daily closing value for the NASDAQ index from January 1994 to March 2004. In addition, the data set is also made available as a data.frame.

Usage
data(nasdaq)
data(nasdaq.df)

Examples
data(nasdaq)
head(nasdaq)
Normal Inverse Gaussian and Hyperbolic Distribution

Description

Functions for fitting uni- and multivariate NIG and HYP distribution.

Usage

```r
fit.NH(data, case = c("NIG", "HYP"), symmetric = FALSE,
       se = FALSE, ...)
fit.mNH(data, symmetric = FALSE, case = c("NIG", "HYP"),
        kvalue = NA, nit = 2000, tol = 1e-10, ...)
MCECM.update(data, mix.pars, mu, Sigma, gamma, optpars, optfunc,
              xieval = FALSE, ...)
MCECM.Qfunc(lambda, chi, psi, delta, eta, xi)
EM.update(data, mix.pars, mu, Sigma, gamma, symmetric,
          scaling = TRUE, kvalue = 1)
```

Arguments

- **case**: character, whether NIG or HYP shall be used.
- **chi**: numeric, chi parameter.
- **data**: numeric, data.
- **delta**: numeric, delta parameter.
- **eta**: numeric, eta parameter.
- **kvalue**: numeric, value to which the determinant of the dispersion matrix is constrained.
- **lambda**: numeric, lambda parameter.
- **mix.pars**: vector, values of lambda, chi and psi.
- **mu**: numeric, value of location parameters.
- **nit**: integer, maximum number of iterations.
- **optpars**: vector, parameters to optimize over.
- **optfunc**: function, the function to be optimized.
- **psi**: numeric, pi parameter.
- **scaling**: logical, whether determinant scaling of Sigma shall be fixed.
- **se**: logical, whether standard errors should be calculated.
- **Sigma**: matrix, value of Sigma.
- **symmetric**: logical, whether symmetric case should be fitted.
- **tol**: numeric, tolerance for convergence.
- **gamma**: numeric, value of gamma.
- **xi**: numeric, xi parameter.
- **xieval**: logical, whether log moment xi shall be evaluated.
- **...**: ellipsis, arguments are passed down to `optim()`.
Details

fit.NH(): See pages 78–80 of QRM. Case ‘NIG’ sets $\lambda = -1/2$; case ‘HYP’ sets $\lambda = 1$.
fit.mNH(): Fitting is accomplished by using a variant of the EM algorithm (see pages 81–83 in QRM).
MCECMupdate(): updates estimates of mixing parameters in EM estimation of generalized hyperbolic (see Algorithm 3.14, steps (5) and (6) on page 83 in QRM).
MCECM.Qfunc(): a functional form that must be optimized when fitting members of generalized hyperbolic family with an MCECM algorithm (see function Q2 on page 82 of QRM).
EMupdate(): updates estimates of location ($\mu$), dispersion ($\Sigma$) and skewness ($\gamma$) parameters in EM estimation of multivariate generalized hyperbolic distributions (see pages 81–83 in QRM; in that case $k$ is the determinant of the sample covariance matrix. “EM” is an acronym for for “Expectation-Maximization” type of algorithm used to fit proposed multivariate hyperbolic models to actual data).

Examples

```r
data(DJ)
r <- returns(DJ)
s <- window(r[, "MSFT"], "1993-01-01", "2000-12-31")
mod.NIG <- fit.NH(100 * s, method = "BFGS")
## multivariate
stocks <- c("AXP", "EK", "BA", "C", "KO", "MSFT", "HWP", "INTC", "JPM", "DIS")
ss <- window(r[, stocks], "1993-01-01", "2000-12-31")
fridays <- time(ss)[isWeekday(time(ss), wday = 5)]
ssw <- aggregate(ss, by = fridays, FUN = sum)
mod.mNIG <- fit.mNH(ssw, symmetric = FALSE, case = "NIG")
```

**Nikkei Stock Market Index**

Description

The `níkkei` timeSeries dataset provides the daily closing value for the Nikkei index from January 1994 to March 2004. In addition, the data set is also made available as a `data.frame`.

Usage

```r
data(níkkei)
data(níkkei.df)
```

Examples

```r
data(níkkei)
head(níkkei)
```
Pconstruct

Assemble a Correlation Matrix for ML Copula Fitting

Description
This function converts a vector of values representing the terms of a lower triangular matrix \( A \) with ones on the diagonal and returns the correlation matrix corresponding to the covariance matrix \( AA' \) (see page 235 in QRM).

Usage
Pconstruct(theta)

Arguments
theta vector, elements of a lower triangular matrix \( A \) with ones on the diagonal.

Value
matrix

See Also
link{Pdeconstruct}

Examples
P <- Pconstruct(1:6)
eigen(P)
Pdeconstruct(P)

Pdeconstruct

Disassemble a Correlation Matrix for ML Copula Fitting

Description
This function takes a correlation matrix \( P \) and returns the elements of a lower-triangular matrix \( A \) with ones on the diagonal such that \( P \) is the corelation matrix corresponding to the covariance matrix \( AA' \) (see page 235 in QRM).

Usage
Pdeconstruct(P)

Arguments
P matrix, a correlation matrix
PointProcess

Value

vector

See Also

Pconstruct

Examples

```r
P <- Pconstruct(1:6)
Pdeconstruct(P)
```

---

PointProcess  Point Processes

Description

Functions for estimating point processes.

Usage

```r
extremalPP(data, threshold = NA, nextremes = NA, ...)
unmark(PP)
fit.POT(PP, markdens = "GPD", ...)
fit.sePP(PP, model = c("Hawkes", "ETAS"), mark.influence = TRUE,
        std.errs = FALSE, ...)
fit.seMPP(PP, markdens = "GPD", model = c("Hawkes", "ETAS"),
         mark.influence = TRUE, predictable = FALSE,
         std.errs = FALSE, ...)
stationary.sePP(sePP)
sePP.negloglik(theta, PP, case)
seMPP.negloglik(theta, PP, case, markdens)
volfunction(anytimes, times, marks, theta, model)
  ## S3 method for class 'MPP'
  plot(x, ...)
  ## S3 method for class 'PP'
  plot(x, ...)
  ## S3 method for class 'sePP'
  plot(x, ...)
```

Arguments

- `anytimes` vector, times at which to calculate self-excitement function.
- `data` timeSeries object or vector.
- `case` numeric, indicating Hawkes or ETAS models and whether marks may have an influence on future points.
markdens  character, name of density of mark distribution, currently only "GPD".
mark.influence logical, whether marks of marked point process may influence the self-excitement.
marks    vector, marks associated with point events.
model    character, name of self-exciting model.
nextremes integer, count of upper extremes to be used.
PP       list, a point process object of class PP or MPP.
predictable logical, whether previous events may influence the scaling of mark distribution.
sePP     list, a fitted self-exciting process created with fit.sePP() or a marked self-exciting process created with fit.seMPP().
std.errs  logical, whether standard errors should be computed.
theta     vector, parameters of self-excitement function.
threshold numeric, threshold value.
times     vector, times of point events.
x         list, a (un/marked) point process object of class PP/MPP.
...       ellipsis, arguments passed to plot() or to fit.GPD() for fit.POT() or to nlminb() for functions fit.sePP() and fit.seMPP or to julian() for extremalPP.

Details

extremalPP(): returns a list describing a marked point process (see pages 298-301 of QRM).
fit.POT(): fits the POT (peaks-over-threshold) model to a point process of class PP or MPP. Note that if point process is of class PP, then function simply estimates the rate of a homogeneous Poisson process (see pages 301–305 of QRM).
fit.seMPP(): fits a marked self-exciting process to a point process object of class MPP.
fit.sePP(): fits self-exciting process to a point process object of class PP (unmarked) or MPP (marked).
seMPP.negloglik(): evaluates negative log-likelihood of a marked self-exciting point process model; this objective function will be passed to the optimizing function.
sePP.negloglik(): evaluates negative log-likelihood of a self-exciting point process model (unmarked).
stationary.sePP(): checks a sufficient condition for stationarity of a self-exciting model and gives information about cluster size.
unmark(): strips marks from a marked point process.
volffunction(): calculates a self-excitement function for use in the negloglik methods used in fit.sePP() and fit.seMPP().

Value

The function extremalPP() returns a list describing class MPP (marked point process) consisting of times and magnitudes of threshold exceedances:

times vector of julian day counts (since 1/1/1960) for each exceedance
marks vector of exceedances values (differences between value and threshold at each mark)
The functions \texttt{fit.POT()}, \texttt{fit.seMPP()}, and \texttt{fit.sePP()} return a list containing the fitted model. The \texttt{plot}-methods return invisibly the data for producing these.

\textbf{See Also}

\texttt{GPD}, \texttt{nlminb}

\textbf{Examples}

```r
## Extremal PP
data(sp500)
l <- -returns(sp500)
lw <- window(l, start = "1995-12-31", end = end(l))
mod1 <- extremalPP(lw, ne = 100)
mod1$marks[1:5]
mod1$threshold
mod2a <- fit.sePP(mod1, mark.influence = FALSE, std.errs = TRUE)
mod2b <- fit.seMPP(mod1, mark.influence = FALSE, std.errs = TRUE)
stationary.sePP(mod2b)
mod2c <- fit.POT(mod1, method = "BFGS")
plot(mod1)
plot(unmark(mod1))
plot(mod2a)
```
showRM(object, alpha, RM = c("VaR", "ES"), extend = 2, ci.p = 0.95,
like.num = 50., ...) 
RiskMeasures(out, p) 

Arguments 

alpha numeric, probability level(s).
auto.scale logical, whether plot should be automatically scaled.
ci numeric, probability for asymptotic confidence bands.
ci.p numeric, confidence levels.
data numeric, data vector or timesSeries.
end integer, maximum number of exceedances to be considered.
extend numeric, extension of plotting range.
fineness integer, count of points at which to evaluate the tail estimate.
information character, whether standard errors should be calculated with “observed” or “expected” information. This only applies to maximum likelihood type; for “pwm” type “expected” information is used if possible.
labels logical, whether axes shall be labelled.
like.num integer, count of evaluations of profile likelihood.
type character, estimation by either ML- or PWM type.
models integer, count of consecutive gpd models to be fitted; i.e., the count of different thresholds at which to re-estimate ξ; this many ξ estimates will be plotted.ne integer, count of excesses above the threshold.
extremes integer, count of upper extremes to be used.
object list, returned value from fitting GPD
omit integer, count of upper plotting points to be omitted.
optfunc character, function used for ML-optimization.
option logical, whether "alpha", "xi" (1 / alpha) or "quantile" (a quantile estimate) should be plotted.
out list, returned value from fitting GPD.
p vector, probability levels for risk measures.
reverse logical, plot ordered by increasing threshold or number of extremes.
RM character, risk measure, either "VaR" or "ES"
start integer, lowest number of exceedances to be considered.
table logical, printing of a result table.
threshold numeric, threshold value.
... ellipsis, arguments are passed down to either plot() or optim() or nlminb().
Details

MEplot(): An upward trend in plot shows heavy-tailed behaviour. In particular, a straight line with positive gradient above some threshold is a sign of Pareto behaviour in tail. A downward trend shows thin-tailed behaviour whereas a line with zero gradient shows an exponential tail. Because upper plotting points are the average of a handful of extreme excesses, these may be omitted for a prettier plot.

RiskMeasures(): calculates risk measures (VaR or ES) based on a generalized Pareto model fitted to losses over a high threshold.

xiplot(): creates a plot showing how the estimate of shape varies with threshold or number of extremes.

hillplot(): This plot is usually calculated from the alpha perspective. For a generalized Pareto analysis of heavy-tailed data using the gpd function, it helps to plot the Hill estimates for xi. See pages 286–289 in QRM. Especially note that Example 7.28 suggests the best estimates occur when the threshold is very small, perhaps 0.1 of the sample size (10–50 order statistics in a sample of size 1000). Hence one should NOT be using a 95 percent threshold for Hill estimates.

plotFittedGPDvsEmpiricalExcesses(): Build a graph which plots the GPD fit of excesses over a threshold u and the corresponding empirical distribution function for observed excesses.

See Also

GEV

Examples

data(danish)
plot(danish)
MEplot(danish)
xiplot(danish)
hillPlot(danish, option = "alpha", start = 5, end = 250, p = 0.99)
hillPlot(danish, option = "alpha", start = 5, end = 60, p = 0.99)
plotFittedGPDvsEmpiricalExcesses(danish, nextremes = 109)
plotFittedGPDvsEmpiricalExcesses(danish, threshold = 10)
findthreshold(danish, 50)
mod1 <- fit.GPD(danish, threshold = 10)
RiskMeasures(mod1, c(0.95, 0.99))
plotTail(mod1)
showRM(mod1, 0.99, RM = "VaR", method = "BFGS")
showRM(mod1, 0.99, RM = "ES", method = "BFGS")
mod2 <- fit.GPD(danish, threshold = 10, type = "pwm")
mod3 <- fit.GPD(danish, threshold = 10, optfunc = "nlminb")

QQplot

Description

Constructs a quantile-quantile plot against a given reference distribution.
Usage

QQplot(x, a = 0.5, reference = c("normal", "exp", "student"), ...)

Arguments

x vector, data for QQ-plot.
a numeric, the offset fraction to be used in ppoints(); typically in (0, 1).
reference character, name of reference distribution.
... ellipsis argument, passed down to quantile function of reference distribution.

Details

Special forms like ParetoQQ plots can also be created via this function. E.g., to create a ParetoQQ plot, merely pass log(data) in place of data as the first parameter and use reference = "exp" as the reference distribution. The ParetoQQ plot should provide a linear graph when a log transform of the data is plotted against the exponential distribution.

Value

Produces QQ-plot and returns invisibly a list of (x, y) pairs.

See Also

ppoints

Examples

QQplot(rnorm(1000), reference = "normal")
QQplot(rexp(1000), reference = "exp", rate = 0.3)

QRM-defunct

Defunct Functions in Package QRM

Description

The functions listed below which were contained in the package QRMlib are now defunct. The user is referred to the suggested functions as an alternative.

Details

aggregateMonthlySeries() is defunct. use aggregate() in package timeSeries.
aggregateQuarterlySeries is defunct. use aggregate() in package timeSeries.
aggregateSignalSeries() is defunct. use aggregate() in package timeSeries.
aggregateWeeklySeries() is defunct. use aggregate() in package timeSeries.
besselM3() is defunct. use besselK() in package base.
ConvertDFToTimeSeries() is defunct. use timeSeries() in package timeSeries.
CovToCor() is defunct. use cov2cor() in package stats.
fit.Archcopula2d() is defunct. use fit.AC().
fit.GPDb() is defunct. use fit.GPD().
fit.tcopula.rank() is defunct. use fit.tcopula().
hessb() is defunct. use hessian() in package numDeriv.
kurtosisSPlus() is defunct. use kurtosis() in package timeDate.
lbeta() is defunct. use lbeta() in package base.
mk.returns() is defunct. use returnSeries() in package timeSeries.
plotMultiTS() is defunct. use plot() in package timeSeries.
psifunc() is defunct. use psi() in package gsl.
signalSeries() is defunct. use series() in package timeSeries.
symmetrize() is defunct. use forceSymmetric() in package Matrix.

---

**smi**  
*Swiss Market Index*

**Description**

The smi timeSeries dataset provides the daily closing value for the Swiss Market index from November 1990 to March 2004. In addition, the data set is also made available as a data.frame.

**Usage**

data(smi)
data(smi.df)

**Examples**
data(smi)  
head(smi)

---

**sp500**  
*Standard and Poors 500 Index*

**Description**

The sp500 timeSeries dataset provides the daily closing value for the S and P 500 Index from January 1980 to March 2004. In addition, the data set is also made available as a data.frame.

**Usage**

data(dji)
data(dji.df)

**Examples**
data(sp500)  
head(sp500)
spdata  
*Standard and Poors Default Data*

**Description**

The spdata timeSeries dataset contains default data for A, BBB, BB, B and C-rated companies for the years 1981 to 2000. In addition, the data set is also made available as a data.frame.

**Usage**

```r
data(spdata)
data(spdata.df)
```

**Source**

Standard and Poors Credit Monitor

**Examples**

```r
data(spdata)
head(spdata)
```

---

spdata.raw  
*Standard and Poors Default Data*

**Description**

The spdata.raw timeSeries contains default data for A, BBB, BB, B and C-rated companies for the years 1981 to 2000.

**Usage**

```r
data(spdata.raw)
data(spdata.raw.df)
```

**Source**

Standard & Poors Credit Monitor

**Examples**

```r
data(spdata.raw)
head(spdata.raw)
```
Spearman

Spearman’s Rank Correlation

Description

Calculates Sperman’s rank correlations. The function is a wrapper to cor().

Usage

Spearman(data, ...)

Arguments

data matrix or data.frame

... ellipsis, arguments are passed down to cor()

Value

matrix

See Also

cor, Kendall

Examples

S <- equicorr(d = 3, rho = 0.5)
data <- rmnorm(1000, Sigma = S)
Spearman(data)

Student

Student’s t Distribution

Description

Functions for evaluating density, fitting and random variates of multivariate Student’s t distribution and routines for quantiles and fitting of univariate distribution.

Usage

dmt(x, df, mu, Sigma, log = FALSE)
rmt(n, df = 4, mu = 0, Sigma)
qst(p, mu = 0, sd = 1, df, scale = FALSE)
fit.st(data, ...)
fit.mst(data, nit = 2000, tol = 1e-10, ...)
Arguments

- **x** matrix, dimension \( n \times d \); density is evaluated for each row.
- **df** numeric, degrees of freedom.
- **mu** numeric, location parameters.
- **Sigma** matrix, dispersion matrix.
- **log** logical, returning log density values.
- **data** numeric, data used for uni- and multivariate fitting.
- **nit** integer, number of iterations of EM-type algorithm.
- **tol** numeric, tolerance of improvement for stopping iteration.
- **p** numeric, probability.
- **sd** numeric, scale parameters.
- **scale** logical, scaling Student’s t distribution.
- **n** integer, count of random variates.
- **...** ellipsis, arguments are passed down to `optim()` in `fit.st()` and to `MCECMupdate()` in `fit.mst()`.

See Also

`link{EMupdate}`, `link{MCECMupdate}`, and `link{MCEM.Qfunc}`

Examples

```r
BiDensPlot(func = dmt, xpts = c(-4, 4), ypts = c(-4, 4), mu = c(0, 0),
           Sigma = equicorr(2, -0.7), df = 4)
## Quantiles of univariate Student’s t
p <- c(0.90, 0.95)
s <- 0.2 * 10000/sqrt(250)
qst(p, sd = s, df = 4, scale = TRUE)
## Fitting multivariate Student’s t
Sigma <- diag(c(3, 4, 5)) %*% equicorr(3, -0.6) %*% diag(c(3, 4, 5))
mu <- c(1, 2, 3)
tdata <- rmt(1000, 4, mu = mu, Sigma = Sigma)
mod1 <- fit.mst(tdata, method = "BFGS")
## DJ data
data(DJ)
r <- returns(DJ)
s <- window(r[, "MSFT"], "1993-01-01", "2000-12-31")
mod.t1 <- fit.st(100 * s)
stocks <- c("AXP", "EK", "BA", "C", "KO", "MSFT",
            "HWP", "INTC", "JPM", "DIS")
ss <- window(r[, stocks], "1993-01-01", "2000-12-31")
fridays <- time(ss)[isWeekday(time(ss), wday = 5)]
ssw <- aggregate(ss, by = fridays, FUN = sum)
mod.t2 <- fit.mst(ssw, method = "BFGS")
```
**xdax**

---

**Xetra DAX German Index**

**Description**

The `xdax` timeSeries dataset provides the daily closing value for the German Xetra DAX index from January 1994 to March 2004. In addition, the data set is also made available as a `data.frame`.

**Usage**

```r
data(xdax)
data(xdax.df)
```

**Examples**

```r
data(xdax)
head(xdax)
```
Index

*Topic **array**
eigenmeth, 15
equicorr, 16
Pconstruct, 28
Pdeconstruct, 28

*Topic **datasets**
cac40, 4
danish, 13
DJ, 13
dji, 14
ftse100, 17
FXGBP.RAW, 18
hsi, 24
nasdaq, 25
nikkei, 27
smi, 35
sp500, 35
spdata, 36
spdata.raw, 36
xdax, 39

*Topic **data**
QRM-defunct, 34

*Topic **distributions**
POT, 31

*Topic **distribution**
CopulaAC, 4
CopulaGauss, 7
CopulaStudent, 8
ES, 16
Gauss, 18
GEV, 19
GHYP, 20
GIG, 22
GPD, 23
Gumbel, 23
NH, 26
Student, 37

*Topic **hplot**
BiDensPlot, 3

QQplot, 33

*Topic **models**
Credit, 9
PointProcess, 29

*Topic **multivariate**
Kendall, 25
Spearman, 37

*Topic **utilities**
edf, 14
aggregateMonthlySeries (QRM-defunct), 34
aggregateQuarterlySeries (QRM-defunct), 34
aggregateSignalSeries (QRM-defunct), 34
aggregateWeeklySeries (QRM-defunct), 34
besselM3 (QRM-defunct), 34
BiDensPlot, 3
cac40, 4
cal.beta (Credit), 9
cal.claytonmix (Credit), 9
cal.probitnorm (Credit), 9
ConvertDFToTimeSeries (QRM-defunct), 34
CopulaAC, 4
CopulaGauss, 7
CopulaStudent, 8
cor, 25, 37
CovToCor (QRM-defunct), 34
Credit, 9
danish, 13
dclaytonmix (Credit), 9
dcopula.AC (CopulaAC), 4
dcopula.clayton (CopulaAC), 4
dcopula.gauss (CopulaGauss), 7
dcopula.gumbel (CopulaAC), 4
dcopula.t (CopulaStudent), 8
dGEV (GEV), 19
dghyp (GHYP), 20
<table>
<thead>
<tr>
<th>Function/Method</th>
<th>Page(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>dghypB (GHYP)</td>
<td>20</td>
</tr>
<tr>
<td>dGPD (GPD)</td>
<td>23</td>
</tr>
<tr>
<td>dGumbel (Gumbel)</td>
<td>23</td>
</tr>
<tr>
<td>DJ</td>
<td>13</td>
</tr>
<tr>
<td>dji</td>
<td>14</td>
</tr>
<tr>
<td>dmghyp (GHYP)</td>
<td>20</td>
</tr>
<tr>
<td>dmnorm (Gauss)</td>
<td>18</td>
</tr>
<tr>
<td>dmt (Student)</td>
<td>37</td>
</tr>
<tr>
<td>dprobitnorm (Credit)</td>
<td>9</td>
</tr>
<tr>
<td>dsmghyp (GHYP)</td>
<td>20</td>
</tr>
<tr>
<td>edf</td>
<td>14</td>
</tr>
<tr>
<td>EGiG (GIG)</td>
<td>22</td>
</tr>
<tr>
<td>eigenmeth</td>
<td>15</td>
</tr>
<tr>
<td>ElogGiG (GIG)</td>
<td>22</td>
</tr>
<tr>
<td>EMupdate (NH)</td>
<td>26</td>
</tr>
<tr>
<td>equicorr</td>
<td>16</td>
</tr>
<tr>
<td>ESnorm (ES)</td>
<td>16</td>
</tr>
<tr>
<td>ESst (ES)</td>
<td>16</td>
</tr>
<tr>
<td>extremalPP (PointProcess)</td>
<td>29</td>
</tr>
<tr>
<td>findthreshold (POT)</td>
<td>31</td>
</tr>
<tr>
<td>fit.AC (CopulaAC)</td>
<td>4</td>
</tr>
<tr>
<td>fit.Archipocula2d (QRM-defunct)</td>
<td>34</td>
</tr>
<tr>
<td>fit.binomial (Credit)</td>
<td>9</td>
</tr>
<tr>
<td>fit.binomialBeta (Credit)</td>
<td>9</td>
</tr>
<tr>
<td>fit.binomialLogitnorm (Credit)</td>
<td>9</td>
</tr>
<tr>
<td>fit.binomialProbitnorm (Credit)</td>
<td>9</td>
</tr>
<tr>
<td>fit.gausscopula (CopulaGauss)</td>
<td>7</td>
</tr>
<tr>
<td>fit.GEV (GEV)</td>
<td>19</td>
</tr>
<tr>
<td>fit.GPD (POT)</td>
<td>31</td>
</tr>
<tr>
<td>fit.GPDb (QRM-defunct)</td>
<td>34</td>
</tr>
<tr>
<td>fit.mNh (NH)</td>
<td>26</td>
</tr>
<tr>
<td>fit.mst (Student)</td>
<td>37</td>
</tr>
<tr>
<td>fit.NH (NH)</td>
<td>26</td>
</tr>
<tr>
<td>fit.norm (Gauss)</td>
<td>18</td>
</tr>
<tr>
<td>fit.POT (PointProcess)</td>
<td>29</td>
</tr>
<tr>
<td>fit.seMPP (PointProcess)</td>
<td>29</td>
</tr>
<tr>
<td>fit.sePP (PointProcess)</td>
<td>29</td>
</tr>
<tr>
<td>fit.st (Student)</td>
<td>37</td>
</tr>
<tr>
<td>fit.tcopula (CopulaStudent)</td>
<td>8</td>
</tr>
<tr>
<td>fit.tcopula.rank (QRM-defunct)</td>
<td>34</td>
</tr>
<tr>
<td>ftse100</td>
<td>17</td>
</tr>
<tr>
<td>FXGBP (FXGBP, RAW)</td>
<td>18</td>
</tr>
<tr>
<td>FXGBP, RAW</td>
<td>18</td>
</tr>
<tr>
<td>Gauss</td>
<td>18</td>
</tr>
<tr>
<td>GEV, 19, 23, 33</td>
<td></td>
</tr>
<tr>
<td>GHYP, 20</td>
<td></td>
</tr>
<tr>
<td>GIG, 22</td>
<td></td>
</tr>
<tr>
<td>GPD, 20, 23, 31</td>
<td></td>
</tr>
<tr>
<td>Gumbel, 23</td>
<td></td>
</tr>
<tr>
<td>hessb (QRM-defunct)</td>
<td>34</td>
</tr>
<tr>
<td>hillPlot (POT)</td>
<td>31</td>
</tr>
<tr>
<td>hsi, 24</td>
<td></td>
</tr>
<tr>
<td>jointnormalTest (Gauss)</td>
<td>18</td>
</tr>
<tr>
<td>Kendall, 25, 37</td>
<td></td>
</tr>
<tr>
<td>kurtosisSPlus (QRM-defunct)</td>
<td>34</td>
</tr>
<tr>
<td>lbeta (QRM-defunct)</td>
<td>34</td>
</tr>
<tr>
<td>MardiaTest (Gauss)</td>
<td>18</td>
</tr>
<tr>
<td>MCECM.Qfunc (NH)</td>
<td>26</td>
</tr>
<tr>
<td>MCECMupdate (NH)</td>
<td>26</td>
</tr>
<tr>
<td>MEplot (POT)</td>
<td>31</td>
</tr>
<tr>
<td>mk.returns (QRM-defunct)</td>
<td>34</td>
</tr>
<tr>
<td>momest (Credit)</td>
<td>9</td>
</tr>
<tr>
<td>nasdaq, 25</td>
<td></td>
</tr>
<tr>
<td>NH, 26</td>
<td></td>
</tr>
<tr>
<td>nikkei, 27</td>
<td></td>
</tr>
<tr>
<td>nlminb, 6, 7, 9, 31</td>
<td></td>
</tr>
<tr>
<td>p Claytonmix (Credit)</td>
<td>9</td>
</tr>
<tr>
<td>Pconstruct, 28, 29</td>
<td></td>
</tr>
<tr>
<td>Pdeconstruct</td>
<td>28</td>
</tr>
<tr>
<td>pGEV (GEV)</td>
<td>19</td>
</tr>
<tr>
<td>pGPD (GPD)</td>
<td>23</td>
</tr>
<tr>
<td>pGumbel (Gumbel)</td>
<td>23</td>
</tr>
<tr>
<td>plotMPP (PointProcess)</td>
<td>29</td>
</tr>
<tr>
<td>plot.PP (PointProcess)</td>
<td>29</td>
</tr>
<tr>
<td>plot.sePP (PointProcess)</td>
<td>29</td>
</tr>
<tr>
<td>plotFittedGPDvsEmpiricalExcesses (POT)</td>
<td>31</td>
</tr>
<tr>
<td>plotMultiTS (QRM-defunct)</td>
<td>34</td>
</tr>
<tr>
<td>plotTail (POT)</td>
<td>31</td>
</tr>
<tr>
<td>PointProcess, 29</td>
<td></td>
</tr>
<tr>
<td>POT, 23, 31</td>
<td></td>
</tr>
<tr>
<td>ppoints, 34</td>
<td></td>
</tr>
<tr>
<td>pprobitnorm (Credit)</td>
<td>9</td>
</tr>
<tr>
<td>psifunc (QRM-defunct)</td>
<td>34</td>
</tr>
<tr>
<td>qGEV (GEV)</td>
<td>19</td>
</tr>
<tr>
<td>qGPD (GPD)</td>
<td>23</td>
</tr>
<tr>
<td>qGumbel (Gumbel)</td>
<td>23</td>
</tr>
</tbody>
</table>
INDEX

QQplot, 33
QRM-defunct, 34
QRM-package, 2
qst (Student), 37

rAC (CopulaAC), 4
rACp (CopulaAC), 4
rBB9Mix (CopulaAC), 4
rbinomial.mixture (Credit), 9
rclaytonmix (Credit), 9
rcopula.clayton (CopulaAC), 4
rcopula.frank (CopulaAC), 4
rcopula.gauss (CopulaGauss), 7
rcopula.gumbel (CopulaAC), 4
rcopula.Gumbel2Gp (CopulaAC), 4
rcopula.GumbelNested (CopulaAC), 4
rcopula.t (CopulaStudent), 8
rFrankMix (CopulaAC), 4
rGEV (GEV), 19
rghyp (GHYP), 20
rghypB (GHYP), 20
rGIG (GIG), 22
rGPD (GPD), 23
rGumbel (Gumbel), 23
RiskMeasures (POT), 31
rlogitnorm (Credit), 9
rmghyp (GHYP), 20
rmnorm (Gauss), 18
rmt (Student), 37
rprobitnorm (Credit), 9
rstable (CopulaAC), 4
rtcopulamix (Credit), 9

seMPP.negloglik (PointProcess), 29
sePP.negloglik (PointProcess), 29
showRM (POT), 31
signalSeries (QRM-defunct), 34
smi, 35
sp500, 35
spdata, 36
spdata.raw, 36
Spearman, 25, 37
stationary.sePP (PointProcess), 29
Student, 37
symmetrize (QRM-defunct), 34

unmark (PointProcess), 29

volfunction (PointProcess), 29

xdax, 39
xiplot (POT), 31