

QRMLib

April 4, 2007

R topics documented:

QRMLib-package	3
QRMBook-workspace	5
storeDataInWorkspace	6
profileLoadLibrary	7
aggregateMonthlySeries	8
aggregateQuarterlySeries	10
aggregateSignalSeries	11
aggregateWeeklySeries	13
besselM3	14
beta (stats)	15
BiDensPlot	16
cac40	17
cal.beta	17
cal.claytonmix	18
cal.probitnorm	19
claytonmix	20
CovToCor	21
Danish	22
dcopula.clayton	22
dcopula.gauss	23
dcopula.gumbel	24
dcopula.t	25
DJComponents	26
dji	26
dmnorm	26
dmt	27
dsmghyp	27
edf	29
EGIG	29
eigenmeth	30
ElogGIG	31
EMupdate	31
equicorr	32
ESnorm	33
ESst	34
extremalPP	35
findthreshold	36

fit.Archcopula2d	36
fit.binomial	37
fit.binomialBeta	38
fit.binomialLogitnorm	39
fit.binomialProbitnorm	40
fit.gausscopula	41
fit.GEV	42
fit.GPD	43
fit.GPDb	44
fit.mNH	45
fit.mst	46
fit.NH	47
fit.norm	48
fit.POT	49
fit.seMPP	50
fit.sePP	51
fit.st	52
fit.tcopula.rank	53
fit.tcopula	54
ftse100	55
FXGBP.RAW	55
GEV	55
ghyp	56
ghypB	57
GPD	59
Gumbel	60
hessb	60
hillPlot	61
hsi	62
jointnormalTest	63
Kendall	64
kurtosisSPlus	64
lbeta	65
MardiaTest	66
MCECM.Qfunc	66
MCECMupdate	67
MEplot	68
mghyp	69
mk.its.exceedances.tS	70
mk.its.exceedances.vector	72
mk.oldts	73
mk.returns	74
momest	74
nasdaq	75
nikkei	76
Pconstruct	76
Pdeconstruct	77
plot.MPP	77
plot.PP	78
plot.sePP	79
plot.timeSeries	80
plot.timeSeriesIts	81

plotFittedGPDvsEmpiricalExcesses	82
plotTail	83
probitnorm	83
psifunc	85
QQplot	85
qst	86
rAC	87
rBB9Mix	87
rbinomial.mixture	88
rcopula.AGumbel	89
rcopula.clayton	90
rcopula.frank	91
rcopula.gauss	91
rcopula.gumbel	92
rcopula.Gumbel2Gp	93
rcopula.GumbelNested	94
rcopula.t	94
rFrankMix	95
rGIG	96
RiskMeasures	97
rlogitnorm	98
rmnorm	99
rmt	100
rstable	101
rtcopulamix	102
seMPP.negloglik	103
sePP.negloglik	104
showRM	105
signalSeries	106
smi	107
sp500	107
spdata.raw	107
spdata	108
Spearman	109
stationary.sePP	110
symmetrize	111
timeSeriesClass	111
TimeSeriesClassRMetrics	112
unmark	114
volfunction	115
xdax	116
xiplot	116

QRMLib-package	<i>This package provides R-language code to investigate concepts in a Quantitative Risk Management book for those users without access to S-Plus.</i>
----------------	---

Description

This is a free R-language translation of the S-Plus library (QRMLib) designed to accompany the book *Quantitative Risk Management: Concepts, Techniques and Tools* by Alexander J. McNeil, Rüdiger Frey and Paul Embrechts. It was built by Scott Ulman (scottulman@hotmail.com). A separate S-Plus version of the library can be downloaded from Alexander McNeil's URL.

Details

Package:	QRMLib
Type:	Package
Version:	1.4
Date:	2007-04-08
Depends:	R(>= 2.4.1),fCalendar,methods,fEcofin, mvtnorm, chron,its,Hmisc
License:	GPL version 2 or newer
URL:	http://www.math.ethz.ch/mcneil/book/qrmllib.html
Packaged:	April 8, 2007
Built:	R 2.4.1; i386-pc-mingw32; 2007-04-08 12:00:00; windows

The package provides an entire library of methods to investigate concepts associated with QRM, including Market Risk, Credit Risk, and Operational Risk, as developed in the textbook. Additionally, it contains a set of chapter scripts which can be used to build many of the graphs and tables in the text. Under the library folder, look for folders Chap2-Chap8 which contain the scripts.

Note

The original S-Plus data files **cac40**, **danish**, **DJ**, **dji**, **ftse100**, **FXGBPRAW**, **hsi**, **nasdaq**, **nikkei**, **smi**, **sp500**, **xdax** are all S-Plus 'timeSeries' object files.

Unfortunately, R-language has several different time-series classes, none of which coincides with the S-Plus version. The R-Metrics' class 'timeSeries' (contained in library fCalendar) is the closest to an S-Plus timeSeries.

Hence data files built in this R-language translation are all R-Metrics 'timeSeries' types. This means you must load the fCalendar library in order to use the R-data files. The setup program should require automatic loading of fCalendar when you open QRMLib.

To automatically load the QRMLib package (and the fCalendar package), see [profileLoadLibrary](#)
 To automatically load the data files and save them in the current workspace, see [storeDataInWorkspace](#)

Author(s)

S-Plus Original by Alexander McNeil; R-language port by Scott Ulman

Maintainer: Scott Ulman <scottulman@hotmail.com> for R-language version

References

Quantitative Risk Management: Concepts, Techniques and Tools by Alexander J. McNeil, Rüdiger Frey and Paul Embrechts
Princeton Press, 2005

See Also

[QRMBook-workspace](#) and [storeDataInWorkspace](#) and [profileLoadLibrary](#)

QRMBook-workspace *How to Build a QRMBook Workspace in R to Use QRMLib*

Description

Follow these instructions to build a QRMBook workspace in R where you can run the book's scripts which build most of the graphics plots and tables in the book *Quantitative Risk Management: Concepts, Techniques and Tools* by Alexander J. McNeil, Rüdiger Frey and Paul Embrechts.

The QRMLib contains scripts which explain topics in most QRM Book chapters.

The folders containing the scripts are named

"C:\Program Files\R\R-2.4.1\library\QRMLib\Chap2)", "...Chap3", etc.

You may open these scripts from within R by choosing *File | Open Script* from the R-menu and then moving to the appropriate Chapter script for the QRM Book. Many chapters contain multiple scripts.

Details

Instructions to Build the QRMBook workspace

The following example assumes you are using R version R-2.4.1. If you are using a later version, substitute your version number in the following instructions.

0. Be sure you have R closed.
1. Using MyComputer or Explorer Create a QRMBook subfolder beneath the folder *C:\Program Files\R\R-2.4.1\Users*.
2. Right-click the desktop and choose *New | Shortcut* from the menu.
3. Copy the following line (including quotation marks) into your clipboard:
"C:\Program Files\R\R-2.4.1\bin\Rgui.exe"
and paste the line into the box labeled "Type the location of the item"
4. Click the Next> button.
5. Type *QRMBook* (without any quotation marks) into the box labeled "Type a name of this shortcut". Then click the Finish button.

6. Find the shortcut you just created on your desktop. It will be labeled “QRMBook”. Right-click the icon for the shortcut and choose ‘Properties’.
7. The ‘Start in’ box says “C:\Program Files\R\R-2.4.1\bin”. Modify it to read “C:\Program Files\R\R-2.4.1\Users\QRMBook” (be sure to include the quotation marks). Then click OK.

Note

You may now launch the QRMBook workspace by double-clicking the newly-created desktop icon. This will open R with a workspace pointing to ‘...Users\QRMBook’. However, there are still two problems with the workspace:

1. You want to avoid having to load the QRMlib each time you open the workspace. See [profileLoadLibrary](#) to resolve this issue
2. You want to use data without issuing the command *data(filename)* each time you open the workspace. See [storeDataInWorkspace](#) to resolve this issue.

See Also

[profileLoadLibrary](#)
[storeDataInWorkspace](#)

storeDataInWorkspace

How to Store Data in a QRMBook Workspace

Description

Data files must be loaded into your workspace before they can be used by scripts. The appropriate command to load data into a workspace is *data(filename)* where filename is the name of one of the data files WITHOUT its R extension. Hence use *data(sp500)* to load the data from the file sp500.R into the workspace

Details

The scripts in the QRM book use data included with the installation. The following data files are located at *C:\Program Files\R\R-2.4.1\Library\QRMlib\data* subfolder. If you examine that folder you may see the data files are compressed in a file named Rdata.zip. You may extract the data into the folder if you wish to see the names of each separate data file by using WinZip or PKZip. The data files include: cac40.R, danish.R, DJ.R, dji.R, ftse100.R, FXGBP.RAW.R, hsi.R, nasdaq.R, nikkei.R, smi.R, sp500.R, spdata.R, spdata.raw.R, and xdax.R.

Note

When you exit the R program, you will be asked whether to save the current workspace environment. If you choose to do so, the data files which you have opened via *data(filename)* calls will be saved in the workspace so you don't need to execute any subsequent *data(filename)* calls to get previously-loaded data. If you do not save the workspace, you must execute *data(filename)* each time you open a script in QRMBook workspace.

See Also

[profileLoadLibrary](#)
[QRMBook-workspace](#)

profileLoadLibrary *Build .Rprofile File to Load QRM Library in QRMBook Workspace*

Description

The QRMLib package (QRMLib.dll) must be loaded into your QRMBook workspace before its functions can be used by scripts.

The appropriate command to load a package into a workspace is

library(QRMLib)

It will be more convenient for you to add a “.Rprofile” file to your QRMBook workspace than to invoke the *library(QRMLib)* command each time you start up. By adding *.Rprofile* to your workspace, you will eliminate the need to load the library each time you run the workspace.

See details below for two ways to install a .Rprofile file into your workspace.

Details

The installation program for QRMLib placed an .Rprofile file in the ...\\library\\QRMLib\\inst\\Chap2 folder. There is also a useful *README.txt* in the ...\\library\\QRMLib folder.

In the Windows Explorer, merely copy the *.Rprofile* file from the QRM library *Chap2* folder into the QRMBook workspace you previously created to run scripts (see [QRMBook-workspace](#)). Once the *.Rprofile* file exists in your QRMBook workspace, the QRMLib will automatically be loaded into the workspace.

Alternatively, you can build an *.Rprofile* file in your QRMBook folder using Notepad or some other text editor. Just perform the following steps:

0. Close your R-workspace if it is open.

1. Copy the next nine lines of code (starting with *.First* and ending with *}*) into the clipboard.

```
.First <- function()  
{  
  library(QRMLib)  
}
```

```
.Last <- function()  
{
```

```
detach(package:QRMLib)
}
```

2. Open Notepad: left-click the Start button, choose Run and type *notepad* into the box. We will try to save a file named “.Rprofile”. Note the entire file name is an extension with no prefix. I.e. there will be no letters prior to the ‘.’ and the type of file is an “Rprofile” type spelled with a capital R followed by all small letters.

3. Paste the copied code into Notepad.

4. In Notepad, choose *File | Save As* from the menu.

5. In the resulting box, click the *Save as Type* drop-down box and choose *All Files*. (We are NOT saving as a .txt type.)

6. Paste the path

“C:\Program Files\R\R-2.4.1\users\QRMBook\.Rprofile”

into the File name box. Be sure to spell *.Rprofile* exactly as shown since R uses case sensitivity in opening files even though Windows does not.

7. Click the *Save* button.

You may now open your QRMBook workspace and the console should show that the QRMLib, fCalendar, fEcofin, mvtnorm, chron, its, and Hmisc libraries have been loaded.

Note

When you exit the R program, you will be asked whether to save the current workspace environment. If you choose to do so, the data files which you have opened via `data(filename)` calls will be saved in the workspace so you don’t need to execute any subsequent `data(filename)` calls to get previously-loaded data. If you do not save the workspace, you must execute `data(filename)` each time you open the QRMBook workspace.

See Also

[QRMBook-workspace](#)
[storeDataInWorkspace](#)

```
aggregateMonthlySeries
      aggregateMonthlySeries() method
```

Description

This is one of several substitutes for the S-Plus language method `aggregateSeries(timeseries, FUN=max, mean, colAvg, colSums,..., by=weeks, months, quarters,...)`.

The R-language **aggregateMonthlySeries()** function allows the user to calculate a less granular timeseries (monthly) from a daily time series by using a statistic like the max, mean, sum, etc. Note the R-methods do NOT contain a **by=“months”** parameter so the R-language user must select either the **aggregateWeeklySeries** method, the **aggregateMonthlySeries()** method, or the **aggregateQuarterlySeries()** method to get the desired result.

Usage

```
aggregateMonthlySeries(timeseries, FUNC = colSums)
```

Arguments

timeseries	a (usually) daily timeSeries (R-Metrics type from fCalendar) from which the user wants to extract a monthly maximum (or monthly mean) timeSeries
FUNC	The name of a function to use in aggregating the data. For example the max, mean, min, etc. The default is colSums.

Details

For example, the user might want to create a series of monthly **colSums** returns from a daily time series of returns. Alternatively, (s)he might want the quarterly or weekly **mean** series. In either case, a less granular set of quarterly/monthly/weekly values is calculated from a daily timeSeries object. Unfortunately, the R-Metrics package has not yet implemented an R-version of the S-Plus aggregateSeries() method.

The **aggregateWeeklySeries()**, **aggregateMonthlySeries()**, and the **aggregateQuarterlySeries()** are interim functions developed to convert daily timeSeries to weekly, monthly, or quarterly timeSeries objects via a statistic like the max, mean, colAvg, or ColSums.

These functions exist in the functionsUtility.R file of the library.

Value

A monthly timeSeries object characterized by some statistic like mean, max, min of the daily series over a month. The positions attribute (dates <- rseries@positions) of the new time series will be the LAST DAYS OF THE RESPECTIVE MONTHS for the timeSeries object.

Author(s)

documentation by Scott Ulman for R-language distribution

See Also

[aggregateWeeklySeries](#), [aggregateQuarterlySeries](#)

Examples

```
#load nasdaq data set:
data(nasdaq);
#Create minus daily return series:
nreturns <- -mk.returns(nasdaq);
#convert to monthly series using max value from each month
#(rather than colSums):
monthly.maxima <- aggregateMonthlySeries(nreturns, FUNC=max);
## Not run:
Ret.DJ <- mk.returns(DJ);
#Choose only 10 of the 30 stocks:
selection1 <- c("AXP", "EK", "BA", "C", "KO", "MSFT", "HWP",
               "INTC", "JPM", "DIS");
partialDJ30dailyTS <- Ret.DJ[,selection1];
partialDJ30daily <- cutSeries(partialDJ30dailyTS, from="1992-12-31",
                             to="2000-12-31");
partialDJ30monthlyTS <- aggregateMonthlySeries(partialDJ30daily,
```

```

                                FUNC= colSums);
## End(Not run)

```

```

aggregateQuarterlySeries
      aggregateQuarterlySeries() method

```

Description

This is one of several substitutes for the S-Plus language method *aggregateSeries(timeseries, FUN=max, mean, colAvg, colSums,...,by=weeks,months,quarters.....)*.

The R-language **aggregateQuarterlySeries()** function allows the user to calculate a less granular timeseries (monthly) from a daily time series by using a statistic like the max, mean, sum, colSums, etc. Note the R-methods do NOT contain a **by="quarters"** parameter so the R-language user must select either the the **aggregateWeeklySeries** method, the **aggregateMonthlySeries()** method, or the **aggregateQuarterlySeries()** method to get the desired result.

Usage

```
aggregateQuarterlySeries(timeseries, FUNC = colSums)
```

Arguments

<code>timeseries</code>	a (usually) daily timeSeries (R-Metrics type from fCalendar) from which the user wants to extract a quarterly colSums (or quarterly mean) timeSeries
<code>FUNC</code>	The name of a function to use in aggregating the data. For example the max, mean, min, etc. Default is 'colSums'.

Details

For example, the user might want to create a series of quarterly **colSums** returns from a daily time series of returns. Alternatively, (s)he might want the quarterly **mean** series. In either case, a less granular (quarterly) set of values is calculated from a daily timeSeries object. Unfortunately, the R-Metrics package has not yet implemented an R-version of the S-Plus *aggregateSeries()* method.

The **aggregateWeeklySeries()**, **aggregateMonthlySeries()**, and the **aggregateQuarterlySeries()** are interim functions developed to convert daily timeSeries to weekly, monthly, or quarterly timeSeries objects via a statistic like the max, mean, colAvg, or ColSums.

These functions exist in the functionsUtility.R file of the library.

Value

A quarterly timeSeries object characterized by some statistic like mean, max, min of the daily series over a quarter. The positions attribute (`dates <- rseries@positions`) of the new time series will be the LAST DAYS OF THE RESPECTIVE QUARTERS for the timeSeries object.

Author(s)

documentation by Scott Ulman for R-language distribution

See Also

[aggregateWeeklySeries](#), [aggregateMonthlySeries](#)

Examples

```
#load nasdaq data set:
data(nasdaq);
#Create daily return series:
nreturns <- -mk.returns(nasdaq)
#convert to quarterly series using maximum value from each quarter:
quarterly.maxima <- aggregateQuarterlySeries(nreturns, FUNC=max);
## Not run:
Ret.DJ <- mk.returns(DJ);
#Choose only 10 of the 30 stocks:
selection1 <- c("AXP", "EK", "BA", "C", "KO", "MSFT", "HWP",
               "INTC", "JPM", "DIS");
partialDJ30dailyTS <- Ret.DJ[,selection1];
partialDJ30daily <- cut(partialDJ30dailyTS, from="1992-12-31",
                       to="2000-12-31");
partialDJ30quarterlyTS <- aggregateQuarterlySeries(partialDJ30daily,
                                                    FUNC= colSums);
## End(Not run)
```

aggregateSignalSeries

aggregateSignalSeries() method

Description

This is a substitute for the S-Plus language method
aggregateSeries(signalSeries, FUN=max, mean, colAvg,...,
by=90,...).

Usage

```
aggregateSignalSeries(x, pos, AGGFUNC, together = FALSE,
                      drop.empty = TRUE, include.ends = FALSE, adj, offset, colnames, by)
```

Arguments

<code>x</code>	The data series to which the AGGFUNC will be applied
<code>pos</code>	a numeric sequence describing the respective positions of each element in the data set
<code>AGGFUNC</code>	the function to be applied to the data set <code>x</code>
<code>together</code>	if TRUE, pass all columns of <code>x</code> together into AGGFUNC; default is to pass each column separately into AGGFUNC for each aggregation block.
<code>drop.empty</code>	logical value telling whether or not to drop aggregation blocks with no positions to aggregate from the output (default is to drop them)
<code>include.ends</code>	logical value telling whether to include positions before the first given aggregation block and after the last in the first/last blocks; default would not include those positions in the output at all.

adj	if provided, adjust the positions of the output series so that they lie a fraction adj towards the blocks ending position; default is to use the lower end of the block for the output position. 0.5 would use the center of the aggregation block for the output position, and 1.0 would use the upper end of the block.
offset	as an alternative to adj, you can provide a constant offset to add to the lower end of the aggregation block to get the output series positions.
colnames	new column names for the output series. Default is to use the same names as the input series if the output series has the same width
by	The number of positions to include for each function application. For example by=90 implies the function will be applied to successive groups of 90 data items.

Details

Input a signalSeries object as parameter x. Input an a function (AGGFUNC) to apply to aggregate data into many smaller subsamples. Use either the 'pos' or 'by' parameter to indicated how to aggregate the data. E.g. 'by=90' will chop the data into separate segments of length 90. The AGGFUNC will then be applied to each aggregation (segment). The R-language **aggregateSignalSeries()** function allows the use of a function evaluation (like Pearson or Kendall correlations) to create from data aggregated into granular group using the by parameter. E.g the *by=90* parameter will divide the dataset into groups of 90 observations and will apply the input function to each group of 90 data items. Hence in 360 total observations, a total of four separate correlation functions may be evaluated on aggregated data sets each containing 90 observations.

Value

A new signalSeries whose positions were adjusted via the 'by' parameter. Hence the new signalSeries data slot contains types returned by the AGGFUN. For example, if AGGFUNC is 'pearson' as in the example, then the data slot contains a vector of correlation coefficients each calculated by splitting the input data into successive blocks specified by using the number of items in 'by' for each new block. For each block, the AGGFUNC is applied to each column (or all columns joined if parameter 'together=TRUE') to calculate the appropriate result. The data slot contains the result applied to each successive block.

Author(s)

documentation by Scott Ulman for R-language distribution

See Also

[signalSeries](#)

Examples

```
## Not run:
set.seed(13);
m <- 90;
n <- 3000;
#Generate a 'matrix' class of simulated values with 2 columns and m*n rows
dataSim <- rmt(m*n,df=3,rho=0.5,d=2);
#create a signal series from simulated data:
dataSimSS <- signalSeries(dataSim);
#local function
pearson <- function(x) cor(x)[1,2];
pearson.cors <- aggregateSignalSeries(dataSimSS,by=m,
```

```

                                together=T, AGGFUNC=pearson);
#Extract the data part only to see a vector of correlation
#coefficients for each contiguous subblock
#in the entire original series.
pearson.cors.data <- pearson.cors@data;
## End(Not run)

```

```
aggregateWeeklySeries
```

```
aggregateWeeklySeries() method
```

Description

This is one of several substitutes for the S-Plus language method

aggregateSeries(timeseries, FUN=max, mean, colAvg, colSums,..., by=weeks, months, quarters,...,...).

The R-language **aggregateWeeklySeries()** function allows the user to calculate a less granular time-series (weekly) from a daily time series by using a statistic like the max, mean, sum, etc. Note the R-methods do NOT contain a **by="months"** parameter so the R-language user must select either the **aggregateWeeklySeries** method, the **aggregateMonthlySeries()** method, or the **aggregateQuarterlySeries()** method to get the desired result.

Usage

```
aggregateWeeklySeries(timeseries, FUNC = colSums)
```

Arguments

<code>timeseries</code>	a daily timeSeries (R-Metrics type from fCalendar) from which the user wants to extract a less granular weekly average timeSeries
<code>FUNC</code>	The name of a function to use in aggregating the data. For example the colAvg, max, mean, min, etc. The default is colSums

Details

For example, the user might want to create a series of weekly **colSums** returns from a daily time series of returns. Alternatively, (s)he might want the quarterly **mean** series. In either case, a less granular set of return values is calculated from a daily timeSeries object. Unfortunately, the R-Metrics package has not yet implemented an R-version of the S-Plus `aggregateSeries()` method.

The **aggregateWeeklySeries()**, **aggregateMonthlySeries()**, and the **aggregateQuarterlySeries()** are interim functions developed to convert daily timeSeries to weekly, monthly, or quarterly time-Series objects via a statistic like the max, mean, colAvg, or ColSums.

These functions exist in the functionsUtility.R file of the library.

Value

A weekly timeSeries object characterized by some statistic like colAvg, of the daily series over a month. The positions attribute (`dates <- rseries@positions`) of the new time series will be the LAST DAYS OF THE RESPECTIVE weeks for the timeSeries object.

Author(s)

documentation by Scott Ulman for R-language distribution

See Also

[aggregateMonthlySeries](#), [aggregateQuarterlySeries](#)

Examples

```
#load nasdaq data set:
data(nasdaq);
#Create daily return series:
nreturns <- mk.returns(nasdaq)
#convert to weekly series using colSums values (adding daily returns to get weekly)
weekly.nasdaq <- aggregateWeeklySeries(nreturns, FUNC=colSums);
## Not run:
Ret.DJ <- mk.returns(DJ);
#Choose only 10 of the 30 stocks:
selection1 <- c("AXP", "EK", "BA", "C", "KO", "MSFT", "HWP",
               "INTC", "JPM", "DIS");
partialDJ30dailyTS <- Ret.DJ[,selection1];
partialDJ30daily <- cut(partialDJ30dailyTS, from="1992-12-31",
                       to="2000-12-31");
partialDJ30weeklyTS <- aggregateWeeklySeries(partialDJ30daily,
                                              FUNC= colSums);
## End(Not run)
```

besselM3

Modified Bessel Function of 3rd Kind

Description

calculates modified Bessel function of third kind

Usage

```
besselM3(lambda=9/2, x=2, logvalue=FALSE)
```

Arguments

lambda	parameter of Bessel function
x	2nd parameter of Bessel function
logvalue	whether or not log value should be returned

Details

see page 497 of QRM and references given there

Value

vector of values of Bessel function with same length as x

See Also

[rGIG](#), [dghyp](#), [dmghyp](#)

beta (stats)

The Beta Distribution

Description

Density, distribution function, quantile function and random generation for the Beta distribution with parameters `shape1` and `shape2` (and optional non-centrality parameter `ncp`).

Arguments

<code>x, q</code>	vector of quantiles.
<code>p</code>	vector of probabilities.
<code>n</code>	number of observations. If <code>length(n) > 1</code> , the length is taken to be the number required.
<code>shape1, shape2</code>	positive parameters of the Beta distribution.
<code>ncp</code>	non-centrality parameter.
<code>log, log.p</code>	logical; if TRUE, probabilities <code>p</code> are given as <code>log(p)</code> .
<code>lower.tail</code>	logical; if TRUE (default), probabilities are $P[X \leq x]$, otherwise, $P[X > x]$.

Details

Usage:

```
dbeta(x, shape1, shape2, ncp=0, log = FALSE);
pbeta(q, shape1, shape2, ncp=0, lower.tail = TRUE, log.p = FALSE);
qbeta(p, shape1, shape2, lower.tail = TRUE, log.p = FALSE);
rbeta(n, shape1, shape2);
```

The Beta distribution with parameters `shape1 = a` and `shape2 = b` has density

$$f(x) = \frac{\Gamma(a+b)}{\Gamma(a)\Gamma(b)} x^a (1-x)^b$$

for $a > 0$, $b > 0$ and $0 \leq x \leq 1$ where the boundary values at $x = 0$ or $x = 1$ are defined as by continuity (as limits).

The mean is $a/(a+b)$ and the variance is $ab/((a+b)^2(a+b+1))$.

`pbeta` is closely related to the incomplete beta function. As defined by Abramowitz and Stegun 6.6.1

$$B_x(a, b) = \int_0^x t^{a-1} (1-t)^{b-1} dt,$$

and 6.6.2 $I_x(a, b) = B_x(a, b)/B(a, b)$ where $B(a, b) = B_1(a, b)$ is the Beta function ([beta](#)). $I_x(a, b)$ is `pbeta(x, a, b)`.

Value

`dbeta` gives the density, `pbeta` the distribution function, `qbeta` the quantile function, and `rbeta` generates random deviates.

Author(s)

documentation by Scott Ulman for R-language distribution

References

Becker, R. A., Chambers, J. M. and Wilks, A. R. (1988) *The New S Language*. Wadsworth, Brooks, and Cole.

Abramowitz, M. and Stegun, I. A. (1972) *Handbook of Mathematical Functions*. New York: Dover. Chapter 6: Gamma and Related Functions.

See Also

[dgamma](#), For more information on the beta function and others, see [Special](#).

Examples

```
x <- seq(0, 1, length=21);
dbeta(x, 1, 1); #actually a standard uniform density
pbeta(x, 1, 1) #actually a standard uniform distribution
```

BiDensPlot

Bivariate Density Plot

Description

makes perspective or contour plot of a bivariate density

Usage

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

Arguments

<code>func</code>	a function that evaluates on a n by 2 matrix to give n values of the bivariate density
<code>xpts</code>	limits of x range
<code>ypts</code>	limits of y range
<code>npts</code>	the number of subdivision points between x and y over the speicified range xpts to ypts
<code>type</code>	"persp" or "contour" plot
<code>...</code>	further parameters of density function

Side Effects

produces a contour or perspective plot

See Also

[dmnorm](#), [dmt](#)

Examples

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

<code>cac40</code>	<i>CAC 40 Stock Market Index</i>
--------------------	----------------------------------

Description

This timeSeries data set provides the daily closing values of the French CAC 40 stock index for the period 1994 to March 2004

<code>cal.beta</code>	<i>Calibrate Beta Mixture of Bernoullis</i>
-----------------------	---

Description

calibrates a beta mixture distribution on unit interval to give an exchangeable Bernoulli mixture model with prescribed default and joint default probabilities

Usage

```
cal.beta(pi1=0.1837, pi2=0.0413)
```

Arguments

<code>pi1</code>	default probability
<code>pi2</code>	joint default probability

Details

see pages 354-355 in QRM

Value

parameters a and b of beta mixing distribution

See Also

[cal.claytonmix](#), [cal.probitnorm](#), [rbinomial.mixture](#)

Examples

```

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);
## Not run:
claytonmix.pars <- cal.claytonmix(pi.B,pi2.B);
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);
abline(h = 0.01);
legend(0.05, 1e-4, c("Probit-normal", "Beta",
                    "Clayton-Mixture"), lty=rep(1,3),col = (1:3));
## End(Not run)

```

cal.claytonmix

*Calibrate Mixture of Bernoullis Equivalent to Clayton Copula Model***Description**

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.

Usage

```
cal.claytonmix(pi1=0.1837, pi2=0.0413)
```

Arguments

pi1	default probability
pi2	joint default probability

Details

see page 362 in QRM

Value

parameters pi and theta for Clayton copula default model

See Also

[cal.beta](#), [cal.probitnorm](#), [rclaytonmix](#), [rbinomial.mixture](#)

Examples

```
pi.B <- 0.2; pi2.B <- 0.05
## Not run:
probitnorm.pars <- cal.probitnorm(pi.B,pi2.B)
probitnorm.pars
beta.pars <- cal.beta(pi.B,pi2.B)
beta.pars
## End(Not run)
claytonmix.pars <- cal.claytonmix(pi.B,pi2.B)
claytonmix.pars
## Not run:
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);
abline(h = 0.01);
legend(0.05, 1e-4, c("Probit-normal", "Beta", "Clayton-Mixture"),
      lty=rep(1,3),col = (1:3))
## End(Not run)
```

cal.probitnorm

Calibrate Probitnormal Mixture of Bernoullis

Description

calibrates a probitnormal mixture distribution on unit interval to give an exchangeable Bernoulli mixture model with prescribed default and joint default probabilities

Usage

```
cal.probitnorm(pi1=0.1837, pi2=0.0413)
```

Arguments

pi1	default probability
pi2	joint default probability

Details

see page 354 in QRM

Value

parameters mu and sigma for probitnormal mixing distribution as well as the implied asset correlation rho.asset

See Also

[cal.beta](#), [cal.claytonmix](#), [dprobitnorm](#), [rbinomial.mixture](#)

Examples

```
pi.B <- 0.2; pi2.B <- 0.05;
probitnorm.pars <- cal.probitnorm(pi.B,pi2.B);
probitnorm.pars;
## Not run:
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);
abline(h = 0.01);
legend(0.05, 1e-4, c("Probit-normal", "Beta",
                    "Clayton-Mixture"), lty=rep(1,3),col = (1:3));
## End(Not run)
```

claytonmix

Mixing Distribution on Unit Interval Yielding Clayton Copula Model

Description

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

Usage

```
dclaytonmix(x, pi, theta)
pclaytonmix(q, pi, theta)
rclaytonmix(n, pi, theta)
```

Arguments

x	values at which density should be evaluated
q	values at which cumulative distribution should be evaluated
n	sample size
pi	parameter of distribution
theta	parameter of distribution

Details

see page 362 in QRM

Value

values of density (dclaytonmix), distribution function (pclaytonmix) or random sample (rclaytonmix)

Author(s)

documentation by Scott Ulman for R-language distribution

See Also

[dbeta](#), [dprobitnorm](#)

Examples

```
#probability of only one obligor defaulting B class (see Table 8.6 in QRM book)
pi.B <- 0.0489603;
#joint probability of two obligors defaulting B class (see Table 8.6 in QRM book)
pi2.B <- 0.003126529;
# Calibrate Calyton copula model to pi.B and pi2.B
claytonmix.pars <- cal.claytonmix(pi.B,pi2.B)
# We could also look at mixing densities. Get probability of Clayton mix
# This picture essentially shows large sample asymptotics
#Build 1000 equally-spaced values on unit interval (multiples of .000999);
#discard all values except those below 0.25
q <- (1:1000)/1001;
q <- q[q<0.25]; #reduce to lowest 250 values
#get probabilities for each of 250 lowest values on unit interval
d.claytonmix <- dclaytonmix(q,claytonmix.pars[1],claytonmix.pars[2]);
```

CovToCor

Covariance To Correlation Matrix

Description

extracts the correlation matrix from a covariance matrix

Usage

```
CovToCor(mat)
```

Arguments

`mat` a covariance matrix

Details

This is a custom function built by Alexander McNeil. It provides the same functionality as R's built in `cov2cor()` method in the stats package.

Value

a correlation matrix

See Also

[equicorr](#) [cov2cor](#)

Danish	<i>Danish Data</i>
--------	--------------------

Description

Danish fire insurances losses from 1980 to 1990 in millions of kroner

<code>dcopula.clayton</code>	<i>Bivariate Clayton Copula Density</i>
------------------------------	---

Description

evaluates density of bivariate Clayton copula

Usage

```
dcopula.clayton(u, theta, logvalue=FALSE)
```

Arguments

`u` matrix of dimension n times 2, where 2 is the dimension of the copula and n is the number of vector values at which to evaluate density

`theta` parameter of Clayton copula

`logvalue` whether or not log density values should be returned (useful for ML)

Details

see page 192 of QRM for Clayton copula

Value

vector of density values of length n

See Also

[fit.Archcopula2d](#), [dcopula.gauss](#), [dcopula.t](#), [dcopula.gumbel](#)

Examples

```
## Not run:
#define a function:
normal.metaclayton <- function(x,theta);
{
  exp(dcopula.clayton(apply(x,2,pnorm),theta,logvalue=TRUE) +
    apply(log(apply(x,2,dnorm)),1,sum));
}
#use function to create perspective plot for bivariate density:
BiDensPlot(normal.metaclayton,xpts=11,ypts=11,npts=80,theta=2.2);
## End(Not run)
```

dcopula.gauss

Gauss Copula Density

Description

evaluates density of Gauss copula

Usage

```
dcopula.gauss(u, P, logvalue=FALSE)
```

Arguments

u	matrix of 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
P	correlation matrix of Gauss copula
logvalue	whether or not log density values should be returned (useful for ML)

Details

see pages 197 and 234 in QRM

Value

vector of density values of length n

See Also

[dmnorm](#), [dcopula.clayton](#), [dcopula.t](#), [dcopula.gumbel](#)

Examples

```
## Not run:
ll <- c(0.01,0.99);
#create perspective plot for bivariate density:
BiDensPlot(func=dcopula.gauss,xpts=11,ypts=11,P=equicorr(2,0.5));
## End(Not run)
```

dcopula.gumbel

Bivariate Gumbel Copula Density

Description

evaluates density of bivariate Gumbel copula

Usage

```
dcopula.gumbel(u, theta, logvalue=FALSE)
```

Arguments

u	matrix of dimension n times 2, where 2 is the dimension of the copula and n is the number of vector values at which to evaluate density
theta	parameter of Gumbel copula
logvalue	whether or not log density values should be returned (useful for ML)

Details

see page 192 of QRM for Gumbel copula

Value

vector of density values of length n

See Also

[fit.Archcopula2d](#), [dcopula.clayton](#), [dcopula.t](#), [dcopula.gauss](#)

Examples

```
## Not run:
#define a function:
normal.metagumbel <- function(x,theta)
{
  exp(dcopula.gumbel(apply(x,2,pnorm),theta,logvalue=TRUE) +
    apply(log(apply(x,2,dnorm)),1,sum));
}
#use function to create perspective plot for bivariate density:
BiDensPlot(normal.metagumbel,xpts=11,ypts=11,npts=80,theta=2);
## End(Not run)
```

dcopula.t

t Copula Density

Description

evaluates density of t copula

Usage

```
dcopula.t(u, nu, P, logvalue=FALSE)
```

Arguments

u	matrix of 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
nu	degrees of freedom of t copula
P	correlation matrix of t copula
logvalue	whether or not log density values should be returned (useful for ML)

Details

see pages 197 and 235 of QRM

Value

vector of density values of length n

See Also

[dmt](#), [dcopula.clayton](#), [dcopula.gumbel](#), [dcopula.gauss](#)

Examples

```
## Not run:
ll <- c(0.01,0.99);
#create perspective plot for bivariate density:
BiDensPlot(func=dcopula.t,xpts=ll,ypts=ll,nu=4,P=equicorr(2,0.5));
#define an internal function:
normal.metat <- function(x,nu,P)
{
  exp(dcopula.t(apply(x,2,pnorm),nu,P,logvalue=TRUE) +
    apply(log(apply(x,2,dnorm)),1,sum));
}
#use function to create perspective plot for bivariate density:
BiDensPlot(normal.metat,xpts=ll,ypts=ll,npts=80,nu=4,P=equicorr(2,0.71));
## End(Not run)
```

DJComponents	<i>Dow Jones 30 Stock Prices</i>
--------------	----------------------------------

Description

This timeSeries data set provides the closing values of the Dow Jones 30 Stocks from 1991-2000

dji	<i>Dow Jones Index</i>
-----	------------------------

Description

This timeSeries data set provides the daily closing values of the Dow Jones index from 1980 to March 2004

dmnorm	<i>Multivariate Normal Density</i>
--------	------------------------------------

Description

evaluates multivariate normal density

Usage

```
dmnorm(x, mu, Sigma, logvalue=FALSE)
```

Arguments

x	matrix with n rows and d columns; density is evaluated at each vector of row values
mu	mean vector
Sigma	covariance matrix
logvalue	should log density be returned; default is FALSE

Value

vector of length n containing values of density or log-density

Author(s)

documentation by Scott Ulman for R-language distribution

See Also

[dmt](#), [dmghyp](#)

Examples

```
### Normal distribution: visualization, simulation, estimation
BiDensPlot(func=dmnorm,mu=c(0,0),Sigma=equicorr(2,-0.7));
```

dmt *Multivariate Student t Density*

Description

evaluates multivariate Student t density

Usage

```
dmt(x, nu, mu, Sigma, logvalue=FALSE)
```

Arguments

x	matrix with n rows and d columns; density is evaluated at each vector of row values
nu	degree of freedom parameter
mu	location vector
Sigma	dispersion matrix
logvalue	should log density be returned; default is FALSE

Value

vector of length n containing values of density or log-density

See Also

[dmnorm](#), [dmghyp](#)

Examples

```
### t distribution: visualization, simulation, estimation
BiDensPlot(func=dmt, xpts=c(-4,4), ypts=c(-4,4), mu=c(0,0),
            Sigma=equicorr(2,-0.7), nu=4);
```

dsmghyp *Symmetric Multivariate Generalized Hyperbolic Distribution*

Description

Density of elliptical subfamily of multivariate generalized hyperbolic family. The symmetric family is a normal-variance mixture since the gamma parameter associated with the mean mixture is by assumption equal to zero.

Usage

```
dsmghyp(x, lambda, chi, psi, mu, Sigma, logvalue=FALSE)
```

Arguments

<code>x</code>	matrix with <code>n</code> rows and <code>d</code> columns; density is evaluated at each vector of row values
<code>lambda</code>	scalar parameter
<code>chi</code>	scalar parameter
<code>psi</code>	scalar parameter
<code>mu</code>	location vector
<code>Sigma</code>	dispersion matrix
<code>logvalue</code>	should log density be returned; default is FALSE

Details

See page 78 in QRM for joint density formula (3.30) with `Sigma` a d -dimensional dispersion matrix ($d > 1$) consistent with a multivariate distribution). This is a more intuitive parameterization of the alpha-beta-delta model used by Blaesild (1981) in earlier literature since it associates all parameters with mixtures of both mean and variance. Since `gamma` is 0, we have a normal-variance mixture where the mixing variable `W` has a GIG (generalized inverse gaussian) distribution with parameters `lambda`, `chi`, `psi`. This thickens the tail.

Since `gamma` equals zero, we have no perturbation of the mean so no ASYMMETRY is introduced and hence the distribution is symmetric.

There is no random number generation associated with the multivariate model in this implementation of the R-language and S-Plus code.

See pp. 77-81 of QRM and appendix A.2.5 for details.

`dsmghyp()` is frequently called from the function `dmghyp()`.

Value

vector of length `n` containing values of density or log-density

Author(s)

documentation by Scott Ulman for R-language distribution

See Also

[dmghyp](#)

Examples

```
## Not run:
dmghyp <- function(x, lambda, chi, psi, mu, Sigma, gamma, logvalue=FALSE)
{
  #Call symmetric form if gamma vector is identically zero:
  if (sum(abs(gamma))==0)
    out <- dsmghyp(x, lambda, chi, psi, mu, Sigma, logvalue=TRUE);
  # lines removed here
}
## End(Not run)
```

edf

*Empirical Distribution Function***Description**

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

Usage

```
edf(v, adjust=FALSE)
```

Arguments

`v` a vector

`adjust` should the denominator be adjusted to be (n+1)? The default is FALSE

Value

vector of probabilities

Examples

```
## Not run:
data(INDEXES.RAW);
INDEXES <- mk.returns(INDEXES.RAW);
PARTIALINDEXES <- cut(INDEXES, "1993-12-31", "2003-12-31");
#Now create a data matrix from the just-created timeSeries
data <- seriesData(PARTIALINDEXES);
#Keep only the data items which are non-zero for both smi and ftse100
data <- data[data[,1]!=0 & data[,2] !=0,];
# Construct pseudo copula data. The 2nd parameter is MARGIN=2
#when applying to columns and 1 applied to rows. Hence this says to
#apply the 'edf()' empirical distribution function() to the columns
#of the data.
Udata <- apply(data,2,edf,adjust=1);
plot(Udata);
## End(Not run)
```

EGIG

*Estimate Moments of GIG Distribution***Description**

Calculates moments of univariate generalized inverse Gaussian (GIG) distribution

Usage

```
EGIG(lambda, chi, psi, k=1)
```

Arguments

lambda	lambda parameter
chi	chi parameter
psi	psi parameter
k	order of moment

Details

Normal variance mixtures are frequently obtained by perturbing the variance component of a normal distribution; we multiply by the square root of a mixing variable assumed to have a GIG (generalized inverse gaussian) distribution depending upon three parameters lambda, chi, and 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.

Also see page 497 of QRM Book for a description of the GIG distribution.

Value

mean of distribution

See Also

[rGIG](#) [ElogGIG](#)

eigenmeth

Make Matrix Positive Definite

Description

adjusts a negative definite symmetric matrix to make it positive definite

Usage

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

Arguments

mat	a symmetric matrix
delta	new size of smallest eigenvalues

Details

see page 231 of QRM

Value

a positive-definite matrix

See Also

[fit.tcopula.rank](#)

ElogGIG

Log Moment of GIG

Description

calculates log moment of generalized hyperbolic distribution

Usage

```
ElogGIG(lambda, chi, psi)
```

Arguments

lambda	lambda parameter
chi	chi parameter
psi	psi parameter

Details

see page 497 of QRM

Value

log moment

See Also

[rGIG](#), [EGIG](#)

EMupdate

EM Update Step for Generalized Hyperbolic Estimation

Description

updates estimates of location (μ), dispersion (Σ) and skewness (γ) parameters in EM estimation of multivariate generalized hyperbolic distributions

Usage

```
EMupdate(data, mix.pars, mu, Sigma, gamma, symmetric,
          scaling=TRUE, kvalue=1)
```

Arguments

<code>data</code>	data matrix
<code>mix.pars</code>	current values of lambda, chi and psi
<code>mu</code>	current value of mu
<code>Sigma</code>	current value of sigma
<code>gamma</code>	current value of gamma
<code>symmetric</code>	logical variable for elliptically symmetric case
<code>scaling</code>	do we scale determinant of Sigma to be fixed value?
<code>kvalue</code>	value of determinant in the case of scaling

Details

See pp 81-83 of QRM; in that case k is the determinant of the sample covariance matrix. ‘EM’ stands for the “Expectation-Maximization” type of algorithm used to fit proposed multivariate hyperbolic models to actual data.

Value

a list with updated estimates of μ (location), Σ (dispersion) and γ (skewness)

Author(s)

documentation by Scott Ulman for R-language distribution

See Also

[fit.mNH](#)

equicorr

Equicorrelation Matrix

Description

constructs an equicorrelation matrix

Usage

```
equicorr(d, rho)
```

Arguments

<code>d</code>	dimension of matrix
<code>rho</code>	value of correlation

Value

an equicorrelation matrix

See Also

[rmnorm](#), [rmt](#)

Examples

```
equicorr(7,0.5);
# Bivariate Visualization
ll <- c(0.01,0.99)
BiDensPlot(func=dcopula.gauss,xpts=ll,ypts=ll,P=equicorr(2,0.5));
BiDensPlot(func=dcopula.t,xpts=ll,ypts=ll,nu=4,P=equicorr(2,0.5));
```

ESnorm	<i>Expected Shortfall for Normal Distribution</i>
--------	---

Description

calculates expected shortfall for normal distribution

Usage

```
ESnorm(p, mean=0, sd=1)
```

Arguments

p	probability level
mean	mean
sd	standard deviation

Details

see page 45 of QRM

Value

expected shortfall

See Also

[ESst](#)

Examples

```
ESnorm(c(0.95,0.99))
```

ESst

Expected Shortfall for Student t Distribution

Description

calculates expected shortfall for Student t distribution

Usage

```
ESst(p, df, mu=0, sigma=1, scale=FALSE)
```

Arguments

p	probability level
df	degrees of freedom
mu	mean
sigma	standard deviation
scale	should t distribution be scaled to have variance one?

Details

see page 45 of QRM

Value

expected shortfall

See Also

[ESnorm](#)

Examples

```
#Set up the quantile probabilities
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);
#Now look at Expected Shortfall for student t with 4 degrees of freedom:
ES.t4 <- ESst(p,4,sigma=sigma,scale=TRUE);
ESst(c(0.95,0.99),4);
```

extremalPP	<i>Extremal Point Process</i>
------------	-------------------------------

Description

creates an extremal point process of class MPP

Usage

```
extremalPP(data, threshold = NA, nextremes = NA)
```

Arguments

data	a timeSeries object or vector of numbers to be interpreted as a regular time series
threshold	threshold value (either this or "nextremes" must be given but not both)
nextremes	the number of upper extremes to be used (either this or "threshold" must be given but not both)

Details

see pages 298-301 of QRM

Value

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)
starttime	the julian count one day prior to the first date in the entire timeSeries
endtime	value of last julian count in entire timeSeries
threshold	value of threshold above which exceedances are calculated

See Also

[unmark](#), [fit.sePP](#), [fit.seMPP](#)

Examples

```
data(sp500);
sp500.nreturns <- -mk.returns(sp500);
tD <- timeDate("12/31/1995", "%m/%d/%Y");
window <- (seriesPositions(sp500.nreturns) > tD);
sp500.nreturns <- sp500.nreturns[window];
tmp <- extremalPP(sp500.nreturns, ne=100);
tmp$marks[1:5];
tmp$threshold;
```

findthreshold	<i>Find a Threshold</i>
---------------	-------------------------

Description

find threshold corresponding to given number of upper order statistics

Usage

```
findthreshold(data, ne)
```

Arguments

data	data vector
ne	vector giving number of extremes above the threshold

Details

when the data are tied a threshold is found so that at least the specified number of extremes lie above

Value

vector of suitable thresholds

See Also

[fit.GPD](#)

Examples

```
data(danish);
# Find threshold giving (at least) fifty exceedances for Danish data
findthreshold(danish, 50);
```

fit.Archcopula2d	<i>Fit 2D Archimedean Copula</i>
------------------	----------------------------------

Description

fits two-dimensional Archimedean copula by maximum likelihood

Usage

```
fit.Archcopula2d(data, name)
```

Arguments

data	matrix of copula data with two columns taking values in unit interval
name	name of Archimedean copula: "clayton", "gumbel"

Details

see pages 234-236 of QRM

Value

list containing parameter estimate, standard error, value of log-likelihood at maximum and convergence flag

See Also

[fit.gausscopula](#), [fit.tcopula](#)

Examples

```
data(ftse100);
data(smi);
TS1 <- cut(ftse100, "1990-11-08", "2004-03-25");
TS1Augment <- alignDailySeries(TS1, method="before");
TS2Augment <- alignDailySeries(smi, method="before");
INDEXES.RAW <- merge(TS1Augment, TS2Augment);
#Cleanup:
rm(TS1, TS1Augment, TS2Augment);
INDEXES <- mk.returns(INDEXES.RAW);
PARTIALINDEXES <- cut(INDEXES, "1993-12-31", "2003-12-31");
#Now create a data matrix from the just-created timeSeries
data <- seriesData(PARTIALINDEXES);
#Keep only the data items which are non-zero for both smi and ftse100
data <- data[data[,1]!=0 & data[,2] !=0,];
# Construct pseudo copula data. The 2nd parameter is MARGIN=2
#when applying to columns and 1 applied to rows. Hence this says to
#apply the 'edf()' empirical distribution function() to the columns
#of the data.
Udata <- apply(data, 2, edf, adjust=1);
#Fit 2-dimensional Archimedian copula: choices are gumbel or clayton
#using pseudo data generated via edf() from observed data:
mod.gumbel <- fit.Archcopula2d(Udata, "gumbel");
## Not run:
mod.clayton <- fit.Archcopula2d(Udata, "clayton");
## End(Not run)
```

fit.binomial

Fit Binomial Distribution

Description

fits binomial distribution by maximum likelihood

Usage

```
fit.binomial(M, m)
```

Arguments

M	vector of numbers of successes
m	vector of numbers of trials

Value

list containing parameter estimates and details of fit

See Also

[fit.binomialBeta](#), [fit.binomialLogitnorm](#), [fit.binomialProbitnorm](#)

Examples

```
## Not run:
spdata.raw;
attach(spdata.raw);
BdefaultRate <- Bdefaults/Bobligors;
mod0 <- fit.binomial(Bdefaults, Bobligors);
## End(Not run)
```

fit.binomialBeta	<i>Fit Beta-Binomial Distribution</i>
------------------	---------------------------------------

Description

fit a beta-binomial distribution by maximum likelihood

Usage

```
fit.binomialBeta(M, m, startvals=c(2, 2), ses=FALSE)
```

Arguments

M	vector of numbers of successes
m	vector of numbers of trials
startvals	starting values for parameter estimates
ses	whether standard errors should be calculated

Value

list containing parameter estimates and details of fit

Author(s)

documentation by Scott Ulman for R-language distribution

See Also

[fit.binomial](#), [fit.binomialLogitnorm](#), [fit.binomialProbitnorm](#)

Examples

```
## Not run:
data(spdata.raw);
spdata.raw;
#attach data so we don't have to qualify the data column names:
attach(spdata.raw);
BdefaultRate <- Bdefaults/Bobligors;
modl <- fit.binomialBeta(Bdefaults, Bobligors);
## End(Not run)
```

fit.binomialLogitnorm

Fit Logitnormal-Binomial Distribution

Description

fits a mixed binomial distribution where success probability has a logitnormal distribution This function has been altered in the R-language edition to contain two extra parameters providing upper and lower limits for the input parameters M and m. if convergence occurs at an endpoint of either limit, you need to reset lower and upper parameter estimators and run the function again

Usage

```
fit.binomialLogitnorm(M, m, startvals=c(-1, 0.5),
lowerParamLimits = c(-5.0, 0.02), upperParamLimits = c(1,0.9))
```

Arguments

M	vector of numbers of successes (e.g. number of defaults in a credit-rating class)
m	vector of numbers of trials (e.g. number of obligors in a credit-rating class)
startvals	starting values for parameter estimates
lowerParamLimits	vector with lower limits for each parameter to be used by optimization algorithm
upperParamLimits	vector with upper limits for each parameter to be used by optimization algorithm

Details

This function calls the R-language method `optim(...method="L-BFGS-B")` which uses input parameter vectors of upper and lower limits. Hence if convergence occurs at an endpoint of either limit, you may need to expand the corresponding upper or lower limit and run the function again.

Value

list containing parameter estimates and details of fit:

par.ests	vector of optimum parameter estimators
maxloglik	value of likelihood function at optimum
converged	T or F indicating convergence
details	any messages associated with convergence algorithm

pi	probability of a single default (see p. 345 in QRM)
pi2	probability of two joint defaults (see p. 345 in QRM)
rhoY	default correlation (see p. 345 in QRM)

Author(s)

documentation by Scott Ulman for R-language distribution

See Also

[fit.binomial](#), [fit.binomialBeta](#), [fit.binomialProbitnorm](#)

Examples

```
## Not run:
spdata.raw;
attach(spdata.raw);
BdefaultRate <- Bdefaults/Bobligors;
# A little patience is required for the next model ...
mod3 <- fit.binomialLogitnorm(Bdefaults, Bobligors);
## End(Not run)
```

fit.binomialProbitnorm

Fit Probitnormal-Binomial Distribution

Description

Fits a mixed binomial distribution where success probability has a probitnormal distribution. This function has been altered in the R-language edition to contain two extra parameters providing upper and lower limits for the input parameters M and m. if convergence occurs at an endpoint of either limit, you need to reset lower and upper parameter estimators and run the function again

Usage

```
fit.binomialProbitnorm(M, m, startvals=c(-1, 0.5),
lowerParamLimits = c(-3.0, 0.02), upperParamLimits = c(1, 0.9))
```

Arguments

M	vector of numbers of successes (e.g. number of defaults in a credit-rating class)
m	vector of numbers of trials (e.g. number of obligors in a credit-rating class)
startvals	starting values for parameter estimates
lowerParamLimits	vector with lower limits for each parameter to be used by optimization algorithm
upperParamLimits	vector with upper limits for each parameter to be used by optimization algorithm

Details

This function calls the R-language method `optim(...method="L-BFGS-B")` which uses input parameter vectors of upper and lower limits. Hence if convergence occurs at an endpoint of either limit, you may need to expand the corresponding upper or lower limit and run the function again.

Value

list containing parameter estimates and details of fit

<code>par.ests</code>	vector of parameter estimators
<code>maxloglik</code>	value of likelihood function at optimum
<code>converged</code>	T or F indicating convergence
<code>details</code>	any messages associated with convergence algorithm
<code>pi</code>	probability of a single default (see p. 345 in QRM)
<code>pi2</code>	probability of two joint defaults (see p. 345 in QRM)
<code>rhoY</code>	default correlation (see p. 345 in QRM)

Author(s)

documentation by Scott Ulman for R-language distribution

See Also

[fit.binomial](#), [fit.binomialBeta](#), [fit.binomialLogitnorm](#)

Examples

```
## Not run:
spdata.raw;
attach(spdata.raw);
BdefaultRate <- Bdefaults/Bobligors;
mod2 <- fit.binomialProbitnorm(Bdefaults, Bobligors);
## End(Not run)
```

<code>fit.gausscopula</code>	<i>Fit Gauss Copula</i>
------------------------------	-------------------------

Description

fits Gauss copula to pseudo-copula data

Usage

```
fit.gausscopula(Udata)
```

Arguments

<code>Udata</code>	matrix of pseudo-copula data where rows are vector observations with all values in unit interval
--------------------	--

Details

see pages 234-235 of QRM

Value

list containing parameter estimates and details of fit

See Also

[fit.tcopula](#), [fit.Archcopula2d](#)

Examples

```
data(ftse100);
data(smi);
TS1 <- cut(ftse100, "1990-11-08", "2004-03-25");
TS1Augment <- alignDailySeries(TS1, method="before");
TS2Augment <- alignDailySeries(smi, method="before");
INDEXES.RAW <- merge(TS1Augment, TS2Augment);
#Cleanup:
rm(TS1, TS1Augment, TS2Augment);
INDEXES <- mk.returns(INDEXES.RAW);
PARTIALINDEXES <- cut(INDEXES, "1993-12-31", "2003-12-31");
#Now create a data matrix from the just-created timeSeries
data <- seriesData(PARTIALINDEXES);
#Keep only the data items which are non-zero for both smi and ftse100
data <- data[data[,1]!=0 & data[,2] !=0,];
# Construct pseudo copula data. The 2nd parameter is MARGIN=2
#when applying to columns and 1 applied to rows. Hence this says to
#apply the 'edf()' empirical distribution function() to the columns
#of the data.
Udata <- apply(data, 2, edf, adjust=1);
mod.gauss <- fit.gausscopula(Udata);
mod.gauss;
```

fit.GEV

Fit Generalized Extreme Value Distribution

Description

fits generalized extreme value distribution (GEV) to block maxima data

Usage

```
fit.GEV(maxima)
```

Arguments

maxima block maxima data

Details

see pages 271-272 of QRM

Value

list containing parameter estimates, standard errors and details of the fit

See Also

[pGEV](#), [pGPD](#), [fit.GPD](#)

Examples

```
data(nasdaq);
nreturns <- -mk.returns(nasdaq);
monthly.maxima <- aggregateMonthlySeries(nreturns, FUN=max);
monthly.maxima <- seriesData(monthly.maxima)
mod1 <- fit.GEV(monthly.maxima);
```

fit.GPD

Fit Generalized Pareto Model

Description

fits a generalized Pareto distribution to threshold exceedances

Usage

```
fit.GPD(data, threshold=NA, nextremes=NA, method="ml", information="observed")
```

Arguments

data	data vector or times series
threshold	a threshold value (either this or "nextremes" must be given but not both)
nextremes	the number of upper extremes to be used (either this or "threshold" must be given but not both)
method	whether parameters should be estimated by the maximum likelihood method "ml" or the probability-weighted moments method "pwm"
information	whether standard errors should be calculated with "observed" or "expected" information. This only applies to maximum likelihood method; for "pwm" method "expected" information is used if possible.

Details

see page 278 of QRM; this function uses optim() for ML

Value

a list containing parameter estimates, standard errors and details of the fit

References

Parameter and quantile estimation for the generalized Pareto distribution, JRM Hosking and JR Wallis, Technometrics 29(3), pages 339-349, 1987.

See Also

[pGPD](#), [fit.GPDb](#), [pGEV](#), [fit.GEV](#)

Examples

```
data(danish);
plot(danish);
losses <- seriesData(danish);
mod <- fit.GPD(danish,10);
mod$par.ests;
modb <- fit.GPD(danish,10,method="pwm");
modb$par.ests;
```

fit.GPDb

Fit Generalized Pareto Model B

Description

fits a generalized Pareto distribution to threshold exceedances using `nlminb()` rather than `nlmin()`

Usage

```
fit.GPDb(data, threshold=NA, nextremes=NA, method="ml", information="observed")
```

Arguments

<code>data</code>	data vector or times series
<code>threshold</code>	a threshold value (either this or "nextremes" must be given but not both)
<code>nextremes</code>	the number of upper extremes to be used (either this or "threshold" must be given but not both)
<code>method</code>	whether parameters should be estimated by the maximum likelihood method "ml" or the probability-weighted moments method "pwm"
<code>information</code>	whether standard errors should be calculated with "observed" or "expected" information. This only applies to maximum likelihood method; for "pwm" method "expected" information is used if possible.

Details

see page 278 of QRM; this function uses "nlminb" for ML

Value

a list containing parameter estimates, standard errors and details of the fit

References

Parameter and quantile estimation for the generalized Pareto distribution, JRM Hosking and JR Wallis, *Technometrics* 29(3), pages 339-349, 1987.

See Also

[fit.GPD](#), [fit.GEV](#), [RiskMeasures](#)

Examples

```
## Not run:
losses <- seriesData(danish);
mod <- fit.GPDb(losses,10);
mod$par.ests ;
## End(Not run)
```

fit.mNH

*Fit Multivariate NIG or Hyperbolic Distribution***Description**

fits multivariate NIG or hyperbolic distribution using variant of EM algorithm

Usage

```
fit.mNH(data=data.hyp.5d, symmetric=FALSE, case="NIG",
        kvalue=NA, nit=2000, tol=1e-10)
```

Arguments

data	matrix of data where rows are vector observations
symmetric	whether symmetric case should be fitted; default is FALSE
case	whether NIG ("NIG") or hyperbolic ("hyp") should be fitted
kvalue	value to which to constrain determinant of dispersion matrix
nit	maximum number of iterations
tol	tolerance for convergence

Details

see pages 81-83 in QRM

Value

list containing parameter estimates, standard errors and details of fit

Author(s)

documentation by Scott Ulman for R-language distribution

See Also

[fit.mst](#), [fit.NH](#), [EMupdate](#), [MCECMupdate](#), [MCECM.Qfunc](#)

Examples

```

data(DJ);
Ret.DJ <- mk.returns(DJ);
window1.start <- timeDate("01/01/1993", format="%m/%d/%Y");
window1.end <- timeDate("12/31/2000", format="%m/%d/%Y");
sample1 <- (seriesPositions(Ret.DJ) > window1.start &
           seriesPositions(Ret.DJ) < window1.end);
selection1 <- c("AXP", "EK", "BA", "C", "KO", "MSFT",
               "HWP", "INTC", "JPM", "DIS");
DJ30daily <- Ret.DJ[sample1, selection1];
DJ30weekly <- aggregateWeeklySeries(DJ30daily, FUNC= colSums);
mod.NIG <- fit.mNH(DJ30weekly, symmetric=FALSE, case="NIG");
## Not run:
mod.GAUSS <- fit.norm(DJ30weekly);
mod.HYP <- fit.mNH(DJ30weekly, symmetric=FALSE, case="hyp");
mod.t <- fit.mst(DJ30weekly);
mod.NIGs <- fit.mNH(DJ30weekly, symmetric=TRUE, case="NIG");
mod.HYPs <- fit.mNH(DJ30weekly, symmetric=TRUE, case="hyp");
round(c(mod.GAUSS$ll.max, mod.t$ll.max, mod.NIGs$ll.max,
        mod.HYPs$ll.max, mod.NIG$ll.max, mod.HYP$ll.max), 1);
## End(Not run)

```

fit.mst

*Fit Multivariate Student t Distribution***Description**

fits multivariate Student's t distribution using variant of EM algorithm

Usage

```
fit.mst(data=data.t.5d, nit=2000, tol=1e-10)
```

Arguments

data	matrix of data where rows are vector observations
nit	number of iterations of EM-type algorithm
tol	tolerance of improvement for stopping iteration

Details

see page 75 of QRM

Value

list containing parameter estimates, standard errors and details of fit

Author(s)

documentation by Scott Ulman for R-language distribution

See Also

[fit.mNH](#), [fit.NH](#), [fit.st](#)

Examples

```
data(DJ);
Ret.DJ <- mk.returns(DJ);
window1.start <- timeDate("01/01/1993",format="%m/%d/%Y");
window1.end <- timeDate("12/31/2000",format="%m/%d/%Y");
sample1 <- (seriesPositions(Ret.DJ) > window1.start
           & seriesPositions(Ret.DJ) < window1.end);
selection1 <- c("AXP", "EK", "BA", "C", "KO", "MSFT",
               "HWP", "INTC", "JPM", "DIS");
DJ30daily <- Ret.DJ[sample1,selection1];
DJ30weekly <- aggregateWeeklySeries(DJ30daily, FUNC= colSums);
mod.t <- fit.mst(DJ30weekly);
## Not run:
mod.GAUSS <- fit.norm(DJ30weekly);
mod.NIG <- fit.mNH(DJ30weekly,symmetric=FALSE,case="NIG");
mod.HYP <- fit.mNH(DJ30weekly,symmetric=FALSE,case="hyp");
mod.NIGs <- fit.mNH(DJ30weekly,symmetric=TRUE,case="NIG");
mod.HYPs <- fit.mNH(DJ30weekly,symmetric=TRUE,case="hyp");
round(c(mod.GAUSS$ll.max,mod.t$ll.max,mod.NIGs$ll.max,
        mod.HYPs$ll.max,mod.NIG$ll.max,mod.HYP$ll.max),1);
## End(Not run)
```

fit.NH

Fit NIG or Hyperbolic Distribution

Description

fits univariate NIG or hyperbolic distribution

Usage

```
fit.NH(data, case="NIG", symmetric=FALSE, se=FALSE)
```

Arguments

data	vector of data
case	whether NIG ("NIG") or hyperbolic ("hyp"); default is NIG
symmetric	whether symmetric or asymmetric; default is FALSE
se	whether standard errors should be calculated

Details

See pages 78-80 of QRM. Case 'NIG' sets lambda to -1/2; case 'hyp' sets lambda to 1; no other cases are allowed.

Value

list containing parameter estimates, standard errors and details of fit

Author(s)

documentation by Scott Ulman for R-language distribution

See Also

[fit.st](#), [fit.mNH](#), [fit.mst](#)

Examples

```
data(DJ);
Ret.DJ <- mk.returns(DJ);
window1.start <- timeDate("01/01/1993", format="%m/%d/%Y");
window1.end <- timeDate("12/31/2000", format="%m/%d/%Y");
sample1 <- (seriesPositions(Ret.DJ) > window1.start
           & seriesPositions(Ret.DJ) < window1.end);
DJ30daily <- Ret.DJ[sample1,];
DJ30daily <- 100*seriesData(DJ30daily);
rseries <- DJ30daily[, "MSFT"];
mod.NIG <- fit.NH(rseries);
## Not run:
mod.gauss <- fit.norm(rseries);
mod.t <- fit.st(rseries);
mod.hyp <- fit.NH(rseries, case="hyp");
c(mod.gauss$ll.max, mod.t$ll.max, mod.NIG$ll.max, mod.hyp$ll.max);
## End(Not run)
```

fit.norm

Fit Multivariate Normal

Description

fits multivariate (or univariate) normal by maximum likelihood

Usage

```
fit.norm(data)
```

Arguments

data matrix of data where each row is a vector observation

Value

list containing MLEs and value of likelihood at maximum

See Also

[dmnorm](#)

Examples

```
data <- rmnorm(1000, rho=0.7, d=3);
fit.norm(data);
```


fit.POT

*Peaks-over-Threshold Model***Description**

fits the POT (peaks-over-threshold) model to a point process of class PP or MPP

Usage

```
fit.POT(PP, markdens = "GPD")
```

Arguments

PP	a point process object of class PP or MPP
markdens	(optional) name of density of mark distribution, currently must be "GPD"

Details

see pages 301-305 of QRM. Note that if point process is of class PP then function simply estimates the rate of a homogeneous Poisson process.

Value

a list containing parameters of fitted POT model

par.ests	vector of parameter estimates
par.ses	vector of parameter std deviations
ll.max	loglikelihood maximum

References

see pages 301-305 of QRM

See Also

[fit.GPD](#), [extremalPP](#)

Examples

```
data(sp500);
sp500.nreturns <- -mk.returns(sp500);
window <- (seriesPositions(sp500.nreturns) >
  timeDate("12/31/1995", format="%m/%d/%Y"));
sp500.nreturns <- sp500.nreturns[window];
tmp <- extremalPP(sp500.nreturns, ne=100);
modl <- fit.POT(tmp);
## Not run:
modlb <- fit.POT(unmark(tmp));
## End(Not run)
```

fit.seMPP

*Fit Marked Self-Exciting Point Process***Description**

fits marked self-exciting process to a point process object of class MPP

Usage

```
fit.seMPP(PP, markdens = "GPD", model = "Hawkes", mark.influence = TRUE,
predictable = FALSE, std.errs = FALSE)
```

Arguments

PP	a point process object of class MPP
markdens	name of density of mark distribution; currently must be "GPD"
model	name of self-exciting model: Hawkes or ETAS
mark.influence	whether marks of marked point process may influence the self-excitement
predictable	whether previous events may influence the scaling of mark distribution
std.errs	whether standard errors should be computed VALUE

Details

see pages 307-309 of QRM

Value

a fitted self-exciting process object of class sePP

See Also

[fit.sePP](#), [plot.sePP](#), [stationary.sePP](#)

Examples

```
data(sp500);
sp500.nreturns <- -mk.returns(sp500);
window <- (seriesPositions(sp500.nreturns) >
timeDate("12/31/1995",format = "%m/%d/%Y"));
sp500.nreturns <- sp500.nreturns[window];
tmp <- extremalPP(sp500.nreturns,ne=100);
mod3a <- fit.seMPP(tmp,mark.influence=FALSE,std.errs=TRUE);
## Not run:

mod3b <- fit.seMPP(tmp,mark.influence=TRUE,std.errs=TRUE);
mod3c <- fit.seMPP(tmp,model="ETAS",mark.influence=FALSE,std.errs=TRUE);
mod3d <- fit.seMPP(tmp,model="ETAS",mark.influence=TRUE,std.errs=TRUE);
mod4a <- fit.seMPP(tmp,mark.influence=FALSE,predictable=TRUE,
std.errs=TRUE);
mod4b <- fit.seMPP(tmp,mark.influence=TRUE,predictable=TRUE,
std.errs=TRUE);
```

```

mod4c <- fit.seMPP(tmp,model="ETAS",mark.influence=FALSE,
                   predictable=TRUE,std.errs=TRUE);
mod4d <- fit.seMPP(tmp,model="ETAS",mark.influence=TRUE,
                   predictable=TRUE,std.errs=TRUE);
## End(Not run)

```

fit.sePP

*Fit Self-Exciting Process***Description**

fits fits self-exciting process to a point process object of class PP (unmarked) or MPP (marked)

Usage

```
fit.sePP(PP, model = "Hawkes", mark.influence = TRUE, std.errs = FALSE)
```

Arguments

PP	a point process object of class PP (unmarked) or MPP (marked)
model	(optional)name of self-exciting model: Hawkes or ETAS
mark.influence	(optional)whether marks of marked point process may influence the self-excitement
std.errs	(optional) whether standard errors should be computed VALUE

Details

see pages 306-307 of QRM

Value

a fitted self-exciting process object of class sePP

See Also

[fit.seMPP](#), [plot.sePP](#), [stationary.sePP](#)

Examples

```

data(sp500);
sp500.nreturns <- -mk.returns(sp500);
window <- (seriesPositions(sp500.nreturns) >
          timeDate("12/31/1995",format="%m/%d/%Y"));
sp500.nreturns <- sp500.nreturns[window];
tmp <- extremalPP(sp500.nreturns,ne=100);
mod2a <- fit.sePP(tmp,mark.influence=FALSE,std.errs=TRUE);
## Not run:
mod2b <- fit.sePP(tmp,mark.influence=TRUE,std.errs=TRUE);
mod2c <- fit.sePP(tmp,model="ETAS",mark.influence=FALSE,std.errs=TRUE);
mod2d <- fit.sePP(tmp,model="ETAS",mark.influence=TRUE,std.errs=TRUE);
## End(Not run)

```

fit.st

*Fit Student t Distribution***Description**

fits univariate Student's t distribution

Usage

```
fit.st(data)
```

Arguments

data vector of data

Details

see page 75 of QRM

Value

list containing parameter estimates, standard errors and details of fit

See Also

[fit.NH](#), [fit.mNH](#), [fit.mst](#)

Examples

```
data(DJ);
Ret.DJ <- mk.returns(DJ);
window1.start <- timeDate("01/01/1993",format="%m/%d/%Y");
window1.end <- timeDate("12/31/2000",format="%m/%d/%Y");
sample1 <- (seriesPositions(Ret.DJ) > window1.start
           & seriesPositions(Ret.DJ) < window1.end);
DJ30daily <- Ret.DJ[sample1,];
DJ30daily <- 100*seriesData(DJ30daily);
rseries <- DJ30daily[, "MSFT"];
mod.t <- fit.st(rseries);
## Not run:
mod.gauss <- fit.norm(rseries);
c(mod.gauss$ll.max, mod.t$ll.max);
mod.NIG <- fit.NH(rseries);
mod.hyp <- fit.NH(rseries,case="hyp");
c(mod.NIG$ll.max, mod.hyp$ll.max);
## End(Not run)
```

fit.tcopula.rank *Fit t Copula Using Rank Correlations*

Description

fits t copula to pseudo-copula data

Usage

```
fit.tcopula.rank(Udata, method="Kendall")
```

Arguments

Udata	matrix of pseudo-copula data where rows are vector observations with all values in unit interval
method	method to use for calculating rank correlations; default is "Kendall", which is theoretically justified

Details

see pages 229-231 of QRM

Value

list containing parameter estimates and details of fit

See Also

[fit.tcopula](#), [fit.gausscopula](#), [fit.Archcopula2d](#)

Examples

```
## Not run:
# Multivariate Fitting with Gauss and t: Simulation
# Create an equicorrelation matrix:
P <- equicorr(3,0.6);
set.seed(113);
#Generate a new set of random data from a t-copula (10df) with the same Sigma matrix:
Udatasim2 <- rcopula.t(1000,df=10,Sigma=P);
#Now fit the copula to the simulated data using (Kendall) rank correlations
#and the fit.tcopula.rank() method:
mod.t2 <- fit.tcopula.rank(Udatasim2);
mod.t2;
## End(Not run)
```

fit.tcopula

*Fit t Copula***Description**

fit t copula to pseudo-copula data

Usage

fit.tcopula(Udata)

Arguments

Udata matrix of pseudo-copula data where rows are vector observations with all values in unit interval

Details

see pages 235-236 of QRM

Value

list containing parameter estimates and details of fit

See Also[fit.gausscopula](#), [fit.Archcopula2d](#)**Examples**

```
data(ftse100);
data(smi);
TS1 <- cut(ftse100, "1990-11-08", "2004-03-25");
TS1Augment <- alignDailySeries(TS1, method="before");
TS2Augment <- alignDailySeries(smi, method="before");
INDEXES.RAW <- merge(TS1Augment, TS2Augment);
#Cleanup:
rm(TS1, TS1Augment, TS2Augment);
INDEXES <- mk.returns(INDEXES.RAW);
PARTIALINDEXES <- cut(INDEXES, "1993-12-31", "2003-12-31");
#Now create a data matrix from the just-created timeSeries
data <- seriesData(PARTIALINDEXES);
#Keep only the data items which are non-zero for both smi and ftse100
data <- data[data[,1]!=0 & data[,2] !=0,];
# Construct pseudo copula data. The 2nd parameter is MARGIN=2
#when applying to columns and 1 applied to rows. Hence this says to
#apply the 'edf()' empirical distribution function() to the columns
#of the data.
Udata <- apply(data, 2, edf, adjust=1);
#Fit a t-copula to the data:
mod.t <- fit.tcopula(Udata);
mod.t;
```

ftse100	<i>FTSE 100 Stock Market Index</i>
---------	------------------------------------

Description

closing values of FTSE 100 from 1990 to March 2004

FXGBP.RAW	<i>Sterling Exchange Rates</i>
-----------	--------------------------------

Description

exchange rates for major currencies (dollar, yen, euro, Swiss franc) against the pound for period 1987 to March 2004

GEV	<i>Generalized Extreme Value Distribution</i>
-----	---

Description

Cumulative probability, quantiles, density and random generation from 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, logvalue=FALSE)
rGEV(n, xi, mu=0, sigma=1)
```

Arguments

x	vector of values at which to evaluate density
q	vector of quantiles
p	vector of probabilities
n	sample size
xi	shape parameter
mu	location parameter
sigma	scale parameter
logvalue	whether or not log values of density should be returned (useful for ML)

Value

Probability (pGEV), quantile (qGEV), density (dGEV) or random sample (rGEV) for the GEV distribution with shape xi (with location parameter mu and location parameter sigma)

See Also

[fit.GEV](#), [fit.GPD](#), [pGPD](#)

Examples

```
## Not run:
#Build a loglikelihood function for MLE which can be called from optim() in fit.GEV()
negloglik <- function(theta)
{
  -sum(dGEV(excesses.nl, theta[1], abs(theta[2]), logvalue=TRUE));
}
## End(Not run)
```

ghyp

Univariate Generalized Hyperbolic Distribution

Description

Density and random number generation for univariate generalized hyperbolic distribution in new QRM (Chi-Psi-Gamma) parameterization. (The dispersion matrix Sigma is identically 1, i.e. a scalar 1.) See pp. 77-81 in QRM.

Usage

```
dghyp(x, lambda, chi, psi, mu=0, gamma=0, logvalue=FALSE)
rghyp(n, lambda, chi, psi, mu=0, gamma=0)
```

Arguments

x	vector of values at which to evaluate density
n	sample size
lambda	scalar mixing parameter
chi	scalar mixing parameter
psi	scalar mixing parameter
mu	location parameter
gamma	skewness parameter
logvalue	should log density be returned; default is FALSE

Details

See page 78 in QRM for joint density formula (3.30) with Sigma (dispersion matrix) the identity and d=1 (meaning a univariate distribution) applies.

The univariate QRM parameterization is defined in terms of parameters chi-psi-gamma instead of the alpha-beta-delta model used by Blaesild (1981) in earlier literature. If gamma is 0, we have a normal variance mixture where the mixing variable W has a GIG (generalized inverse gaussian) distribution with parameters lambda, chi, psi. This thickens the tail.

If gamma exceeds zero, we have a normal mean-variance mixture where the mean is also perturbed to equal $\mu + (W * \gamma)$ which introduces ASYMMETRY as well.

Values for lambda and mu are identical in both QRM and B parameterizations. Sigma does not appear in the parameter list since in the univariate case its value is identically 1.

Value

values of density or log-density (dghyp) or random sample (rghyp)

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.

Author(s)

documentation by Scott Ulman for R-language distribution

See Also

[dghypB](#), [besselM3](#), [dmghyp](#)

Examples

```
data(DJ);
#Make returns from timeSeries (the default is log-returns).
#Ret.DJ is a timeSeries class.
Ret.DJ <- mk.returns(DJ);
#In R, cut() method works by selecting data only between the
#'to' and 'from' dates. Hence we will use the remaining (cut) data from
#1993-01-01 to 2000-12-31. MUST use PRIOR DAY on 'from'
DJ30dailyTS <- cut(Ret.DJ, from="1992-12-31", to="2000-12-31");
DJ30daily <- 100*seriesData(DJ30dailyTS);
#Extract only the Microsoft returns as 'rseries'; remember this is a vector--not a timeSe
rseries <- DJ30daily[, "MSFT"];
#The default case for fit.NH() is NIG requiring lambda = -1/2.
mod.NIG <- fit.NH(rseries);
xvals <- seq(from=min(rseries), to=max(rseries), length=100);
yvals.NIG <- dghyp(xvals, lambda=-1/2, chi=mod.NIG$par.ests[1],
  psi=mod.NIG$par.ests[2], mu=mod.NIG$par.ests[3], gamma=mod.NIG$par.ests[4]);
```

ghypB

Univariate Generalized Hyperbolic Distribution B

Description

Density and random number generation for univariate generalized hyperbolic distribution in standard parameterization (alpha-beta-delta). (The dispersion matrix Sigma is identically 1, i.e. a scalar 1.) See pp. 77-81 in QRM.

Usage

```
dghypB(x, lambda, delta, alpha, beta=0, mu=0, logvalue=FALSE)
rghypB(n, lambda, delta, alpha, beta=0, mu=0)
```

Arguments

x	values at which to evaluate density
n	sample size
lambda	scalar parameter
delta	scalar parameter
alpha	scalar parameter
beta	skewness parameter
mu	location parameter
logvalue	Should log density be returned? Default is FALSE

Details

See page 78 in QRM for joint density formula (3.30) with Sigma (dispersion matrix) the identity and d=1 (meaning a univariate distribution) applies.

The B parameterization corresponds to the original alpha-beta-delta model used by Blaesild (1981) in earlier literature. If gamma is 0, we have a normal variance mixture defined by the parameters alpha-beta-delta. This thickens the tail.

If gamma exceeds zero, we have a normal mean-variance mixture where the mean is also perturbed to equal $\mu + (W * \gamma)$ which introduces ASYMMETRY as well.

Values for lambda and mu are identical in both QRM and B parameterizations.

Sigma does not appear in parameter list since in the univariate case its value is assumed to be identically 1.

Value

values of density or log-density (dghypB) or random sample (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.

Author(s)

documentation by Scott Ulman for R-language distribution

See Also

[dghyp](#), [besselM3](#)

GPD

*Generalized Pareto Distribution***Description**

Cumulative probability, quantiles, density and random generation from the generalized Pareto distribution.

Usage

```
pGPD(q, xi, beta=1)
qGPD(p, xi, beta=1)
dGPD(x, xi, beta=1, logvalue=FALSE)
rGPD(n, xi, beta=1)
```

Arguments

x	vector of values at which to evaluate density
q	vector of quantiles
p	vector of probabilities
n	sample size
xi	shape parameter
beta	scale parameter
logvalue	whether or not log values of density should be returned (useful for ML)

Value

Probability (pGPD), quantile (qGPD), density (dGPD) or random sample (rGPD) for the GPD with shape xi.

Author(s)

documentation by Scott Ulman for R-language distribution

See Also

[fit.GPD](#), [pGEV](#), [fit.GEV](#)

Examples

```
## Not run:
#Build a loglikelihood function for MLE which can be called from optim()
negloglik <- function(theta)
{
  -sum(dGPD(excesses.nl, theta[1], abs(theta[2]), logvalue=TRUE));
}
## End(Not run)
```

Gumbel

Gumbel Distribution

Description

Density, quantiles, and cumulative probability of the Gumbel distribution.

Usage

```
dGumbel(x, logvalue=FALSE)
qGumbel(p)
pGumbel(q)
```

Arguments

x	vector of values at which to evaluate density
q	vector of quantiles
p	vector of probabilities
logvalue	whether or not log values of density should be returned (useful for ML)

Value

Probability (pGumbel), quantile (qGumbel) and density (dGumbel) for the Gumbel distribution with shape xi.

Examples

```
## Not run:
#If the shape parameter of the GEV is xi=0, then the distribution collapses to
#the Gumbel with parameters properly scaled:
x <- 2.5; mu <- 1.0; sigma = 2.5;
xx = x - mu/sigma;
if(xi == 0) dGEVvalue <- dGumbel(xx, logvalue=TRUE) - log(sigma);
## End(Not run)
```

hessb

Approximate Hessian Matrix

Description

calculates a numerical approximation of Hessian matrix

Usage

```
hessb(f, x, ep=0.0001, ...)
```

Arguments

f	function
x	value of function at which to approximate Hessian
ep	precision for numerical differencing
...	other arguments of function f

Value

matrix of approximate second derivatives

Examples

```
## Not run:
#within fit.NH we approximate 2nd derivatives to calc standard errors
if(se)
{
  hessmatrix <- hessb(negloglik,par.ests)
  vcmatrix <- solve(hessmatrix)
  par.ses <- sqrt(diag(vcmatrix))
  names(par.ses) <- names(par.ests)
  dimnames(vcmatrix) <- list(names(par.ests), names(par.ests))
}
else
{
  par.ses <- NA
  vcmatrix <- NA
}
## End(Not run)
```

hillPlot

Create Hill Plot

Description

Plot the Hill estimate of the tail index of heavy-tailed data, or of an associated quantile estimate.

Usage

```
hillPlot(data, option = c("alpha", "xi", "quantile"), start = 15,
  end = NA, reverse = FALSE,
  p = NA, ci = 0.95, auto.scale = TRUE, labels = TRUE, ...)
```

Arguments

data	data vector
option	whether "alpha", "xi" (1/alpha) or "quantile" (a quantile estimate) should be plotted
start	lowest number of order statistics at which to plot a point
end	highest number of order statistics at which to plot a point
reverse	whether plot is to be by increasing threshold (TRUE) or increasing number of order statistics (FALSE)

<code>p</code>	probability required when option "quantile" is chosen
<code>ci</code>	probability for asymptotic confidence band; for no confidence band set <code>ci</code> to zero
<code>auto.scale</code>	whether or not plot should be automatically scaled; if not, <code>xlim</code> and <code>ylim</code> graphical parameters may be entered
<code>labels</code>	whether or not axes should be labelled
<code>...</code>	other graphics parameters

Details

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 ξ . See pp. 286-289 in QRM. Especially note that Example 7.28 suggests the best estimates occur when the threshold is very small, perhaps 0.1 statistics in a sample of size 1000. Hence you should NOT be using a 95 estimates.

Value

None

Author(s)

documentation by Scott Ulman for R-language distribution

See Also

[xipLOT](#), [plotTail](#)

Examples

```
## Not run:
data(danish);
#Run hillPlot to show what happens with the Hill Plot.
#See Example 7.27, p. 287 in QRM
hillPlot(danish, option = "alpha", start = 5, end = 250, p = 0.99);
hillPlot(danish, option = "alpha", start = 5, end = 60, p = 0.99);
## End(Not run)
```

hsi

Hang Seng Stock Market Index

Description

This timeSeries data set provides the daily closing values of the Hanh Seng Index from 1994 to March 2004

jointnormalTest	<i>Test of Multivariate Normality</i>
-----------------	---------------------------------------

Description

provides test of multivariate normality based on analysing Mahalanobis distances

Usage

```
jointnormalTest(data, dist="chisquare")
```

Arguments

data	matrix of data with each row representing an observation
dist	"chisquare" performs test against chi-squared distribution, which is an approximation; "beta" performs test against a scaled beta

Details

see pages 69-70 of QRM

Value

p-value for Kolmogorov-Smirnov test

Side Effects

a QQplot against the reference distribution is created

See Also

[MardiaTest](#)

Examples

```
data(DJ);
Ret.DJ <- mk.returns(DJ);
selection1 <- c("AXP", "EK", "BA", "C", "KO", "MSFT",
               "HWP", "INTC", "JPM", "DIS");
partialDJ30dailyTS <- Ret.DJ[,selection1];
#Choose only the data from 1/1/1993 to 12/31/2000. Note
#'from' date must be day prior to desired start date.
partialDJ30daily <- cut(partialDJ30dailyTS, from="1992-12-31",
                       to="2000-12-31");
partialDJ30dailyMatrix <- seriesData(partialDJ30daily);
#Note the tests on the ten stocks selected from DJ30 fail the test miserably
#except possibly the quarterly values. The QQ plots are very revealing.
#See p. 72 in QRM Book.
jointnormalTest(partialDJ30dailyMatrix);
```

Kendall

Kendall's Rank Correlation

Description

calculates a matrix of Kendall's rank correlations

Usage

```
Kendall(data, noforce=TRUE)
```

Arguments

data	data matrix
noforce	must be set to FALSE if you really want to calculate Kendall's rank correlations for more than 5000 data (which will be slow)

Details

see pages 229-230 in QRM

Value

matrix of rank correlations

See Also

[Spearman](#), [fit.tcopula.rank](#)

Examples

```
data <- rmnorm(1000, d=3, rho=0.5);
Kendall(data);
```

kurtosisSPlus

S-Plus Version of Kurtosis which differs from the R-versions

Description

The values calculated by R and S-Plus differ when we use the call `kurtosis(x, method="moment")` which causes serious consequences in the `fit.NH()` function call. Hence we introduce the S-Plus version here. S-Plus has only the "moment" and "fisher" methods. R has a 3rd type, the "excess" which should parallel the R "moment" type but fails.

Usage

```
kurtosisSPlus(x, na.rm = FALSE, method = "fisher")
```


Arguments

<code>x</code>	data vector
<code>na.rm</code>	TRUE or FALSE indicating whether to remove any NA values from the data vector
<code>method</code>	either the 'moment' or 'fisher' method

Details

use R-code which reflects the way S-Plus calculates Kurtosis so we match up the answer regardless of whether using S-Plus or R

Value

a single number reflecting the kurtosis statistic as calculated via the S-Plus method (either "moment" or "fisher")

Author(s)

documentation by Scott Ulman for R-language distribution

lbeta	<i>Log Beta Function</i>
-------	--------------------------

Description

calculates logarithm of beta function

Usage

```
lbeta(a, b)
```

Arguments

<code>a</code>	vector of values of argument 1
<code>b</code>	vector of values of argument 2

Value

vector of values of logarithm of beta function

See Also

For other special functions, see [Special](#)

MardiaTest

Mardia's Tests of Multinormality

Description

conducts Mardia's tests of multinormality based on multivariate skewness and kurtosis statistics

Usage

```
MardiaTest(data)
```

Arguments

data data matrix

Details

see page 70 of QRM

Value

vector consisting of skewness statistic, p-value for skewness statistic, kurtosis statistic and p-value for kurtosis statistic

See Also

[jointnormalTest](#)

Examples

```
data <- rmnorm(1000,d=10,rho=0.6);
MardiaTest(data);
```

MCECM.Qfunc

Optimization Function for MCECM Fitting of GH

Description

a functional form that must be optimized when fitting members of generalized hyperbolic family with an MCECM algorithm

Usage

```
MCECM.Qfunc(lambda, chi, psi, delta, eta, xi)
```

Arguments

lambda	lambda parameter
chi	chi parameter
psi	pi parameter
delta	delta statistic
eta	eta statistic
xi	xi statistic

Details

this is the Q2 function on page 82 of QRM

Value

value of function

See Also

[MCECMupdate](#), [EMupdate](#), [fit.mNH](#)

MCECMupdate	<i>MCECM Update Step for Generalized Hyperbolic</i>
-------------	---

Description

updates estimates of mixing parameters in EM estimation of generalized hyperbolic

Usage

```
MCECMupdate(data, mix.pars, mu, Sigma, gamma, optpars, optfunc, xieval=FALSE)
```

Arguments

data	data matrix
mix.pars	current values of lambda, chi and psi
mu	current value of mu
Sigma	current value of Sigma
gamma	current value of gamma
optpars	parameters we need to optimize over (may differ from case to case)
optfunc	the function to be optimized
xieval	is it necessary to evaluate the log moment xi?

Details

see Algorithm 3.14, steps (5) and (6) on page 83 of QRM

Value

list containing new estimates of mixing parameters as well as convergence information for optimization

See Also

[fit.mNH](#)

MEplot

Sample Mean Excess Plot

Description

Plots sample mean excesses over increasing thresholds.

Usage

```
MEplot(data, omit = 3, labels=TRUE, ...)
```

Arguments

data	data vector or time series
omit	number of upper plotting points to be omitted
labels	whether or not axes are to be labelled
...	further parameters of MEplot function

Details

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.

See Also

[fit.GPD](#)

Examples

```
# Sample mean excess plot of heavy-tailed Danish fire insurance data
data(danish);
MEplot(danish);
```

mghyp

*Multivariate Generalized Hyperbolic Distribution***Description**

Density and random number generation for density of multivariate generalized hyperbolic distribution in new QRM (Chi-Psi-Sigma- Gamma) parameterization. Note Sigma is the dispersion matrix. See pp. 77-81.

Usage

```
dmghyp(x, lambda, chi, psi, mu, Sigma, gamma, logvalue=FALSE)
rmghyp(n, lambda, chi, psi, Sigma=equicorr(d, rho), mu=rep(0, d),
       gamma=rep(0, d), d=2, rho=0.7)
```

Arguments

x	matrix with n rows and d columns; density is evaluated at each vector of row values
lambda	scalar parameter
chi	scalar parameter
psi	scalar parameter
mu	location vector
Sigma	dispersion matrix
d	dimension of distribution
rho	correlation value to build equicorrelation matrix
gamma	vector of skew parameters
logvalue	should log density be returned; default is FALSE
n	length of vector

Details

See page 78 in QRM for joint density formula (3.30) with Sigma a d-dimensional dispersion matrix ($d > 1$) consistent with a multivariate distribution). This is a more intuitive parameterization of the alpha-beta-delta model used by Blaesild (1981) in earlier literature since it associates all parameters with mixtures of both mean and variance. Here gamma is assumed equal to 0 so we have a normal variance mixture where the mixing variable W has a GIG (generalized inverse gaussian) distribution with parameters lambda, chi, psi. This thickens the tail.

If gamma exceeds zero, we have a normal mean-variance mixture where the mean is also perturbed to equal $\mu + (W * \gamma)$ which introduces ASYMMETRY as well.

The default $d=2$ for the random generator gives a two-dimensional matrix of n values.

See pp. 77-81 of QRM and appendix A.2.5 for details.

Value

values of density or log-density or randomly generated values

Note

See page 78 in QRM; if gamma is a zero vector distribution is elliptical and dsmghyp is called. If $\lambda = (d+1)/2$, we drop generalized and call the density a d-dimensional hyperbolic density. If $\lambda = 1$, the univariate marginals are one-dimensional hyperbolics. If $\lambda = -1/2$, distribution is NIG (normal inverse gaussian). If λ greater than 0 and $\chi = 0$, we get the VG (variance gamma) If we can define a constant ν such that $\lambda = (-1/2)*\nu$ AND $\chi = \nu$ then we have a multivariate skewed-t distribution. See p. 80 of QRM for details.

Author(s)

documentation by Scott Ulman for R-language distribution

See Also

[dsmghyp](#), [dmt](#), [dmnorm](#)

Examples

```
## Not run:
Sigma <- diag(c(3,4,5)) %% equicorr(3,0.6)
      %% diag(c(3,4,5));
mu <- c(1,2,3);
ghdata <- rmghyp(n=1000,lambda=0.5,chi=1,psi=1,Sigma,mu);
### (Multivariate generalized) Hyperbolic distribution: visualization with
# PERSPECTIVE or CONTOUR plots
par(mfrow=c(2,2));
ll <- c(-4,4);
#pass the multivariate generalized hyperbolic density to be plotted:
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=ll,ypts=ll,mu=c(0,0),
           Sigma=equicorr(2,-0.7),lambda=1,chi=1,psi=1,gamma=c(0.5,-0.5));
par(mfrow=c(1,1))
## End(Not run)
```

mk.its.exceedances.tS

*Extract Exceedances over a Threshold into Irregular Time Series for
Plotting when Input is TimeSeries*

Description

Builds an irregular time series object of the 'its' class to hold the exceedances above a specified threshold ordered according to date. The 'its' time series may then be plotted easily to show clustering effects ("volatility clustering")

Usage

```
mk.its.exceedances.tS(timeseries, col = 1, thresholdValue)
```

Arguments

<code>timeseries</code>	fCalendar timeSeries object containing data set for which you wish to plot exceedances over a threshold
<code>col</code>	the column number of the timeSeries data from which you wish to extract the exceedances
<code>thresholdValue</code>	the minimum data value above which you want to collect exceedances into the irregular time series ('its' object); calculate by calling QRM function <code>findthreshold()</code>

Details

This function passes a timeSeries object containing both the Data and positions slots for associating dates with Data when building the irregular time series.

See `mk.its.exceedances.vector()` for a method which passes a separate vector of data and a matching-length position slot from a different timeSeries object.

Value

An irregular time series ('its' object requiring the 'its' package to be loaded) with exceedances ordered according to date. Input 'its' into a `plot()` function to graph the exceedances.

Author(s)

documentation by Scott Ulman for R-language distribution

References

see pages 117-8 of QRM. Chapter 4 discusses the usefulness of graphing exceedances to display "volatility clustering".

See Also

`timeSeriesClass`, `mk.its.exceedances.vector`, `mk.returns`

Examples

```
data(sp500);
#Select data from the SP500 INDEX time series between
#01/01/1998 and 12/31/2003
index <- cut(sp500,"1998-01-01", "2003-12-31");
#Create a return time series using log differences.
rseries <- mk.returns(index);
kval <- 50;
#set threshold so we have 50 observations above threshold
#Find the threshold associated with the top 50 observations:
upper <- findthreshold(rseries@Data ,kval)
#Create an 'its' (irregular time series) associated with the
#observations above the threshold 'upper'.
itsSample <- mk.its.exceedances.tS(rseries,1,upper);
#Plot the irregular time series of exceedances
plot(itsSample,type="h", main="Clustering Evidence-S&P500
      Index:1998-2003",ylab="Large Returns");
## Not run:
```

```
#3rd stock in series of DJ 30 stocks from timeSeries DJ containing
#all 30 stocks (not an index)
newSeries <- mk.returns(DJ);
u3 <- findthreshold(newSeries@Data[,3],kval);
#use 3rd column in Data slot
itsSample2 <- mk.its.exceedances.tS(newSeries, 3, u3);
#pass 3 for third column in Data slot
plot(itsSample2,type="h", main="Clustering Evidence-DJ[,3]",
      ylab="Large Returns");
## End(Not run)
```

mk.its.exceedances.vector

*Extract Exceedances over Threshold into Irregular Time Series for
Plotting when Input is Vector*

Description

Builds an irregular time series object of the ‘its’ class to hold the exceedances above a specified threshold ordered according to date. The ‘its’ time series may then be plotted easily to show clustering effects (“volatility clustering”).

Usage

```
mk.its.exceedances.vector(datavector, paralleltimeseriesPos, thresholdValue)
```

Arguments

datavector	a data vector for which you wish to plot exceedances over a threshold
paralleltimeseriesPos	a vector of length equal to the datavector containing the positions (time-date) attribute from an R-Metrics type timeSeries object
thresholdValue	the minimum data value above which you want to collect exceedances into the irregular time series (‘its’ object); calculate by calling QRM function findthreshold()

Details

This function passes a vector (not a matrix or timeSeries) containing the data. It passes a separate positions slot from a timeSeries for associating dates with Data when building the irregular time series.

See *mk.its.exceedances.tS()* for a method which passes a timeSeries object containing both data and position.

Value

An irregular time series ‘its’ object requiring the ‘its’ package to be loaded with exceedances ordered according to date. Input ‘its’ into plot function to graph the exceedances

Author(s)

documentation by Scott Ulman for R-language distribution

References

See pages 117-8 of QRM. Chapter 4 discusses the usefulness of graphing exceedances to display “volatility clustering”.

See Also

`timeSeriesClass`, `mk.its.exceedances.tS`, `mk.returns`

Examples

```
## Not run:
#In this example, a simulated data series tdata was
#generated using parameters derived from fitting a timeSeries
#for the SP500 index called rseries over a specified period.
#tdata was randomly generated from the t-distribution using
#parameters estimated from the actual SP500 time series of data
#returns (log differences). Thus tdata is a VECTOR (one column)
#rather than a timeSeries or matrix. If you have a matrix, pass
#only the single column you want to analyze.
#set threshold so we have 50 observations above threshold from tdata.
kval <- 50;
upper <- findthreshold(tdata,kval);
#Create an its (irregular time series) associated with observations
#above threshold upper. We pass the simulated data vector tdata,
#the dates (positions slot) from a separate timeseries over the same
#period for which we are considering the simulated data,
#and the threshold value for our data vector.
itsSample <- mk.its.exceedances.vector(tdata,rseries@positions,upper);
#Plot the irregular time series of exceedances
plot(itsSample,type="h", main="Clustering Evidence-Simulated t data",
     ylab="Large Returns");
## End(Not run)
```

mk.oldts

Make Old Style Time Series

Description

This function converts a ‘timeSeries’ object (from package fCalendar in R) to an ‘its’ object (an irregular times series from package ‘its’ in R). The conversion is useful when plotting irregular (not-evenly spaced in time) series.

Usage

```
mk.oldts(timeseries)
```

Arguments

`timeseries` a timeSeries object

Details

this is occasionally useful for plotting

Value

an old-style 'its' object

<code>mk.returns</code>	<i>Make Financial Return Data</i>
-------------------------	-----------------------------------

Description

makes financial return data from asset price data

Usage

```
mk.returns(tpdata, type="log")
```

Arguments

<code>tpdata</code>	a timeSeries object containing prices
<code>type</code>	whether "log" or "relative" returns should be constructed

Value

a timeSeries object containing returns

See Also

[timeSeriesClass](#), [TimeSeriesClassRMetrics](#)

Examples

```
data(ftse100);
ftse100.r <- mk.returns(ftse100);
```

<code>momest</code>	<i>Moment Estimator of Default Probabilities</i>
---------------------	--

Description

calculates moment estimator of default probabilities and joint default probabilities for a homogeneous group

Usage

```
momest(data, trials, limit=10.)
```

Arguments

data	vector of numbers of defaults in each time period
trials	vector of group sizes in each time period
limit	maximum order of joint default probability to estimate

Details

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

Value

vector of default probability and joint default probability estimates

Author(s)

documentation by Scott Ulman for R-language distribution

See Also

[fit.binomialBeta](#), [fit.binomialLogitnorm](#), [fit.binomialProbitnorm](#)

Examples

```
## Not run:
#MODEL RISK See especially Section 8.4.6 on p. 364 of QRM book
spdata.raw;
attach(spdata.raw);
#momest() is an internal function in functionsCredit.R to
#calculate moment estimators for default probabilities. The first
#parameter input is a vector containing the number of defaults in
#each time period; the 2nd parameter input is a vector containing the
#number of credits in the group during the time period.
momest(Bdefaults,Bobligors);
#The values calculated from momest(Bdefaults,Bobligors) are the
#parameter estimates shown in Table 8.6, p.365 of QRM book under the
#model column labeled 'B'
#The first value returned is the probability of a single default.
pi.B <- momest(Bdefaults, Bobligors)[1]; #one obligor defaulting pi = .04896
#second value returned is probability of joint default probability for two firms.
pi2.B <- momest(Bdefaults, Bobligors)[2]; #two obligors defaulting jointly pi2 = .0031265
## End(Not run)
```

nasdaq

NASDAQ Stock Market Index

Description

This timeSeries data set provides the daily closing values of the NASDAQ index from 1994 to March 2004

nikkei	<i>Nikkei Stock Market Index</i>
--------	----------------------------------

Description

This timeSeries data set provides the daily closing values of the Nikkei index from 1994 to March 2004

Pconstruct	<i>Assemble a Correlation Matrix for ML Copula Fitting</i>
------------	--

Description

takes a vector of values representing the terms of a lower triangular matrix A with ones on the diagonal and calculates the correlation matrix corresponding to the covariance matrix AA'

Usage

```
Pconstruct(theta)
```

Arguments

theta elements of a lower triangular matrix A with ones on the diagonal

Details

see page 235 in QRM

Value

correlation matrix corresponding to covariance matrix AA'

See Also

[Pdeconstruct](#), [fit.gausscopula](#), [fit.tcopula](#)

Examples

```
P <- Pconstruct(c(1,2,3,4,5,6));
## Not run:
eigen(P);
Pdeconstruct(P);
## End(Not run)
```

Pdeconstruct

Disassemble a Correlation Matrix for ML Copula Fitting

Description

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 correlation matrix corresponding to the covariance matrix AA'

Usage

```
Pdeconstruct (P)
```

Arguments

P a correlation matrix

Details

see page 235 in QRM

Value

elements of a lower-triangular matrix

See Also

[Pconstruct](#), [fit.gausscopula](#), [fit.tcopula](#)

Examples

```
P <- Pconstruct (c (1, 2, 3, 4, 5, 6)) ;
Pdeconstruct (P) ;
```

plot.MPP

Plot Marked Point Process

Description

creates a picture of a marked point process

Usage

```
plot.MPP (x, ...)
```

Arguments

x a point process object of class MPP (marked point process)
 \dots further parameters which may be passed to the plot function (see R help about the plot function for further information)

Details

Creates an appropriate plot on graphical device. The input variable PP will be internally separated into x and y values to pass to plot()

Author(s)

documentation by Scott Ulman for R-language distribution

See Also

[extremalPP](#)

Examples

```
data(sp500);
sp500.nreturns <- -mk.returns(sp500);
window <- (seriesPositions(sp500.nreturns) >
  timeDate("12/31/1995", format = "%m/%d/%Y")) &
  (seriesPositions(sp500.nreturns) <
    timeDate("01/01/2004", format = "%m/%d/%Y"));
sp500.nreturns <- sp500.nreturns[window];
tmp <- extremalPP(sp500.nreturns, ne=100);
plot.MPP(tmp);
```

plot.PP

Plot Point Process

Description

creates a picture of an unmarked point process.

Usage

```
plot.PP(x, ...)
```

Arguments

x	a point process object of class PP which must be unmarked
...	further parameters which may be passed to the plot function (see R help about the plot function for further information)

Details

Creates an appropriate plot on graphical device. The input variable x will be internally separated into starttime and endtime values to pass to plot.stepfun()

Author(s)

documentation by Scott Ulman for R-language distribution

See Also

[extremalPP](#), [unmark](#)

Examples

```

data(sp500);
sp500.nreturns <- -mk.returns(sp500);
window <- (seriesPositions(sp500.nreturns) >
  timeDate("12/31/1995",format = "%m/%d-%Y")) &
  (seriesPositions(sp500.nreturns) <
    timeDate("01/01/2004",format = "%m/%d/%Y"));
sp500.nreturns <- sp500.nreturns>window];
#The following functions are contained in functionsHawkes.R.
#Plot the 100 largest exceedances. Create an MPP (marked point process) class
tmp <- extremalPP(sp500.nreturns,ne=100);
#Be sure to graph with plot.PP instead of plot.MPP:
tmp2 <- unmark(tmp);
plot.PP(tmp2);

```

plot.sePP

*Plot Self-Exciting Point Process***Description**

plots a fitted self-exciting point process model, either unmarked or marked

Usage

```
plot.sePP(x, ...)
```

Arguments

<code>x</code>	a fitted self-exciting point process model created by either <code>fit.sePP</code> or <code>fit.seMPP</code> . 'x' is the generic value passed to all S3 plot functions.
<code>...</code>	further parameters which may be passed to the plot function (see R help about the plot function for further information)

Details

Creates an appropriate plot on graphical device. The input variable will be internally separated into x and y values to pass to `plot()`

Author(s)

documentation by Scott Ulman for R-language distribution

See Also

[fit.sePP](#), [fit.seMPP](#)

Examples

```
data(sp500);
sp500.nreturns <- -mk.returns(sp500);
window <- (seriesPositions(sp500.nreturns) > timeDate("12/31/1995",
format = "%m/%d/%Y")) & (seriesPositions(sp500.nreturns) <
timeDate("01/01/2004",format = "%m/%d/%Y"));
sp500.nreturns <- sp500.nreturns>window];
tmp <- extremalPP(sp500.nreturns,ne=100);
mod2a <- fit.sePP(tmp,mark.influence=FALSE,std.errs=TRUE);
plot.sePP(mod2a);
```

plot.timeSeries	<i>Plot multiple timeSeries objects on same graph</i>
-----------------	---

Description

Plots timeSeries objects with time on the x-axis and corresponding values on the y-axis. Builds a reference grid by default. This function converts the timeSeries input object to an its (irregular time series) and hence requires the 'its' library to be loaded.

Usage

```
plot.timeSeries(x, reference.grid = TRUE, lty = 1, ...)
```

Arguments

x	a timeSeries object (may have multiple series)
reference.grid	TRUE if you desire the graph to have a grid
lty	line type for graphed series
...	any other plot attributes which should be passed

Details

Not working properly in fCalendar 240.10068 so instituted here. We will deprecate this function eventually.

Value

Describe the value returned If it is a LIST, use

comp1	Description of 'comp1'
comp2	Description of 'comp2'
...	

Author(s)

documentation by Scott Ulman for R-language distribution

Examples

```
data(ftse100);
data(smi);
TS1 <- cut(ftse100, "1990-11-08", "2004-03-25");
TS1Augment <- alignDailySeries(TS1, method="before");
TS2Augment <- alignDailySeries(smi, method="before");
INDEXES.RAW <- merge(TS1Augment, TS2Augment);
#Cleanup:
rm(TS1, TS1Augment, TS2Augment);
plot.timeSeries(INDEXES.RAW);
```

plot.timeSeriesIts *Plot single or multiple timeSeries objects on same graph*

Description

Plots individual or multiple timeSeries objects with time on the x-axis and corresponding series values on the y-axis. Builds a reference grid by default. This function converts the timeSeries input object to an 'its' (irregular time series object) and hence requires the 'its' library to be loaded. Use plot.timeSeriesIts() whenever the timeSeries x has multiple columns of data.

Usage

```
plot.timeSeriesIts(x, reference.grid = TRUE, lty = 1, ...)
```

Arguments

x	a timeSeries object (may have multiple series)
reference.grid	TRUE if you desire the graph to have a grid
lty	line type for graphed series
...	any other plot attributes which should be passed

Details

plot.timeSeries() is not working properly in fCalendar 240.10068 for multiple timeSeries on the same graph. Hence we've reinstituted an older version and named it plot.timeSeriesIts() to indicate we must have the 'its' library loaded. We may deprecate this function eventually.

Author(s)

documentation by Scott Ulman for R-language distribution

Examples

```
#Plot two time series (ftse100 and smi indices) on same graph
data(ftse100);
data(smi);
TS1 <- cut(ftse100, "1990-11-08", "2004-03-25");
TS1Augment <- alignDailySeries(TS1, method="before");
TS2Augment <- alignDailySeries(smi, method="before");
INDEXES.RAW <- merge(TS1Augment, TS2Augment);
```

```
#Cleanup:
rm(TS1, TS1Augment, TS2Augment);
plot.timeSeriesIts(INDEXES.RAW);
```

```
plotFittedGPDvsEmpiricalExcesses
```

Graphically Compare Empirical Distribution of Excesses and GPD Fit

Description

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

Usage

```
plotFittedGPDvsEmpiricalExcesses(data, threshold = NA, nextremes = NA)
```

Arguments

data	data vector or times series
threshold	a threshold value (either this or "nextremes" must be given but not both)
nextremes	the number of upper extremes to be used (either this or "threshold" must be given but not both)

Details

See graphs 7.4(c) and 7.5(c) in QRM, pp. 281-2.

Value

a plot showing empirical cdf of excesses vs points fitted to the estimated GPD for excesses

See Also

[fit.GPD](#), [plotTail](#), [MEplot](#), [xiplot](#)

Examples

```
## Not run:
plotFittedGPDvsEmpiricalExcesses(danish, threshold=10);
plotFittedGPDvsEmpiricalExcesses(danish, nextremes=109);
plotFittedGPDvsEmpiricalExcesses(danish@Data, nextremes=109);
plotFittedGPDvsEmpiricalExcesses(danish@Data, threshold=10);
## End(Not run)
```

plotTail	<i>Tail Plot of GPD Model</i>
----------	-------------------------------

Description

plots the tail estimate corresponding to a GPD model of excesses over a high threshold

Usage

```
plotTail(object, extend=2, fineness=1000, ...)
```

Arguments

object	result of fitting GPD to excesses over a high threshold
extend	how far plot should extend expressed as multiple of largest data value
fineness	number of points at which to evaluate the tail estimate
...	additional arguments for plot function

Details

see pages 282-284 in QRM

Side Effects

a plot of the tail estimate is produced on a graphical device

See Also

[fit.GPD](#), [MEplot](#)

Examples

```
data(danish);
mod <- fit.GPD(danish,10);
mod$par.ests;
plotTail(mod);
```

probitnorm	<i>Probit-Normal Distribution</i>
------------	-----------------------------------

Description

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

Usage

```
dprobitnorm(x, mu, sigma)
pprobitnorm(q, mu, sigma)
rprobitnorm(n, mu, sigma)
```

Arguments

x	vector of values in unit interval at which to evaluate density
q	vector of values in unit interval at which to evaluate cumulative probabilities
n	sample size
mu	scalar parameter
sigma	scalar parameter

Details

see pages 353-354 in QRM

Value

vector of density values (dprobitnorm), cumulative probabilities (pprobitnorm) or random sample (rprobitnorm)

Author(s)

documentation by Scott Ulman for R-language distribution

See Also

[dbeta](#), [dclaytonmix](#)

Examples

```
## Not run:
#MODEL RISK See especially Section 8.4.6 on p. 364 of QRM book
spdata.raw;
attach(spdata.raw);
pi.B <- momest(Bdefaults, Bobligors)[1]; #one obligor defaulting pi = .04896
#second value returned is probability of joint default probability for two firms.
pi2.B <- momest(Bdefaults, Bobligors)[2]; #two obligors defaulting jointly pi2 = .0031265
#Build 1000 equally-spaced value on unit interval as multiples of .000999; discard
#all values except those below 0.25 because we want to look at the tail, i.e. Q > 0.25
#via the tail function [1 - P(Q <= 0.25)]
# Model Risk Experiment
# Calibrate a 1-Factor Creditmetrics (probitnormal) model to pi.B and pi2.B for all model
#The following values are shown in Table 8.6, column B, row labeled 'Probit-normal'.
#In other words, find the probitnorm mu and sigma values which give same probabilities as
#momest()
probitnorm.pars <- cal.probitnorm(pi.B, pi2.B);
probitnorm.pars;
q <- (1:1000)/1001;
q <- q[q<0.25];
# We could also look at mixing densities. Remember that density values for continuous
#variables may exceed 1 since they give an approximation for the change in the cdf value
#as we change the x value. Hence if the cdf increases by 0.2 as we increase x from 0.1 to
#0.2, the density should be about 2.0 (dF(x)/dx).
d.probitnorm <- dprobitnorm(q, probitnorm.pars[1], probitnorm.pars[2]);
## End(Not run)
```

psifunc

*Psi or Digamma Function***Description**

calculates psi or digamma function

Usage

```
psifunc(x=2, logvalue=FALSE)
```

Arguments

x	vector of values at which to calculate function
logvalue	whether logarithm of function should be returned; default is FALSE

Value

vector of values of psi or digamma function

See Also

[besselM3](#), [gamma](#)

QQplot

*Generic Quantile-Quantile Plot***Description**

constructs a quantile-quantile plot against a given reference distribution

Usage

```
QQplot(data, position=0.5, reference="normal", ...)
```

Arguments

data	vector of data
position	determines the plotting positions (see ppoints in R-help)
reference	name of reference distribution (normal, exp, student, etc.)
...	any further parameters required by 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. See Beirlant et al, "Statistics of Extremes", Chapter 1.2.1 for descriptions of various QQ plots.

Value

NULL returned

Side Effects

QQ-plot is created on graphical device

See Also

[qqnorm](#), [dghyp](#)

Examples

```
QQplot(rnorm(1000), ref="normal");
QQplot(rexp(1000), ref="exp");
```

qst

Student's t Distribution (3 parameter)

Description

Quantiles for 3-parameter version of Student's t distribution

Usage

```
qst(p, df, mu=0, sigma=1, scale=FALSE)
```

Arguments

p	vector of probabilities
df	vector of degrees of freedom
mu	vector of location parameters
sigma	vector of scale parameters
scale	whether distribution should be scaled so that mu and sigma are mean and standard deviation; default is FALSE

Value

quantiles for Student's t distribution

See Also

[qt](#), [ESst](#)

Examples

```
#Set up the quantile probabilities
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);
#Now look at VaR for student t with 4 degrees of freedom:
VaR.t4 <- qst(p, 4, sigma=sigma, scale=TRUE);
```

rAC

*Generate Archimedean Copula***Description**

generates data from a multivariate Archimedean copula with arbitrary dimension using the mixture construction of Marshall and Olkin

Usage

```
rAC(name, n, d, theta)
```

Arguments

name	name of the Archimedean copula from following list: clayton, gumbel, frank, BB9
n	number of realizations
d	dimension of copula
theta	parameter(s) of copula

Details

this function may easily be augmented with further Archimedean copulas

Value

a matrix of dimension n times d where rows are realizations

See Also

`rcopula.clayton`, `rcopula.frank`, `rcopula.gumbel`, `rcopula.gauss`, `rcopula.t`

Examples

```
#simulate values from Archimedean copula of type gumbel
rAC("gumbel", n=3000, d=7, theta =3);
```

rBB9Mix

*Mixture Distribution Yielding BB9 Copula***Description**

generates random sample from mixing distribution required for sampling from Joe's BB9 copula using Laplace transform method

Usage

```
rBB9Mix(n, theta)
```

Arguments

n	size of sample
theta	values of two parameters of BB9 copula; first must be positive; second must be greater than one.

Details

see page 224 of QRM. Algorithm essentially generates V corresponding to Joe's BB9 copula. For this copula algorithm uses fairly naive rejection and is SLOW!

Value

random sample of size n

References

Joe, H. Multivariate Models and Dependence Concepts, Chapman and Hall, 1997. See page 154 for BB9 copula.

See Also

[rAC](#)

rbinomial.mixture *Sample Mixed Binomial Distribution*

Description

random generation from mixed binomial distribution

Usage

```
rbinomial.mixture(n=1000, m=100, model="probitnorm", ...)
```

Arguments

n	sample size
m	vector of numbers of coin flips
model	name of mixing distribution: "probitnorm", "logitnorm", "beta", "
...	further parameters of mixing distribution

Details

see pages 354-355 and pages 375-377 of QRM

Value

vector of numbers of successes

See Also

[rbeta](#), [rprobitnorm](#), [rlogitnorm](#)

Examples

```
## Not run:
pi <- 0.04896; #one obligor defaulting pi = .04896
pi2 <- 0.00321; #two obligors defaulting jointly pi2 = .0031265
beta.pars <- cal.beta(pi,pi2);
probitnorm.pars <- cal.probitnorm(pi,pi2);
n <- 1000;
m <- rep(500,n);
M.beta <- rbinomial.mixture(n,m,"beta",shape1=beta.pars[1],
                           shape2=beta.pars[2]);
M.probitnorm <- rbinomial.mixture(n,m,"probitnorm",
                                 mu=probitnorm.pars[1],sigma=probitnorm.pars[2]);
## End(Not run)
```

rcopula.AGumbel	<i>Generate Asymmetric Gumbel Copula</i>
-----------------	--

Description

generates sample from asymmetric Gumbel copula

Usage

```
rcopula.AGumbel(n, theta, alpha=rep(1, d), d=2)
```

Arguments

n	sample size
theta	scalar parameter of an exchangeable Gumbel copula
alpha	vector of length d containing parameters of asymmetry
d	dimension of copula

Details

see pages 224-226 of QRM for bivariate example of this copula; the idea obviously carries over to higher dimensions

Value

a matrix of dimension n times d where rows are realizations

See Also

[rAC](#), [rcopula.Gumbel2Gp](#), [rcopula.GumbelNested](#)

Examples

```
data.AGumbel <- rcopula.AGumbel(10000, theta=4,
                                alpha=c(0.95, 0.7));
plot(data.AGumbel);
data.AGumbel <- rcopula.AGumbel(5000, theta=2,
                                c(0.1, 0.7, 0.8, 0.9));
pairs(data.AGumbel);
hist(data.AGumbel[, 4]);
```

rcopula.clayton	<i>Clayton Copula Simulation</i>
-----------------	----------------------------------

Description

generates a random sample from the Clayton copula

Usage

```
rcopula.clayton(n, theta, d)
```

Arguments

n	sample size
theta	parameter value
d	dimension of copula

Details

see pages 222-224 in QRM

Value

matrix with n rows and d columns where rows are realizations

See Also

[rAC](#), [rcopula.gumbel](#), [rcopula.gauss](#), [rcopula.t](#), [rcopula.frank](#)

Examples

```
data <- rcopula.clayton(1000, 2, 4);
pairs(data);
```

rcopula.frank	<i>Frank Copula Simulation</i>
---------------	--------------------------------

Description

generates a random sample from the Frank copula

Usage

```
rcopula.frank(n, theta, d)
```

Arguments

n	sample size
theta	parameter value
d	dimension of copula

Details

see pages 222-224 in QRM

Value

matrix with n rows and d columns where rows are realizations

See Also

[rAC](#), [rcopula.gumbel](#), [rcopula.clayton](#), [rcopula.gauss](#), [rcopula.t](#)

Examples

```
## Not run:  
data <- rcopula.frank(1000, 1.5, 4);  
pairs(data);  
## End(Not run)
```

rcopula.gauss	<i>Gauss Copula Simulation</i>
---------------	--------------------------------

Description

generates a random sample from the Gaussian copula

Usage

```
rcopula.gauss(n, Sigma=equicorr(d, rho), d=2, rho=0.7)
```

Arguments

n	number of observations
Sigma	correlation matrix
d	dimension of copula
rho	correlation parameter for specifying an equicorrelation structure

Details

This function is set up to allow quick simulation of Gauss copulas with an equicorrelation structure. Simply enter a value for the dimension d and the correlation parameter rho. For more general correlation matrices specify Sigma.

Value

a matrix with n rows and d columns

See Also

[rAC](#), [rcopula.gumbel](#), [rcopula.clayton](#), [rcopula.frank](#), [rcopula.t](#)

Examples

```
data <- rcopula.gauss(2000,d=6,rho=0.7);
pairs(data);
```

rcopula.gumbel	<i>Gumbel Copula Simulation</i>
----------------	---------------------------------

Description

generates a random sample from the Gumbel copula

Usage

```
rcopula.gumbel(n, theta, d)
```

Arguments

n	sample size
theta	parameter value
d	dimension of copula

Details

see pages 222-224 in QRM

Value

matrix with n rows and d columns where rows are realizations

```
rcopula.GumbelNested
```

Gumbel Copula with Nested Structure

Description

generates sample from a d dimensional Gumbel copula with nested structure constructed using (d-1) Gumbel generators

Usage

```
rcopula.GumbelNested(n, theta)
```

Arguments

n	sample size
theta	vector of admissible Gumbel copula parameters of length (d-1) ordered by increasing size

Details

see page 226 of QRM for trivial thrivariate example

Value

matrix of dimension n by d where rows are realizations

See Also

[rAC](#), [rcopula.AGumbel](#), [rcopula.Gumbel2Gp](#)

Examples

```
data <- rcopula.GumbelNested(n=3000,theta=1:6);
pairs(data);
```

```
rcopula.t
```

t Copula Simulation

Description

generates a random sample from the t copula

Usage

```
rcopula.t(n, df, Sigma=equicorr(d, rho), d=2, rho=0.7)
```

Arguments

n	number of observations
df	degrees of freedom
Sigma	correlation matrix
d	dimension of copula
rho	correlation parameter for specifying an equicorrelation structure

Details

This function is set up to allow quick simulation of t copulas with an equicorrelation structure. Simply enter a value for the dimension d and the correlation parameter rho. For more general correlation matrices specify Sigma.

Value

a matrix with n rows and d columns

See Also

[rAC](#), [rcopula.gumbel](#), [rcopula.clayton](#), [rcopula.gauss](#), [rcopula.frank](#)

Examples

```
data <- rcopula.t(2000,df=4,d=6,rho=0.7);
pairs(data);
```

rFrankMix

Mixture Distribution Yielding Frank Copula

Description

generates random sample from discrete mixing distribution required for sampling from Frank's copula using Laplace transform method

Usage

```
rFrankMix(n, theta)
```

Arguments

n	size of sample
theta	value of parameter of Frank copula

Details

see page 224 of QRM. Algorithm generates V corresponding to Frank's copula.

Value

random sample of size n

See Also[rAC](#)**Examples**

```
## Not run:
#Pass the parameter values n=20 and theta=0.5
result <- rFrankMix(20,0.5);
## End(Not run)
```

rGIG

Generate Random Vector from Generalized Inverse Gaussian Distribution

Description

random generation for the generalized inverse Gaussian distribution

Usage

```
rGIG(n, lambda, chi, psi, envplot=FALSE, messages=FALSE)
```

Arguments

n	sample size
lambda	scalar parameter
chi	scalar parameter
psi	scalar parameter
envplot	whether a plot of the rejection envelope should be made; default is FALSE
messages	whether a message about rejection rate should be returned; default is FALSE

Details

uses a rejection algorithm suggested by Atkinson (1982)

Value

random sample of size n

References

Atkinson A.C. (1982). The simulation of generalized inverse Gaussian and hyperbolic random variables. SIAM Journal on Scientific Computing 3(4): 502-515.

See Also[rghyp](#), [rmghyp](#)

Examples

```
## Not run:
#Create a mean-variance normal mixture of random
#variables called the generalized hyperbolic
#It is not necessarily elliptical but its univariate
#version will be. See p. 78 in QRM.
# This is the GH model.
rghyp <- function(n, lambda, chi, psi, mu=0, gamma=0)
{
  #generate a series of random Generalized Inverse Gaussian
  #variables: see p. 77 of QRM text
  W <- rGIG(n, lambda, chi, psi);
  # Generate a similar random sequence of standard normals:
  Z <- rnorm(n);
  #Mix the two distributions using equation 3.25 (p. 77) but
  #with gamma possibly 0 or a scalar
  sqrt(W) * Z + mu + gamma * W;
}
## End(Not run)
```

RiskMeasures

*Calculate Risk Measures from GPD Fit***Description**

calculates risk measures like VaR and expected shortfall based on a generalized Pareto model fitted to losses over a high threshold

Usage

```
RiskMeasures(out, p)
```

Arguments

out	results of a GPD fit to excesses over high thresholds
p	vector of probability levels for risk measures

Details

see pages 282-284 of QRM

Value

matrix with quantile and shortfall estimates for each probability level

See Also

[fit.GPD](#), [showRM](#)

Examples

```
data(danish);
out <- fit.GPD(danish,10);
RiskMeasures(out,c(0.99,0.999));
```

`rlogitnorm`*Random Number Generation from Logit-Normal Distribution*

Description

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 μ and σ

Usage

```
rlogitnorm(n, mu, sigma)
```

Arguments

<code>n</code>	sample size
<code>mu</code>	scalar parameter
<code>sigma</code>	scalar parameter

Details

see pages 353-354 in QRM

Value

random sample of size n

Author(s)

documentation by Scott Ulman for R-language distribution

See Also

[rbeta](#), [rclaytonmix](#), [rprobitnorm](#)

Examples

```
#set number, mean, and variance:
num <- 1000;
mu <- 2.0;
sigma <- 1.25;
#Simulate values from logistic norm mix
simVals <- rlogitnorm(num,mu,sigma);
```

rmnorm*Multivariate Normal Random Sample*

Description

generates random sample from multivariate normal

Usage

```
rmnorm(n, Sigma=equicorr(d, rho), mu=rep(0, d), d=2, rho=0.7)
```

Arguments

n	number of realizations
Sigma	a covariance matrix
mu	a mean vector
d	dimension of distribution
rho	correlation value to build equicorrelation matrix

Details

function is set up to quickly simulate equicorrelation structures by specifying d and rho

Value

an n by d matrix

Author(s)

documentation by Scott Ulman for R-language distribution

See Also

[rmt](#), [equicorr](#)

Examples

```
ndata <- rmnorm(1000, rho=0.7, d=3);  
## Not run:  
Sigma <- diag(c(3,4,5)) %*% equicorr(3,0.6) %*% diag(c(3,4,5));  
mu <- c(1,2,3);  
ndata <- rmnorm(1000, Sigma, mu);  
fit.norm(ndata);  
## End(Not run)
```

rmt

*Multivariate t***Description**

generates random sample from multivariate t

Usage

```
rmt(n, df=4, Sigma=equicorr(d, rho), mu=rep(0, d), d=2, rho=0.7)
```

Arguments

n	number of realizations
df	degrees of freedom
Sigma	a dispersion matrix
mu	a mean vector
d	dimension of distribution
rho	correlation value to build equicorrelation matrix

Details

function is set up to quickly simulate equicorrelation structures by specifying d and rho

Value

an n by d matrix

See Also

[rmnorm](#), [equicorr](#), [rmghyp](#)

Examples

```
## Not run:
ndata <- rmnorm(1000, df=4, rho=0.7, d=3);
Sigma <- diag(c(3, 4, 5)) %*% equicorr(3, 0.6)
      %*% diag(c(3, 4, 5));
mu <- c(1, 2, 3);
tdata <- rmt(1000, 4, Sigma, mu);
modl <- fit.mst(tdata);
## End(Not run)
```

rstable

*Stable Distribution***Description**

random sample from stable distribution

Usage

```
rstable(n, alpha, beta=1)
```

Arguments

n	sample size
alpha	scalar parameter strictly larger than 0 and smaller than 2 (but avoid alpha=1)
beta	scalar parameter between -1 and 1

Details

see pages 224 and 498 of QRM; default value beta=1 combined with an alpha value less than 1 gives positive stable distribution which we require for Gumbel copula generation; the case alpha=1 has not been implemented

Value

sample of size n

References

Forthcoming John Nolan Book; see Definition 1.8 and Theorem 1.19

See Also

[rcopula.gumbel](#)

Examples

```
## Not run:
#Use rstable() method in copula simulation function
rcopula.Gumbel2Gp <- function(n = 1000,
                             gpsizes = c(2,2), theta = c(2,3,5))
{
  Y <- rstable(n, 1/theta[1]) * (cos(pi/(2*theta[1])))^theta[1];
  innerU1 <- rcopula.gumbel(n, theta[2]/theta[1], gpsizes[1]);
  innerU2 <- rcopula.gumbel(n, theta[3]/theta[1], gpsizes[2]);
  U <- cbind(innerU1, innerU2);
  Y <- matrix(Y, nrow = n, ncol = sum(gpsizes));
  out <- exp( - ( - log(U)/Y)^(1/theta[1]));
  return(out);
}
## End(Not run)
```

rtcopulamix	<i>Mixing Distribution on Unit Interval Yielding t Copula Model</i>
-------------	---

Description

random generation for mixing distribution on unit interval yielding t copula model

Usage

```
rtcopulamix(n, pi, rho.asset, nu)
```

Arguments

n	sample size
pi	default probability
rho.asset	asset correlation parameter
nu	degree of freedom parameter

Details

see page 361 in QRM; we consider exchangeable case of this model

Value

random values on unit interval

See Also

[rbeta](#), [rclaytonmix](#), [rlogitnorm](#), [rprobitnorm](#)

Examples

```
#set number, mean, and variance:
num <- 1000;
pi <- 0.9;
rho = 0.5;
df <- 4;
simVals <- rtcopulamix(num,pi,rho,df);
```

seMPP.negloglik *Marked Self-Exciting Point Process Log-Likelihood*

Description

evaluates negative log-likelihood of a marked self-exciting point process model; this will be objective function massed to `nlminb()` or `optim()`.

Usage

```
seMPP.negloglik(theta, PP, case, markdens)
```

Arguments

theta	vector of parameters of self-exciting model
PP	point-process data
case	a numerical variable coding whether Hawkes or ETAS forms are used and whether marks may have an influence on future points
markdens	name of density for marks; currently must be "GPD"

Value

value of log-likelihood

Author(s)

documentation by Scott Ulman for R-language distribution

See Also

[fit.seMPP](#), [fit.sePP](#)

Examples

```
## Not run:
#Example of using seMPP.negloglik as objective function passed
#to optimizer function
fit.seMPP <- function(PP,markdens="GPD",model="Hawkes",
  mark.influence=TRUE,predictable=FALSE,std.errs=FALSE)
{
  if (class(PP) != "MPP") stop("Not marked point process data");
  marks <- PP$marks;
  groundmod <- fit.sePP(PP,model,mark.influence=TRUE,std.errs=FALSE);
  #lines removed here...
  if(predictable)
  {
    theta <- c(par.ests,0);
    fit <- nlminb(start=theta, objective=seMPP.negloglik,
      PP=PP,case=case, markdens=markdens);
    #Lines removed here ...
  }
}
## End(Not run)
```

sePP.negloglik

*Self-Exciting Point Process Log-Likelihood***Description**

evaluates negative log-likelihood of a self-exciting point process model (unmarked)

Usage

```
sePP.negloglik(theta, PP, case)
```

Arguments

theta	parameters of self-exciting model
PP	point-process data
case	a numerical variable coding whether Hawkes or ETAS forms are used and whether marks may have an influence on future points

Value

value of log-likelihood

Author(s)

documentation by Scott Ulman for R-language distribution

See Also

[fit.sePP](#), [fit.seMPP](#)

Examples

```
## Not run:
#Example of using sePP.negloglik as objective function passed
#to optimizer function
fit.sePP <- function(PP,markdens="GPD",model="Hawkes",
  mark.influence=T,predictable=F,std.errs=F)
{
  #lines removed here...
  fit <- nlminb(start=theta, objective=sePP.negloglik, PP=PP,case=case);
  par.ests <- fit$par;
  par.ests <- abs(par.ests)
  ll.max <- -sePP.negloglik(par.ests,PP,case)
  #Lines removed here ...
}
## End(Not run)
```


showRM

*Show Risk Measure Estimates on Tailplot***Description**

shows estimates of risk measures (like VaR and ES) on a tailplot

Usage

```
showRM(object, alpha, RM="VaR", extend=2, ci.p=0.95, like.num=50.)
```

Arguments

object	results of fit.GPD
alpha	probability level
RM	risk measure, VaR or ES
extend	how far to extend picture; x-axis extends to this value times the largest observation
ci.p	confidence level for confidence interval
like.num	number of evaluations of profile likelihood

Details

see pages 282-284 in QRM

Value

point estimate and confidence interval for risk measure

Side Effects

plotTail is called

See Also

[plotTail](#), [fit.GPD](#), [RiskMeasures](#)

Examples

```
## Not run:
data(danish);
#Fit the GPD using MLE a
mod <- fit.GPD(danish,10);
showRM(mod,0.99, RM="VaR");
showRM(mod,0.99, RM="ES");
showRM(mod,0.995, RM="VaR");
showRM(mod,0.995, RM="ES");
## End(Not run)
```

signalSeries	<i>signalSeries object</i>
--------------	----------------------------

Description

Structured after the S-Plus signalSeries object. It contains a data slot of any type and a NUMERIC positions slot rather than the date slot of a timeSeries. In other words, each data value has a numeric value associated with its position in the overall list

Usage

```
signalSeries(data, positions., units, units.position, from = 1, by = 1)
```

Arguments

data	a component which is typically a dataframe
positions.	a numeric component describing the positions of the data values
units	character vector describing the type of units used in the data structure
units.position	character vector describing the type of units used for the positions
from	starting value of positions
by	amount to skip between positions

Details

If no arguments are supplied, the default (empty) signalSeries object is returned. Otherwise, a signalSeries object is created with the given positions and data, and units if they are supplied. As an alternative to supplying the positions directly, they can be supplied by giving from and by, in which case the positions are generated as a numeric sequence with the right length to match the data

Value

a signalSeries object with the given data and positions

See Also

[aggregateSignalSeries](#)

Examples

```
signalSeries(); #default object with no data or positions
#Create matrix of simulated values from multivariate-t distribution
m <- 90; n <- 3000;
dataSim <- rmt(m*n,df=3,rho=0.5,d=2);
dataSimSS <- signalSeries(dataSim);
```

smi	<i>Swiss Market Index</i>
-----	---------------------------

Description

This timeSeries data set provides the daily closing values of the SMI (Swiss Market Index) from November 1990 to March 2004

sp500	<i>Standard and Poors 500 Index</i>
-------	-------------------------------------

Description

This timeSeries data set provides the daily closing values of the SandP 500 Index from 1990 to March 2004

spdata.raw	<i>Standard and Poors Default Data</i>
------------	--

Description

The spdata.raw data frame has 20 rows and 11 columns. It contains default data for A, BBB, BB, B and C-rated companies for the years 1981 to 2000

Usage

```
data(spdata.raw)
```

Format

This data frame contains the following 11 columns:

<i>year</i>	year of default
<i>Aobligors</i>	number of A-rated companies
<i>Adefaults</i>	number of A-rated companies defaulting in year
<i>BBBobligors</i>	number of BBB-rated companies
<i>BBBdefaults</i>	number of BBB-rated companies that default in year
<i>BBobligors</i>	number of BB-rated companies
<i>BBdefaults</i>	number of BB-rated companies that default in year
<i>Bobligors</i>	number of B-rated companies
<i>Bdefaults</i>	number of B-rated companies that default in year
<i>CCCObligors</i>	number of CCC-rated companies
<i>CCCdefaults</i>	number of CCC-rated companies that default in year

There are 20 rows with values for the years from 1981 to 2000

Source

Standard & Poors Credit Monitor

See Also

[spdata](#), [momest](#)

Examples

```
data(spdata.raw);
attach(spdata.raw);
BdefaultRate <- Bdefaults/Bobligors;
BBdefaultRate <- BBdefaults/BBobligors;
BBBdefaultRate <- BBBdefaults/BBBobligors;
AdefaultRate <- Adefaults/Aobligors;
CCCdefaultRate <- CCCdefaults/CCCobligors;
## Not run:
plot(year, CCCdefaultRate, xlab="Year", ylab="Rate", type="l");
lines(year, BdefaultRate, col=2);
lines(year, BBdefaultRate, col=3);
lines(year, BBBdefaultRate, col=4);
lines(year, AdefaultRate, col=5);
momest(Bdefaults, Bobligors);
pi.B <- momest(Bdefaults, Bobligors)[1];
pi2.B <- momest(Bdefaults, Bobligors)[2];
rhoY.B <- (pi2.B-pi.B^2)/(pi.B-pi.B^2);
## End(Not run)
```

spdata	<i>Standard and Poors Default Data</i>
--------	--

Description

The spdata data frame has 100 rows and 4 columns. It contains default data for A, BBB, BB, B and C-rated companies for the years 1981 to 2000

Usage

```
data(spdata)
```

Format

a matrix containing 100 rows and 4 columns.
The columns are:

<i>year</i>	year of default,
<i>rating</i>	rating category (A, BBB, BB, B, CCC),
<i>firms</i>	number of companies in rating category,
<i>number of defaults</i>	number of companies defaulting in category

The rows are the years from 1981-2000

Author(s)

documentation by Scott Ulman for R-language distribution

Source

Standard and Poors Credit Monitor

See Also

[spdata.raw](#), [momest](#)

Examples

```
## Not run:
library(MASS);
library(nlme);
#the data being used is spdata which has 100 rows and 4 columns:
#'year', 'rating', 'firms', defaults'
#Use R- library MASS to get glmmPQL
#'year' -'ratings' determine the unique results(20 years 1981-2000 with 5 obligor
#class ratings each year
results <- glmmPQL(cbind(defaults,firms-defaults) ~ -1 + rating,
  random = ~1| year, family=binomial(probit), data=spdata);
summary(results);
summary(results)$tTable[,1];
detach("package:nlme");
detach("package:MASS");
## End(Not run)
```

Spearman

Spearman's Rank Correlation

Description

calculates a matrix of Sperman's rank correlations

Usage

```
Spearman(data)
```

Arguments

data data matrix

Details

see pages 229-230 in QRM

Value

matrix of rank correlations

See Also

[Kendall](#)

Examples

```
data <- rmnorm(1000,d=3,rho=0.5);
Spearman(data);
```

stationary.sePP	<i>Stationarity of Self-Exciting Model</i>
-----------------	--

Description

checks a sufficient condition for stationarity of a self-exciting model and gives information about cluster size

Usage

```
stationary.sePP(sePP)
```

Arguments

sePP	a fitted self-exciting process created with fit.sePP or marked self-exciting process created with fit.seMPP
------	---

Value

a vector consisting of binary flag for stationarity condition, estimated number of direct decendents of any event and estimated size of cluster generated by any new event

References

Daley and Vere-Jones, An Introduction to the Theory of Point Processes, Springer, 2nd Edition 2003, page 203

See Also

[fit.sePP](#), [fit.seMPP](#)

Examples

```
data(sp500);
sp500.nreturns <- -mk.returns(sp500);
window <- (seriesPositions(sp500.nreturns) >
  timeDate("12/31/1995",format = "%m/%d/%Y"));
sp500.nreturns <- sp500.nreturns>window;
tmp <- extremalPP(sp500.nreturns,ne=100);
mod3a <- fit.seMPP(tmp,mark.influence=FALSE,std.errs=TRUE);
#Note that stationary.sePP applies to both sePP and seMPP processes.
stationary.sePP(mod3a);
```

symmetrize	<i>Ensure Symmetric Matrix</i>
------------	--------------------------------

Description

ensures a matrix that should be symmetric is really symmetric

Usage

```
symmetrize(matrix)
```

Arguments

`matrix` a matrix that should be symmetric

Details

deals with situations where rounding errors cause symmetric matrices to appear asymmetric

Value

a matrix that is symmetric

Examples

```
## Not run:
#lines of code taken from fit.mst() in functionsNormix.R
# ...
Sigma <- var(data);
Sigma <- symmetrize(Sigma);
beta <- as.vector(solve(Sigma))
mean <- as.numeric(mu+EW*gamma);
covariance <- EW*Sigma + varW*outer(gamma, gamma);
## End(Not run)
```

timeSeriesClass	<i>timeSeries Objects in R</i>
-----------------	--------------------------------

Description

The R-language has developed multiple 'time-series' type objects through multiple contributors. The time-series objects in R-language closest to those in S-Plus appear to be those belonging to the timeSeries class described in fCalendar library from R-metrics. Dates and times are implemented as 'timeDate' objects within 'timeSeries'. The class contains functions for the generation and representation of 'timeSeries' objects and mathematical operations on the objects.

Use timeSeries() as the constructor for the class.

Usage:

```
timeSeries(data, charvec, units=NULL, format="ISO", zone="GMT",
FinCenter=myFinCenter,
recordIDs=data.frame(),title=NULL, documentaion = NULL, ...)
```

Arguments

<code>data</code>	a vector or matrix or data frame containing numeric values
<code>charvec</code>	a character vector of dates and times
<code>units</code>	(optional) character string allowing overwrite of current column names
<code>format</code>	(optional) timeDate string format, defaulting to 'ISO'
<code>zone</code>	(optional) time zone where the data were recorded
<code>FinCenter</code>	location of financial center as Continent/City

Details

IMPORTANT INFORMATION: You can extract the DATE segment from a timeSeries object using

- 1) the seriesPosition function: (e.g. use seriesPositions(sp500))
- 2) the positions attribute (e.g. dates <- sp500@positions)

You can extract the NUMERIC segment from a timeSeries object using

- 1) the seriesData function (e.g. use seriesData(sp500))
- 2) the Data attribute (e.g. returns <- sp500@Data)

Value

a timeSeries object

Author(s)

documentation by Scott Ulman for R-language distribution

See Also

[TimeSeriesClassRMetrics](#)

TimeSeriesClassRMetrics

timeSeries Class and Methods

Description

A collection and description of functions and methods dealing with regular and irregular 'timeSeries' objects. Dates and times are implemented as 'timeDate' objects. Included are functions and methods for the generation and representation of 'timeSeries' objects, and for mathematical operations.

The functions and methods for the Generation of 'timeSeries' Objects are:

<code>'timeSeries()'</code>	Creates a 'timeSeries' object from scratch,
<code>'read.timeSeries()'</code>	Reads a 'timeSeries' from a spreadsheet file,
<code>'as.timeSeries()'</code>	S3: Creates 'time Series' from a 'matrix',
<code>'is.timeSeries()'</code>	S3: Tests if an object is of class a 'timeSeries',
<code>'print.timeSeries()'</code>	S3: Print method for a 'timeSeries' object,
<code>'plot.timeSeries()'</code>	S3: Plot method for a 'timeSeries' object,
<code>'lines.timeSeries()'</code>	S3: Lines method for a 'timeSeries' object,
<code>'Ops.timeSeries()'</code>	S3: Arith method for a 'timeSeries' object,
<code>'[.timeSeries()'</code>	S3: "[" method for a 'timeSeries' object,
<code>'head.timeSeries()'</code>	S3: returns the head of a 'timeSeries' object,
<code>'tail.timeSeries()'</code>	S3: returns the tail of a 'timeSeries' object.

The functions and methods for the Representation of 'timeSeries' Objects are:

'seriesData()'	Extracts data slot from a 'timeSeries',
'seriesPositions()'	Extracts positions slot from a 'timeSeries',
'start.timeSeries()'	S3: Extracts start date of a 'timeSeries',
'end.timeSeries()'	S3: Extracts end date of a 'timeSeries',
'as.vector.timeSeries()'	S3: Converts a 'timeSeries' to a vector,
'as.matrix.timeSeries()'	S3: Converts a 'timeSeries' to a matrix,
'as.data.frame.timeSeries()'	S3: Converts a 'timeSeries' to a data.frame.

The functions and methods for Math Operations of 'timeSeries' Objects are:

'applySeries()'	Applies a function to margins of a 'timeSeries',
'alignDailySeries()'	Aligns a daily 'timeSeries' to new positions,
'cut()'	Cuts out a piece from a 'timeSeries' object,
'mergeSeries()'	Merges a 'timeSeries' object with a 'matrix',
'ohlcDailyPlot()'	Plots open high low close bar chart,
'revSeries()'	Reverts the order of 'timeSeries' object,
'diffSeries()'	Takes differences from a 'timeSeries' object,
'lagSeries()'	Lags a 'timeSeries' object,
'outlierSeries()'	Removes outliers from a 'timeSeries' object,
'returnSeries()'	Computes returns from a 'timeSeries' object,
'logSeries()'	Returns logarithms of a 'timeSeries' object,
'absSeries()'	Returns absolute values of a 'timeSeries' object.

Functions calls include:

```
timeSeries(data, charvec, units = NULL, format = "ISO", zone = "GMT", FinCenter = myFinCenter, recordIDs = data.frame(), title = NULL, documentation = NULL, ...)
```

```
read.timeSeries(file, zone = "GMT", FinCenter = "", title = "", documentation = "", sep = ";")
```

```
as.timeSeries(x, dimnames = TRUE, format = "")
```

```
is.timeSeries(object)
```

The following are S3 method for class 'timeSeries':

```
print(x, ...)
```

```
plot(x, reference.grid = TRUE, lty = 1, ...)
```

```
lines(x, ...)
```

```
Ops(e1, e2)
```

```
x[i = min(1, nrow(x@Data)):nrow(x@Data),
```

```
j = min(1, ncol(x@Data)):ncol(x@Data))
```

```
head(x, ...)
```

```
tail(x, ...)
```

```
seriesData(object)
```

```
seriesPositions(object)
```

The following are S3 method for class 'timeSeries':

```
start(x, ...)
```

```
end(x, ...)
```

```
as.vector(x, mode = "any")
```

```
as.matrix(x)
```

```
as.data.frame(x, row.names = NULL, optional = NULL)
```

```

applySeries(x, from = NULL, to = NULL, by=c("monthly","quarterly"), FUN = colAves, units =
NULL, ...);
cut(x, from, to);
diffSeries(x, lag = 1, diff = 1, trim = FALSE, pad = NA);
lagSeries(x, k = 1, trim = FALSE, units = NULL);
outlierSeries(x, sd = 10, complement = TRUE);
mergeSeries(x, y, units = NULL);
returnSeries(x, type = c("continuous", "discrete"), percentage = FALSE, trim = TRUE, digits = 4,
units = NULL);
revSeries(x);
logSeries(x);
absSeries(x);
alignDailySeries(x, method = c("before", "after", "interp", "fillNA"), include.weekends = FALSE,
units = NULL);
ohlcdailyPlot(x, volume = TRUE, colOrder = c(1:5), units = 1e6, xlab = c("Date", "Date"), ylab =
c("Price", "Volume"), main = c("O-H-L-C", "Volume"), grid.nx = 7, grid.lty = "solid", ...);

```

Details

Generation of Time Series Objects:

We have defined a `timeSeries` class which is in many aspects similar to the S-Plus class with the same name, but has also some important differences. The class has seven Slots, the ‘Data’ slot which holds the time series data in matrix form, the ‘position’ slot which holds the time/date as a character vector, the ‘format’ and ‘FinCenter’ slots which are the same as for the ‘timeDate’ object, the ‘units’ slot which holds the column names of the data matrix, and a ‘title’ and a ‘documentation’ slot which hold descriptive character strings. Date and time is managed in the same way as for `timeDate` objects.

Author(s)

documentation by Scott Ulman for R-language distribution

See Also

[timeSeriesClass](#)

unmark

Unmark Point Process

Description

strips marks from a marked point process

Usage

```
unmark (PP)
```

Arguments

PP a point process object of class PP

Details

If necessary, more details than the description above

Value

Describe the value returned If it is a LIST, use

comp1	Description of 'comp1'
comp2	Description of 'comp2'
...	

See Also

[fit.sePP](#), [fit.seMPP](#), [extremalPP](#)

Examples

```
data(sp500);
sp500.nreturns <- -mk.returns(sp500);
window <- (seriesPositions(sp500.nreturns) >
  timeDate("12/31/1995", format = "%m/%d/%Y"));
sp500.nreturns <- sp500.nreturns>window;
tmp <- extremalPP(sp500.nreturns, ne=100);
tmp2 <- unmark(tmp);
```

volfunction

Self-Excitement Function

Description

calculates a self-excitement function for use in the negloglik() methods used in fit.sePP() and fit.seMPP()

Usage

```
volfunction(anytimes, times, marks, theta, model)
```

Arguments

anytimes	vector of times at which to calculate self-excitement function
times	times of point events
marks	marks associated with point events
theta	parameters of self-excitement function
model	model number

Details

see page 306 of QRM

Value

a vector of same length as "anytimes"

See Also

[fit.sePP](#), [fit.seMPP](#)

Examples

```
## Not run:
seMPP.negloglik <- function(theta, PP, case, markdens)
{
  theta <- abs(theta);
  times <- PP$times;
  marks <- PP$marks;
  endtime <- PP$endtime;
  starttime <- PP$starttime;
  mu <- theta[1];
  phi <- theta[2];
  voltheta <- theta[-c(1,2,(length(theta)-2),
    (length(theta)-1),length(theta))];
  evol <- volfunction(times,times,marks,voltheta,case);
  lambda.contrib <- mu+phi*evol
  # ... remaining lines omitted here
}
## End(Not run)
```

xdax	<i>Xetra DAX Index</i>
------	------------------------

Description

This timeSeries data set provides the daily closing values of the Xetra DAX index from 1994 to March 2004

xiplot	<i>GPD Shape Parameter Plot</i>
--------	---------------------------------

Description

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

Usage

```
xiplot(data, models=30., start=15., end=500., reverse=TRUE,
ci=0.95, auto.scale=TRUE, labels=TRUE, table=FALSE, ...)
```

Arguments

<code>data</code>	vector or time series of data
<code>models</code>	number of consecutive gpd models to be fitted; i.e. the number of different thresholds at which to re-estimate ξ ; this many ξ estimates will be plotted
<code>start</code>	lowest number of exceedances to be considered
<code>end</code>	maximum number of exceedances to be considered
<code>reverse</code>	should plot be by increasing threshold (TRUE) or number of extremes (FALSE)
<code>ci</code>	probability for asymptotic confidence band; for no confidence band set to FALSE
<code>auto.scale</code>	whether or not plot should be automatically scaled; if not, <code>xlim</code> and <code>ylim</code> graphical parameters may be entered
<code>labels</code>	whether or not axes should be labelled; default is TRUE
<code>table</code>	should a table of results be printed; default is FALSE
<code>...</code>	further parameters of <code>xipLOT</code> function

Details

For every model "fit.GPD" is called. Evaluation may be slow.

See Also

[fit.GPD](#), [MEplot](#)

Examples

```
# Shape plot of heavy-tailed Danish fire insurance data:
data(danish);
xipLOT(danish);
```

Index

*Topic **array**

- CovToCor, 19
- eigenmeth, 28
- equicorr, 30
- Pconstruct, 74
- Pdeconstruct, 75
- symmetrize, 109

*Topic **classes**

- signalSeries, 104
- TimeSeriesClassRMetrics, 110

*Topic **datasets**

- cac40, 14
- Danish, 19
- DJComponents, 23
- dji, 23
- ftse100, 53
- FXGBP.RAW, 53
- hsi, 60
- nasdaq, 73
- nikkei, 74
- smi, 105
- sp500, 105
- spdata, 106
- spdata.raw, 105
- xdax, 114

*Topic **distribution**

- beta (stats), 12
- claytonmix, 18
- dcopula.clayton, 20
- dcopula.gauss, 21
- dcopula.gumbel, 21
- dcopula.t, 22
- dmnorm, 24
- dmt, 24
- dsmghyp, 25
- EGIG, 27
- ElogGIG, 29
- ESnorm, 31
- ESst, 32
- GEV, 53
- ghyp, 54
- ghypB, 55
- GPD, 57

- Gumbel, 58
- mghyp, 67
- probitnorm, 81
- qst, 84
- rAC, 85
- rBB9Mix, 85
- rbinomial.mixture, 86
- rcopula.AGumbel, 87
- rcopula.clayton, 88
- rcopula.frank, 89
- rcopula.gauss, 89
- rcopula.gumbel, 90
- rcopula.Gumbel2Gp, 91
- rcopula.GumbelNested, 92
- rcopula.t, 92
- rFrankMix, 93
- rGIG, 94
- rlogitnorm, 96
- rmnorm, 97
- rmt, 98
- rstable, 99
- rtcopulamix, 100

*Topic **hplot**

- BiDensPlot, 14
- hillPlot, 59
- MEplot, 66
- plot.MPP, 75
- plot.PP, 76
- plot.sePP, 77
- plot.timeSeries, 78
- plot.timeSeriesIts, 79
- plotFittedGPDvsEmpiricalExcesses, 80
- plotTail, 81
- QQplot, 83
- showRM, 103
- xiplot, 114

*Topic **htest**

- momest, 72

*Topic **math**

- besselM3, 11
- lbeta, 63
- psifunc, 83

***Topic methods**

aggregateSignalSeries, 8
 findthreshold, 34
 fit.Archcopula2d, 34
 fit.binomial, 35
 fit.binomialBeta, 36
 fit.binomialLogitnorm, 37
 fit.binomialProbitnorm, 38
 fit.gausscopula, 39
 fit.GEV, 40
 fit.GPD, 41
 fit.GPDb, 42
 fit.mNH, 43
 fit.mst, 44
 fit.NH, 45
 fit.norm, 46
 fit.POT, 47
 fit.seMPP, 48
 fit.sePP, 49
 fit.st, 50
 fit.tcopula, 52
 fit.tcopula.rank, 51
 Kendall, 62
 kurtosisSPlus, 62
 mk.its.exceedances.tS, 68
 mk.its.exceedances.vector, 70
 mk.returns, 72
 unmark, 112
 volfunction, 113

***Topic misc**

RiskMeasures, 95

***Topic models**

aggregateMonthlySeries, 6
 aggregateWeeklySeries, 10
 beta (stats), 12
 cal.beta, 15
 cal.claytonmix, 16
 cal.probitnorm, 17
 extremalPP, 33
 stationary.sePP, 108

***Topic multivariate**

EMupdate, 29
 jointnormalTest, 61
 Kendall, 62
 MardiaTest, 64
 MCECM.Qfunc, 64
 MCECMupdate, 65
 Spearman, 107

***Topic optimize**

seMPP.negloglik, 101
 sePP.negloglik, 102

***Topic package**

profileLoadLibrary, 4
 QRMBBook-workspace, 2
 QRMLib-package, 1
 storeDataInWorkspace, 3

***Topic ts**

aggregateMonthlySeries, 6
 aggregateQuarterlySeries, 7
 aggregateWeeklySeries, 10
 mk.its.exceedances.tS, 68
 mk.its.exceedances.vector, 70
 signalSeries, 104
 timeSeriesClass, 109
 TimeSeriesClassRMetrics, 110

***Topic utilities**

edf, 26
 hessb, 58
 mk.oldts, 71

aggregateMonthlySeries, 6, 8, 11
 aggregateQuarterlySeries, 7, 7, 11
 aggregateSignalSeries, 8, 104
 aggregateWeeklySeries, 7, 8, 10

besselm3, 11, 55, 56, 83
 beta, 13
 beta (stats), 12
 BetaDist (beta (stats)), 12
 BiDensPlot, 14

cac40, 14
 cal.beta, 15, 16, 17
 cal.claytonmix, 15, 16, 17
 cal.probitnorm, 15, 16, 17
 claytonmix, 18
 cov2cor, 19
 CovToCor, 19

Danish, 19
 danish (Danish), 19
 dbeta, 18, 82
 dbeta (beta (stats)), 12
 dclaytonmix, 82
 dclaytonmix (claytonmix), 18
 dcopula.clayton, 20, 21–23
 dcopula.gauss, 20, 21, 22, 23
 dcopula.gumbel, 20, 21, 21, 23
 dcopula.t, 20, 21, 22, 22
 dgamma, 13
 dGEV (GEV), 53
 dghyp, 12, 56, 84
 dghyp (ghyp), 54
 dghypB, 55
 dghypB (ghypB), 55

- dGPD (*GPD*), 57
- dGumbel (*Gumbel*), 58
- DJ (*DJComponents*), 23
- DJComponents, 23
- dji, 23
- dmghyp, 12, 24–26, 55
- dmghyp (*mghyp*), 67
- dmnorm, 14, 21, 24, 25, 46, 68
- dmt, 14, 23, 24, 24, 68
- dprobitnorm, 17, 18
- dprobitnorm (*probitnorm*), 81
- dsmghyp, 25, 68

- edf, 26
- EGIG, 27, 29
- eigenmeth, 28
- ElogGIG, 28, 29
- EMupdate, 29, 43, 65
- equicorr, 19, 30, 97, 98
- ESnorm, 31, 32
- ESst, 31, 32, 84
- extremalPP, 33, 47, 76, 113

- findthreshold, 34
- fit.Archcopula2d, 20, 22, 34, 40, 51, 52
- fit.binomial, 35, 36, 38, 39
- fit.binomialBeta, 36, 36, 38, 39, 73
- fit.binomialLogitnorm, 36, 37, 39, 73
- fit.binomialProbitnorm, 36, 38, 38, 73
- fit.gausscopula, 35, 39, 51, 52, 74, 75
- fit.GEV, 40, 42, 54, 57
- fit.GPD, 34, 41, 41, 42, 47, 54, 57, 66, 80, 81, 95, 103, 115
- fit.GPDb, 42, 42
- fit.mNH, 30, 43, 45, 46, 50, 65, 66
- fit.mst, 43, 44, 46, 50
- fit.NH, 43, 45, 45, 50
- fit.norm, 46
- fit.POT, 47
- fit.seMPP, 33, 48, 49, 77, 101, 102, 108, 113, 114
- fit.sePP, 33, 48, 49, 77, 101, 102, 108, 113, 114
- fit.st, 45, 46, 50
- fit.tcopula, 35, 40, 51, 52, 74, 75
- fit.tcopula.rank, 28, 51, 62
- ftsel00, 53
- FXGBP.RAW, 53

- gamma, 83
- GEV, 53
- ghyp, 54
- ghypB, 55
- GPD, 57
- Gumbel, 58

- hessb, 58
- hillPlot, 59
- hsi, 60

- jointnormalTest, 61, 64

- Kendall, 62, 108
- kurtosisSPlus, 62

- lbeta, 63

- MardiaTest, 61, 64
- MCECM.Qfunc, 43, 64
- MCECMupdate, 43, 65, 65
- MEplot, 66, 80, 81, 115
- mghyp, 67
- mk.its.exceedances.ts, 68, 71
- mk.its.exceedances.vector, 69, 70
- mk.oldts, 71
- mk.returns, 69, 71, 72
- momest, 72, 106, 107

- nasdaq, 73
- nikkei, 74

- pbeta (*beta (stats)*), 12
- pclaytonmix (*claytonmix*), 18
- Pconstruct, 74, 75
- Pdeconstruct, 74, 75
- pGEV, 41, 42, 57
- pGEV (*GEV*), 53
- pGPD, 41, 42, 54
- pGPD (*GPD*), 57
- pGumbel (*Gumbel*), 58
- plot.MPP, 75
- plot.PP, 76
- plot.sePP, 48, 49, 77
- plot.timeSeries, 78
- plot.timeSeriesIts, 79
- plotFittedGPDvsEmpiricalExcesses, 80
- plotTail, 60, 80, 81, 103
- pprobitnorm (*probitnorm*), 81
- probitnorm, 81
- profileLoadLibrary, 2, 3, 4, 4
- psifunc, 83

- qbeta (*beta (stats)*), 12
- qGEV (*GEV*), 53
- qGPD (*GPD*), 57

- qGumbel (*Gumbel*), 58
- qqnorm, 84
- QQplot, 83
- QRMBBook-workspace, 2, 4–6
- QRMBBook-workspace, 2
- QRMLib (*QRMLib*-package), 1
- QRMLib-package, 1
- qst, 84
- qt, 84

- rAC, 85, 86–94
- rBB9Mix, 85
- rbeta, 87, 96, 100
- rbeta (*beta (stats)*), 12
- rbinomial.mixture, 15–17, 86
- rclaytonmix, 16, 96, 100
- rclaytonmix (*claytonmix*), 18
- rcopula.AGumbel, 87, 91, 92
- rcopula.clayton, 85, 88, 89–91, 93
- rcopula.frank, 85, 88, 89, 90, 91, 93
- rcopula.gauss, 85, 88, 89, 89, 91, 93
- rcopula.gumbel, 85, 88, 89, 90, 90, 93, 99
- rcopula.Gumbel2Gp, 87, 91, 92
- rcopula.GumbelNested, 87, 91, 92
- rcopula.t, 85, 88–91, 92
- rFrankMix, 93
- rGEV (*GEV*), 53
- rghyp, 94
- rghyp (*ghyp*), 54
- rghypB (*ghypB*), 55
- rGIG, 12, 28, 29, 94
- rGPD (*GPD*), 57
- RiskMeasures, 42, 95, 103
- rlogitnorm, 87, 96, 100
- rmghyp, 94, 98
- rmghyp (*mghyp*), 67
- rmnorm, 31, 97, 98
- rmt, 31, 97, 98
- rprobitnorm, 87, 96, 100
- rprobitnorm (*probitnorm*), 81
- rstable, 99
- rtcopulamix, 100

- seMPP.negloglik, 101
- sePP.negloglik, 102
- showRM, 95, 103
- signalSeries, 10, 104
- smi, 105
- sp500, 105
- spdata, 106, 106
- spdata.raw, 105, 107
- Spearman, 62, 107
- Special, 13, 63
- stationary.sePP, 48, 49, 108
- storeDataInWorkspace, 2, 3, 3, 6
- symmetrize, 109

- timeSeriesClass, 69, 71, 72, 109, 112
- TimeSeriesClassRMetrics, 72, 110, 110

- unmark, 33, 76, 112

- volfunction, 113

- xdax, 114
- xiplot, 60, 80, 114