

# Package ‘fBasics’

June 17, 2022

**Title** Rmetrics - Markets and Basic Statistics

**Date** 2022-06-16

**Version** 3042.89.2

**Author** Diethelm Wuertz [aut],  
Tobias Setz [cre],  
Yohan Chalabi [ctb]  
Martin Maechler [ctb],  
CRAN Team [ctb]

**Maintainer** Tobias Setz <tobias.setz@live.com>

**Description** Provides a collection of functions to explore and to investigate basic properties of financial returns and related quantities. The covered fields include techniques of explorative data analysis and the investigation of distributional properties, including parameter estimation and hypothesis testing. Even more there are several utility functions for data handling and management.

**Depends** R (>= 2.15.1), timeDate, timeSeries

**Imports** stats, grDevices, graphics, methods, utils, MASS, spatial, gss, stabledist

**Suggests** interp, RUnit, tcltk

**LazyData** yes

**License** GPL (>= 2)

**Encoding** UTF-8

**URL** <https://www.rmetrics.org>

**NeedsCompilation** yes

**Repository** CRAN

**Date/Publication** 2022-06-17 10:50:58

**R topics documented:**

fBasics-package . . . . .	4
acfPlot . . . . .	13
akimaInterp . . . . .	15
baseMethods . . . . .	17
BasicStatistics . . . . .	18
BoxPlot . . . . .	19
characterTable . . . . .	20
colorLocator . . . . .	20
colorPalette . . . . .	21
colorTable . . . . .	25
colVec . . . . .	25
correlationTest . . . . .	26
decor . . . . .	28
distCheck . . . . .	29
DistributionFits . . . . .	29
fBasics-deprecated . . . . .	31
fBasicsData . . . . .	32
fHTEST . . . . .	34
getS4 . . . . .	35
gh . . . . .	36
ghFit . . . . .	38
ghMode . . . . .	40
ghMoments . . . . .	41
ghRobMoments . . . . .	42
ghSlider . . . . .	43
ght . . . . .	44
ghtFit . . . . .	45
ghtMode . . . . .	46
ghtMoments . . . . .	47
ghtRobMoments . . . . .	48
gld . . . . .	49
gldFit . . . . .	51
gldMode . . . . .	52
gldRobMoments . . . . .	53
gridVector . . . . .	54
Heaviside . . . . .	55
hilbert . . . . .	56
HistogramPlot . . . . .	57
hyp . . . . .	59
hypFit . . . . .	61
hypMode . . . . .	62
hypMoments . . . . .	63
hypRobMoments . . . . .	65
hypSlider . . . . .	66
Ids . . . . .	66
interactivePlot . . . . .	67

inv . . . . .	68
krigeInterp . . . . .	69
kron . . . . .	70
ks2Test . . . . .	71
lcg . . . . .	72
linearInterp . . . . .	74
listDescription . . . . .	75
listFunctions . . . . .	76
listIndex . . . . .	76
locationTest . . . . .	77
maxdd . . . . .	79
nig . . . . .	81
nigFit . . . . .	82
nigMode . . . . .	84
nigMoments . . . . .	85
nigRobMoments . . . . .	86
nigShapeTriangle . . . . .	87
nigSlider . . . . .	88
norm . . . . .	89
NormalityTests . . . . .	90
normRobMoments . . . . .	94
pascal . . . . .	95
pdl . . . . .	96
positiveDefinite . . . . .	97
print . . . . .	97
QuantileQuantilePlots . . . . .	98
ReturnSeriesGUI . . . . .	100
rk . . . . .	100
rowStats . . . . .	101
sampleLMoments . . . . .	103
sampleRobMoments . . . . .	103
scaleTest . . . . .	104
ScalingLawPlot . . . . .	106
sgh . . . . .	108
sghFit . . . . .	109
sght . . . . .	110
snig . . . . .	112
snigFit . . . . .	113
ssd . . . . .	114
ssdFit . . . . .	116
StableSlider . . . . .	117
symbolTable . . . . .	117
TimeSeriesPlots . . . . .	118
tr . . . . .	120
triang . . . . .	120
tsHessian . . . . .	121
tslag . . . . .	122
varianceTest . . . . .	122

vec . . . . . 124

**Index** **126**

fBasics-package      *Portfolio Modelling, Optimization and Backtesting*

## Description

The Rmetrics "fBasics" package is a collection of functions to explore and to investigate basic properties of financial returns and related quantities.

The covered fields include techniques of explorative data analysis and the investigation of distributional properties, including parameter estimation and hypothesis testing. Evenmore there are several utility functions for data handling and managemnet.

## Details

```
Package:  \tab fBasics\cr
Type:    \tab Package\cr
Date:    \tab 2014\cr
License: \tab GPL Version 2 or later\cr
Copyright: \tab (c) 1999-2014 Rmetrics Association\cr
Repository: \tab R-FORGE\cr
URL:     \tab \url{https://www.rmetrics.org}
```

## 1 Introduction

The fBasics package contains *basics tools* often required in computational finance and financial engineering. The topics are: basic statistics functions, financial return distributions, hypothesis testing, plotting routines, matrix computations and linear algebra, and some usefule utility functions.

## 2 Basic Statistics Functions

### *Financial Return Statistics*

basicStats      Returns a basic statistics summary

### *Distribution Function of Maximum Drawdowns*

```
dmaxdd      Density function of mean Max-Drawdowns
pmaxdd      Probability function of mean Max-Drawdowns
rmaxdd      Random Variates of mean Max-Drawdowns
maxddStats      Expectation of Drawdowns for BM with drift
```

### *Calculation of Sample Moments*

sampleLmoments	Computes sample L-moments
sampleMED	Returns sample median
sampleIQR	returns sample inter quartal range
sampleSKEW	returns robust sample skewness
sampleKURT	returns robust sample kurtosis

*Bivariate Interpolation:*

akimaInterp	Interpolates irregularly spaced points
akimaInterpp	Interpolates and smoothes pointwise
krigeInterp	Kriges irregularly spaced data points
linearInterp	Interpolates irregularly spaced points
linearInterpp	Interpolates linearly pointwise

*Utility Statistics Functions:*

colStats	Computes sample statistics by col
colSums	Computes sums of values in each col
colMeans	Computes means of values in each col
colSds	Computes standard deviation of each col
colVars	Computes sample variance by col
colSkewness	Computes sample skewness by col
colKurtosis	Computes sample kurtosis by col
colMaxs	Computes maximum values in each col
colMins	Computes minimum values in each col
colProds	Computes product of values in each col
colQuantiles	Computes product of values in each col
rowStats	Computes sample statistics by row
rowSums	Computes sums of values in each row
rowMeans	Computes means of values in each row
rowSds	Computes standard deviation of each row
rowVars	Computes sample variance by row
rowSkewness	Computes sample skewness by row
rowKurtosis	Computes sample kurtosis by row
rowMaxs	Computes maximum values in each row
rowMins	Computes minimum values in each row
rowProds	Computes product of values in each row
rowQuantiles	Computes product of values in each row

**3 Financial Return Distributions***Generalized Hyperbolic Distribution:*

dghReturns	Density for the GH distribution
pghreturns	Probability for the GH distribution
qghreturns	Quantiles for the GH distribution
rghreturns	Random variates for the GH distribution
ghFitFits	Fits parameters of the GH distribution
ghMode	Computes mode of the GH distribution.
ghMean	Returns true mean of the GH distribution
ghVar	Returns true variance of the GH distribution
ghSkew	Returns true skewness of the GH distribution
ghKurt	Returns true kurtosis of the GH distribution
ghMoments	Returns true n-th moment of the GH distribution
ghMED	Returns true median of te GH distribution
ghIQR	Returns true inter quartal range of te GH
ghSKEW	Returns true robust skewness of te GH
ghKURT	Returns true robust kurtosis of te GH

*Hyperbolic Distribution:*

dhyp	Returns density for the HYP distribution
phyp	Returns probability for the HYP distribution
qhyp	Returns quantiles for the HYP distribution
rhyp	Returns random variates for the HYP distribution
hypFit	Fits parameters of the HYP distribution
hypMode	Computes mode of the HYP distribution
hypMean	Returns true mean of the HYP distribution
hypVar R	Returns true variance of the HYP distribution
hypSkew	Returns true skewness of the HYP distribution
hypKurt	Returns true kurtosis of the HYP distribution
hypMoments	Returns true n-th moment of the HYP distribution
hypMED	Returns true median of the HYP distribution
hypIQR	Returns true inter quartal range of the HYP
hypSKEW	Returns true robust skewness of the HYP
hypKURT	Returns true robust kurtosis of the HYP

*Normal Inverse Gaussian:*

dnig	Returns density for the NIG distribution
pnig	Returns probability for the NIG distribution
qnig	Returns quantiles for the NIG distribution
rnig	Returns random variates for the NIG distribution
.pnigC	fast C Implementation of function pnig()
.qnigC	fast CImplementation of function qnig()
nigFit	Fits parameters of a NIG distribution
.nigFit.mle	Uses max Log-likelihood estimation
.nigFit.gmm	Uses generalized method of moments
.nigFit.mps	Maximum product spacings estimation

.nigFit.vmps	Minimum variance mps estimation
nigMode	Computes mode of the NIG distribution
nigMean	Returns true mean of the NIG distribution
nigVar	Returns true variance of the NIG distribution
nigSkew	Returns true skewness of the NIG distribution
nigKurt	Returns true kurtosis of the NIG distribution
nigMoments	Returns true n-th moment of the NIG distribution
nigMED	Returns true median of the NIG distribution
nigIQR	Returns true inter quartal range of the NIG
nigSKEW	Returns true robust skewness of the NIG
nigKURT	Returns true robust kurtosis of the NIG

*Generalized Hyperbolic Student-t Distribution:*

dght	Returns density for the GHT distribution
pght	Returns probability for the GHT distribution
qght	Returns quantiles for the GHT distribution
rght	Returns random variates for the GHT distribution
ghtFit	Fits parameters of the GHT distribution
ghtMode	Computes mode of the GHT distribution
ghtMean	Returns true mean of the NIG distribution
ghtVar	Returns true variance of the GHT distribution
ghtSkew	Returns true skewness of the GHT distribution
ghtKurt	Returns true kurtosis of the GHT distribution
ghtMoments	Returns true n-th moment of the GHT distribution
ghtMED	Returns true median of the GHT distribution
ghtIQR	Returns true inter quartal range of the GHT
ghtSKEW	Returns true robust skewness of the GHT
ghtKURT	Returns true robust kurtosis of the GHT

*Stable Distribution:*

dstable	Returns density for the stable distribution
pstable	Returns probability for the stable distribution
qstable	Returns quantiles for the stable distribution
rstable	Returns random variates for the dtsble distribution
stableFit	Fits parameters of a the stable distribution
.phiStable	Creates contour table for McCulloch estimators
.PhiStable	Contour table created by function .phiStable()
.qStableFit	Estimates parameters by McCulloch's approach
.mleStableFit	Estimates stable parameters by MLE approach
.stablePlot	Plots results of stable parameter estimates
stableMode	Computes mode of the stable distribution

*Generalized Lambda Distribution:*

dgld	Returns density for the GLD distribution
pgld	Returns probability for the GLD distribution
qgld	Returns quantiles for the GLD distribution
rgld	Returns random variates for the GLD distribution
gldFit	Fits parameters of the GLD distribution
.gldFit.mle	fits GLD using maximum log-likelihood
.gldFit.mps	fits GLD using maximum product spacings
.gldFit.gof	fits GLD using Goodness of Fit statistics
.gldFit.hist	fits GLD using a histogram fit
.gldFit.rob	fits GLD using robust moments fit
gldMode	Computes mode of the GLD distribution.
gldMED	Returns true median of the GLD distribution
gldIQR	Returns true inter quartal range of the GLD
gldSKEW	Returns true robust skewness of the GLD
gldKURT	Returns true robust kurtosis of the GLD

#### *Spline Smoothed Distribution:*

dssd	Returns spline smoothed density function
pssd	Returns spline smoothed probability function
qssd	Returns spline smoothed quantile function
rssd	Returns spline smoothed random variates.
ssdFit	Fits parameters for a spline smoothed distribution

## **4 Hypthesis Testing**

### *One Sample Normality Tests:*

ksnormTest	One sample Kolmogorov-Smirnov normality test
shapiroTest	Shapiro-Wilk normality test
jarqueberaTest	Jarque-Bera normality test
normalTest	Normality tests S-Plus compatible call
dagoTest	D'Agostino normality test
adTest	Anderson-Darling normality test
cvmTest	Cramer-von Mises normality test
lillieTest	Lilliefors (KS) normality test
pchiTest	Pearson chi-square normality test
sfTest	Shapiro-Francia normality test
jbTest	Finite sample adjusted JB LM and ALM test

### *One Sample Location, Scale and variance Tests:*



locationTest	Performs locations tests on two samples
.tTest	Unpaired t test for differences in mean
.kw2Test	Kruskal-Wallis test for differences in locations
scaleTest	Performs scale tests on two samples
.ansariTest	Ansari-Bradley test for differences in scale
.moodTest	Mood test for differences in scale
varianceTest	Performs variance tests on two samples
.varfTest	F test for differences in variances
.bartlett2Test	Bartlett's test for differences in variances
.fligner2Test	Fligner-Killeen test for differences in variances

*Two Sample Tests:*

ks2Test	Performs a two sample Kolmogorov-Smirnov test
correlationTest	Performs correlation tests on two samples
pearsonTest	Pearson product moment correlation coefficient
kendallTest	Kendall's tau correlation test
spearmanTest	Spearman's rho correlation test

*Test Utilities:*

'fHTEST'	S4 Class Representation
show.fHTEST	S4 Print Method
.jbALM	Jarque Bera Augmented Lagrange Multiplier Data
.jbLM	Jarque-Bera Lagrange Multiplier Data
.jbTable	Finite sample p values for the Jarque Bera test
.jbPlot	Plots probabilities
.pjb	Returns probabilities for JB given quantiles
.qjb	Returns quantiles for JB given probabilities

**5 Plotting Routines***Financial Time Series Plots:*

seriesPlot	Displays a time series plot
cumulatedPlot	Displays cumulated series give returns
returnPlot	Displays returns given cumulated series
drawdownPlot	Displays drawdown series from returns

*Correlation Plots:*

acfPlot	Displays tailored ACF plot
pacfPlot	Displays tailored partial ACF plot
teffectPlot	Displays the Taylor effect
lacfPlot	Displays lagged autocorrelations

*Distribution Plots:*

histPlot	Returns tailored histogram plot
densityPlot	Returns tailored density plot
logDensityPlot	Returns tailored log density plot
boxPlot	Returns side-by-side standard box plot
boxPercentile	Plotreturns box-percentile plot
qqnormPlot	Returns normal quantile-quantile plot
qqnigPlot	Returns NIG quantile-quantile plot
qqghtPlot	Returns GHT quantile-quantile plot
qqgldPlot	Returns GLD quantile-quantile plot

*Time Series Aggregation Plots:*

scalinglawPlot	Displays scaling law behavior
----------------	-------------------------------

**5. Matrix Computations and Linear Algebra***Elementar Matrix Operation Addons:*

kron	Returns the Kronecker product
vec	Stacks a matrix as column vector
vech	Stacks a lower triangle matrix
pdl	Returns regressor matrix for polynomial lags
tslag	Returns Lagged/leading vector/matrix

*Linear Algebra Addons:*

inv	Returns the inverse of a matrix
norm	Returns the norm of a matrix
rk	Returns the rank of a matrix
tr	Returns the trace of a matrix

*General Matrix Utility Addons:*

isPositiveDefinite	Checks if a matrix is positive definite
makePositiveDefinite	Forces a matrix to be positive definite
colVec	Creates a column vector from a data vector
rowVec	Creates a row vector from a data vector
gridVector	Creates from two vectors rectangular grid
triang	Extracts lower tridiagonal part from a matrix
Triang	Extracts upper tridiagonal part from a matrix

*Selected Matrix Examples:*

hilbert	Creates a Hilbert matrix
pascal	Creates a Pascal matrix

**6 Utility Functions***Color Utilities:*

colorLocator	Plots Rs 657 named colors for selection
colorMatrix	Returns matrix of R's color names.
colorTable	Table of Color Codes and Plot Colors itself
rainbowPalette	Contiguous rainbow color palette
heatPalette	Contiguous heat color palette
terrainPalette	Contiguous terrain color palette
topoPalette	Contiguous topo color palette
cmPalette	Contiguous cm color palette
greyPalette	R's gamma-corrected gray palette
timPalette	Tim's Matlab like color palette
rampPalette	Color ramp palettes
seqPalette	Sequential color brewer palettes
divPalette	Diverging color brewer palettes
qualiPalette	Qualified color brewer palettes
focusPalette	Red, green blue focus palettes
monoPalette	Red, green blue mono palettes

*Graphics Utilities:*

symbolTable	Shows a table of plot symbols
characterTable	Shows a table of character codes
decor	Adds horizontal grid and L shaped box
hgrid	Adds horizontal grid lines
vgrid	Adds vertical grid lines
boxL	Adds L-shaped box
box	Adds unterlined box
.xrug	Adds rugs on x axis
.yrug	Adds rugs on y axis
copyright	Adds copyright notice
interactivePlot	Plots several graphs interactively

*Special Function Utilities:*

Heaviside	Computes Heaviside unit step function
Sign	Another signum function
Delta	Computes delta function

Boxcar	Computes boxcar function
Ramp	Computes ramp function
tsHessian	Computes Two Sided Hessian matrix

*Other Utilities:*

.unirootNA	Computes zero of a function without error exit
getModel	Extracts the model slot from a S4 object
getTitle	Extracts the title slot from a S4 object
getDescription	Extracts the description slot
getSlot	Extracts a specified slot from a S4 object

### About Builtin Functions

Builtin functions are borrowed from contributed R packages and other sources. There are several reasons why we have modified and copied code from other sources and included in this package.

- \* The builtin code is not available on Debian, so that Linux users have no easy access to this code.
- \* The original code conflicts with other code from this package or conflicts with Rmetrics design objectives.
- \* We only need a very small piece of functionality from the original package which may depend on other packages which are not needed.
- \* The package from which we builtin the code is under current development, so that the functions often change and thus leads to unexpected behavior in the Rmetrics packages.
- \* The package may be incompatible since it uses other time date and time series classes than the 'timDate' and 'timeSeries' objects and methods from Rmetrics.

We put the code in script files named *builtin-funPackage.R* where "fun" denotes the (optional) major function name, and "Package" the name of the contributed package from which we copied the original code.

Builtin functions include:

gelGmm	gll function from gmm package
gmmGMM	gmm function from gmm package
kweightsSandwhich	kweights from sandwich package
glGld	gl functions from gld package
ssdenGss	ssden from the gss package
hypHyperbolicDist	hyp from HyperbolicDist package

### Compiled Fortran and C Code:

gld.c	source code from gld package
nig.c	source code from Kersti Aas
gss.f	source code from sandwich package

**About Rmetrics:**

The fBasics Rmetrics package is written for educational support in teaching "Computational Finance and Financial Engineering" and licensed under the GPL.

---

acfPlot *Autocorrelation Function Plots*

---

**Description**

Returns plots of autocorrelations including the autocorrelation function ACF, the partial ACF, the lagged ACF, and the Taylor effect plot.

The functions to display stylized facts are:

acfPlot	autocorrelation function plot,
pacfPlot	partial autocorrelation function plot,
lacfPlot	lagged autocorrelation function plot,
teffectPlot	Taylor effect plot.

**Usage**

```
acfPlot(x, labels = TRUE, ...)
pacfPlot(x, labels = TRUE, ...)
```

```
lacfPlot(x, n = 12, lag.max = 20, type = c("returns", "values"),
         labels = TRUE, ...)
```

```
teffectPlot(x, deltas = seq(from = 0.2, to = 3, by = 0.2), lag.max = 10,
            ymax = NA, standardize = TRUE, labels = TRUE, ...)
```

**Arguments**

deltas	the exponents, a numeric vector, by default ranging from 0.2 to 3.0 in steps of 0.2.
labels	a logical value. Whether or not x- and y-axes should be automatically labeled and a default main title should be added to the plot. By default TRUE.
lag.max	maximum lag for which the autocorrelation should be calculated, an integer.
n	an integer value, the number of lags.
standardize	a logical value. Should the vector x be standardized?
type	[lacf] - a character string which specifies the type of the input series, either "returns" or series "values". In the case of a return series as input, the required value series is computed by cumulating the financial returns: <code>exp(colCumsums(x))</code>

x	an uni- or multivariate return series of class <code>timeSeries</code> or any other object which can be transformed by the function <code>as.timeSeries()</code> into an object of class <code>timeSeries</code> .
ymax	maximum y-axis value on plot, <code>is.na(ymax) TRUE</code> , then the value is selected automatically.
...	arguments to be passed.

## Details

### Autocorrelation Functions:

The functions `acfPlot` and `pacfPlot`, `plot` and `estimate` autocorrelation and partial autocorrelation function. The functions allow to get a first view on correlations within the time series. The functions are synonyme function calls for R's `acf` and `pacf` from the `ts` package.

### Taylor Effect:

The "Taylor Effect" describes the fact that absolute returns of speculative assets have significant serial correlation over long lags. Even more, autocorrelations of absolute returns are typically greater than those of squared returns. From these observations the Taylor effect states, that that the autocorrelations of absolute returns to the the power of  $\delta$ ,  $\text{abs}(x - \text{mean}(x))^\delta$  reach their maximum at  $\delta=1$ . The function `teffect` explores this behaviour. A plot is created which shows for each lag (from 1 to `max.lag`) the autocorrelations as a function of the exponent  $\delta$ . In the case that the above formulated hypothesis is supported, all the curves should peak at the same value around  $\delta=1$ .

## Value

`acfPlot`, `pacfplot`,  
return an object of class "acf", see [acf](#).

`lacfPlot` returns a list with the following two elements: `Rho`, the autocorrelation function, `lagged`, the lagged correlations.

`teffectPlot`  
returns a numeric matrix of order `deltas` by `max.lag` with the values of the autocorrelations.

## References

Taylor S.J. (1986); *Modeling Financial Time Series*, John Wiley and Sons, Chichester.

Ding Z., Granger C.W.J., Engle R.F. (1993); *A long memory property of stock market returns and a new model*, Journal of Empirical Finance 1, 83.

## Examples

```
## data -
  SPI <- LPP2005REC[, "SPI"]
```

```

plot(SPI, type = "l", col = "steelblue", main = "SP500")
abline(h = 0, col = "grey")

## teffectPlot -
# Taylor Effect:
teffectPlot(SPI)

```

---

akimaInterp

*Bivariate Spline Interpolation*


---

## Description

Interpolates bivariate data sets using Akima spline interpolation.

## Usage

```

akimaInterp(x, y = NULL, z = NULL, gridPoints = 21,
            xo = seq(min(x), max(x), length = gridPoints),
            yo = seq(min(y), max(y), length = gridPoints), extrap = FALSE)

akimaInterpp(x, y = NULL, z = NULL, xo, yo, extrap = FALSE)

```

## Arguments

x, y, z	for akimaInterp the arguments x and y are two numeric vectors of grid points, and z is a numeric matrix or any other rectangular object which can be transformed by the function <code>as.matrix</code> into a matrix object. For akimaInterpp we consider either three numeric vectors of equal length or if y and z are NULL, a list with entries x, y, z, or named data frame with x in the first, y in the second, and z in the third column.
gridPoints	an integer value specifying the number of grid points in x and y direction.
xo, yo	for akimaInterp two numeric vectors of data points spanning the grid, and for akimaInterpp two numeric vectors of data points building pairs for pointwise interpolation.
extrap	a logical, if TRUE then the data points are extrapolated.

## Details

Two options are available gridded and pointwise interpolation.

akimaInterp is a function wrapper to the interp function provided by the contributed R package akima. The Fortran code of the Akima spline interpolation routine was written by H. Akima.

Linear surface fitting and krige surface fitting are provided by the functions linearInterp and krigeInterp.

**Value**

**akimaInterp** returns a list with at least three entries, x, y and z. Note, that the returned values, can be directly used by the persp and contour 3D plotting methods.

**akimaInterpp** returns a data.frame with columns "x", "y", and "z".

**Note**

IMPORTANT: The contributed package **akima** is not in the dependence list of the DESCRIPTION file due to license conditions. The user has to install this package from CRAN on his own responsibility, please check the license conditions.

**References**

Akima H., 1978, *A Method of Bivariate Interpolation and Smooth Surface Fitting for Irregularly Distributed Data Points*, ACM Transactions on Mathematical Software 4, 149-164.

Akima H., 1996, *Algorithm 761: Scattered-Data Surface Fitting that has the Accuracy of a Cubic Polynomial*, ACM Transactions on Mathematical Software 22, 362-371.

**See Also**

[linearInterp](#), [krigeInterp](#).

**Examples**

```
## Does not run for r-solaris-x86
## akimaInterp -- Akima Interpolation:
if (requireNamespace("interp")) {
  set.seed(1953)
  x <- runif(999) - 0.5
  y <- runif(999) - 0.5
  z <- cos(2*pi*(x^2+y^2))
  ans <- akimaInterp(x, y, z, gridPoints = 41, extrap = FALSE)
  persp(ans, theta = -40, phi = 30, col = "steelblue",
        xlab = "x", ylab = "y", zlab = "z")
  contour(ans)
}

## Use spatial as alternative on r-solaris-x86
## spatialInterp - Generate Kriged Grid Data:
if (requireNamespace("spatial")) {
  RNGkind(kind = "Marsaglia-Multicarry", normal.kind = "Inversion")
  set.seed(4711, kind = "Marsaglia-Multicarry")
  x <- runif(999)-0.5
  y <- runif(999)-0.5
  z <- cos(2*pi*(x^2+y^2))
  ans <- krigeInterp(x, y, z, extrap = FALSE)
  persp(ans)
  title(main = "Kriging")
  contour(ans)
}
```



```

    title(main = "Kriging")
  }

```

---

baseMethods

*Generic Functions Extensions*


---

## Description

Basic extensions which which add and/or modify additional functionality which is not available in R's basic packages.

Added and/or modified functions:

attach	extends attach function,
rank	extends rank function,
stdev	adds stdev function,
termPlot	adds term plot function,
volatility	adds volatility function.

## Usage

```
## Default S3 method:
stdev(x, na.rm = FALSE)
```

```
## Default S3 method:
termPlot(model, ...)
```

```
## Default S3 method:
volatility(object, ...)
```

## Arguments

na.rm	an logical value - should the NA values be removed.
model	a fitted model object.
object	an object from which to extract the volatility.
x	[align] - x-coordinates of the points to be aligned. [log][sort][var] - first argument. [print.control] - cr prints an unlisted object of class control. [as.matrix.ts][as.matrix.mts] - an univariate or multivariate time series object of class "ts" or "mts" which will be transformed into an one-column or multi-column rectangular object of class "matrix". [as.POSIXlt] - an object to be converted.

... arguments to be passed.

### Details

For details we refer to the original help pages.

---

BasicStatistics      *Basic Time Series Statistics*

---

### Description

Computes basic financial time series statistics.

List of Functions:

`basicStats`    Computes an overview of basic statistical values.

### Usage

```
basicStats(x, ci = 0.95)
```

### Arguments

`ci`            confidence interval, a numeric value, by default 0.95, i.e. 95 percent.

`x`             an object of class "timeSeries" or any other object which can be transformed by the function `as.timeSeries` into an object of class `timeSeries`. The latter case, other than `timeSeries` objects, is more or less untested.

### Value

`basicsStats`  
returns a data frame with the following entries and row names: nobs, NAs, Minimum, Maximum, 1. Quartile, 3. Quartile, Mean, Median, Sum, SE Mean, LCL Mean, UCL Mean, Variance, Stdev, Skewness, Kurtosis.

### Examples

```
## basicStats -  
# Simulated Monthly Return Data:  
tS = timeSeries(matrix(rnorm(12)), timeCalendar())  
basicStats(tS)
```

---

BoxPlot	<i>Time Series Box Plots</i>
---------	------------------------------

---

**Description**

Returns a box or a box percentile plot.

List of Functions:

boxPlot	Returns a side-by-side standard box plot,
boxPercentilePlot	Returns a side-by-side box-percentile plot.

**Usage**

```
boxPlot(x, col = "steelblue", title = TRUE, ...)
boxPercentilePlot(x, col = "steelblue", title = TRUE, ...)
```

**Arguments**

col	the color for the series. In the univariate case use just a color name like the default, col="steelblue", in the multivariate case we recommend to select the colors from a color palette, e.g. col=heat.colors(ncol(x)).
title	a logical flag, by default TRUE. Should a default title added to the plot?
x	an object of class "timeSeries" or any other object which can be transformed by the function as.timeSeries into an object of class timeSeries. The latter case, other then timeSeries objects, is more or less untested.
...	optional arguments to be passed.

**Value**

displays a time series plot.

**Examples**

```
## data -
LPP <- LPP2005REC[, 1:6]
plot(LPP, type = "l", col = "steelblue", main = "SP500")
abline(h = 0, col = "grey")

## boxPlot -
boxPlot(LPP)
```

---

characterTable      *Table of Characters*

---

### Description

Displays a table of numerical equivalents to Latin characters.

### Usage

```
characterTable(font = 1, cex = 0.7)
```

### Arguments

`cex`                a numeric value, determines the character size, the default size is 0.7.  
`font`                an integer value, the number of the font, by default font number 1.

### Value

characterTable  
 displays a table with the characters of the requested font. The character on line "xy" and column "z" of the table has code "\xyz", e.g `cat("\126")` prints: V for font number 1. These codes can be used as any other characters.

### See Also

[link{colorTable}](#), [link{symbolTable}](#).

### Examples

```
## Character Table for Font 2:
characterTable(font = 1)
```

---

colorLocator      *Color Selection*

---

### Description

Displays R's 657 named colors for selection and returns optionally R's color names.

### Usage

```
colorLocator(locator = FALSE, cex.axis = 0.7)
colorMatrix()
```

**Arguments**

locator	logical, if true, <a href="#">locator</a> is used for interactive selection of color names, default is FALSE.
cex.axis	size of axis labels.

**Details**

Color Locator:

The `colorLocator` function plots R's 657 named colors. If `locator=TRUE` then you can interactively point and click to select the colors for which you want names. To end selection, right click on the mouse and select 'Stop', then R returns the selected color names.

The functions used here are wrappers to the functions provided by Tomas Aragon in the contributed R package. `epitools`.

**Value**

Color Locator:

`colorsLocator()` generates a plot with R colors and, when `locator` is true, returns matrix with graph coordinates and names of colors selected. `colorsMatrix()` quietly returns the matrix of names.

**See Also**

[colorPalette](#), [colorTable](#).

**Examples**

```
colorLocator()
```

---

colorPalette

*Color Palettes*

---

**Description**

Functions to create color palettes.

The functions are:

<code>rainbowPalette</code>	Contiguous rainbow color palette,
<code>heatPalette</code>	Contiguous heat color palette,
<code>terrainPalette</code>	Contiguous terrain color palette,
<code>topoPalette</code>	Contiguous topo color palette,
<code>cmPalette</code>	Contiguous cm color palette,
<code>greyPalette</code>	R's gamma-corrected gray palette,
<code>timPalette</code>	Tim's Matlab like color palette,
<code>rampPalette</code>	Color ramp palettes,

seqPalette	Sequential color brewer palettes,
divPalette	Diverging color brewer palettes,
qualiPalette	Qualified color brewer palettes,
focusPalette	Red, green blue focus palettes,
monoPalette	Red, green blue mono palettes.

## Usage

```
rainbowPalette(n = 64, ...)
heatPalette(n = 64, ...)
terrainPalette(n = 64, ...)
topoPalette(n = 64, ...)
cmPalette(n = 64, ...)

greyPalette(n = 64, ...)
timPalette(n = 64)

rampPalette(n, name = c("blue2red", "green2red", "blue2green",
  "purple2green", "blue2yellow", "cyan2magenta"))

seqPalette(n, name = c(
  "Blues", "BuGn", "BuPu", "GnBu", "Greens", "Greys", "Oranges",
  "OrRd", "PuBu", "PuBuGn", "PuRd", "Purples", "RdPu", "Reds",
  "YlGn", "YlGnBu", "YlOrBr", "YlOrRd"))
divPalette(n, name = c(
  "BrBG", "PiYG", "PRGn", "PuOr", "RdBu", "RdGy", "RdYlBu", "RdYlGn",
  "Spectral"))
qualiPalette(n, name = c(
  "Accent", "Dark2", "Paired", "Pastel1", "Pastel2", "Set1", "Set2",
  "Set3"))

focusPalette(n, name = c("redfocus", "greenfocus", "bluefocus"))
monoPalette(n, name = c("redmono", "greenmono", "bluemono"))
```

## Arguments

n                    an integer, giving the number of greys or colors to be constructed.

name                a character string, the name of the color set.

...                 arguments to be passed, see the details section

## Details

All Rmetrics' color sets are named as fooPalette where the prefix foo denotes the name of the underlying color set.

### R's Contiguous Color Palettes:

Palettes for  $n$  contiguous colors are implemented in the `grDevices` package. To be conform with Rmetrics' naming convention for color palettes we have build a wrapper around the underlying functions. These are the `rainbowPalette`, `heatPalette`, `terrainPalette`, `topoPalette`, and the `cmPalette`. Conceptually, all of these functions actually use (parts of) a line cut out of the 3-dimensional color space, parametrized by the function `hsv(h, s, v, gamma)`, where `gamma=1` for the `fooPalette` function, and hence, equispaced hues in RGB space tend to cluster at the red, green and blue primaries. Some applications such as contouring require a palette of colors which do not wrap around to give a final color close to the starting one. To pass additional arguments to the underlying functions we refer to consult `help(rainbow)`. With `rainbow`, the parameters `start` and `end` can be used to specify particular subranges of hues. Synonyme function calls are `rainbow.colors`, `heat.colors`, `terrain.colors`, `topo.colors`, and the `cm.colors`.

### **R's Gamma-Corrected Gray Palette:**

The function `grayPalette` chooses a series of  $n$  gamma-corrected gray levels. The range of the gray levels can be optionally monitored through the `...` arguments, for details `help(gray.colors)`, which is a synonyme function call in the `grDevices` package.

### **Tim's Matlab like Color Palette:**

The function `timPalette` creates a color set ranging from blue to red, and passes through the colors cyan, yellow, and orange. It comes from the Matlab software, originally used in fluid dynamics simulations. The function here is a copy from R's contributed package `fields` doing a spline interpolation on  $n=64$  color points.

### **Color Ramp Palettes:**

The function `rampPalette` creates several color ramps. The function is implemented from Tim Keitt's contributed R package `colorRamps`. Supported through the argument `name` are the following color ramps: "blue2red", "green2red", "blue2green", "purple2green", "blue2yellow", "cyan2magenta".

### **Color Brewer Palettes:**

The functions `seqPalette`, `divPalette`, and `qualiPalette` create color sets according to R's contributed `RColorBrewer` package. The first letter in the function name denotes the type of the color set: "s" for sequential palettes, "d" for diverging palettes, and "q" for qualitative palettes.

*Sequential palettes* are suited to ordered data that progress from low to high. Lightness steps dominate the look of these schemes, with light colors for low data values to dark colors for high data values. The sequential palettes names are: Blues, BuGn, BuPu, GnBu, Greens, Greys, Oranges, OrRd, PuBu, PuBuGn, PuRd, Purples, RdPu, Reds, YlGn, YlGnBu, YlOrBr, YlOrRd.

*Diverging palettes* put equal emphasis on mid-range critical values and extremes at both ends of the data range. The critical class or break in the middle of the legend is emphasized with light colors and low and high extremes are emphasized with dark colors that have contrasting hues. The diverging palettes names are: BrBG, PiYG, PRGn, PuOr, RdBu, RdGy, RdYlBu, RdYlGn, Spectral.

*Qualitative palettes* do not imply magnitude differences between legend classes, and hues are used to create the primary visual differences between classes. Qualitative schemes are best suited to rep-

representing nominal or categorical data. The qualitative palettes names are: Accent, Dark2, Paired, Pastel1, Pastel2, Set1, Set2, Set3.

In contrast to the original color brewer palettes, the palettes here are created by spline interpolation from the color variation with the most different values, i.e for the sequential palettes these are 9 values, for the diverging palettes these are 11 values, and for the qualitative palettes these are between 8 and 12 values depending on the color set.

### Graph Color Palettes:

The function `perfanPalette` creates color sets inspired by R's contributed package Performance Analytics. These color palettes have been designed to create readable, comparable line and bar graphs with specific objectives.

*Focused Color Palettes:* Color sets designed to provide focus to the data graphed as the first element. This palette is best used when there is clearly an important data set for the viewer to focus on, with the remaining data being secondary, tertiary, etc. Later elements graphed in diminishing values of gray.

*Monochrome Color Palettes:* These include color sets for monochrome color displays.

### Value

returns a character string of color strings.

### Note

The palettes are wrapper functions provided in several contributed R packages. These include:

Cynthia Brewer and Mark Harrower for the brewer palettes,  
 Peter Carl and Brian G. Peterson for the "PerformanceAnalytics" package,  
 Tim Keitt for the "colorRamps" package,  
 Ross Ihaka for the "colorspace" package,  
 Tomas Aragon for the "epitools" package,  
 Doug Nychka for the "fields" package,  
 Erich Neuwirth for the "RColorBrewer" package.

Additional undocumented hidden functions:

<code>.asRGB</code>	Converts any R color to RGB (red/green/blue),
<code>.chcode</code>	Changes from one to another number system,
<code>.hex.to.dec</code>	Converts heximal numbers do decimal numbers,
<code>.dec.to.hex</code>	Converts decimal numbers do heximal numbers.

### Examples

```
## GreyPalette:
greyPalette()
```



---

colorTable	<i>Table of Colors</i>
------------	------------------------

---

**Description**

Displays a Table of color codes and plots the colors themselves.

**Usage**

```
colorTable(cex = 0.7)
```

**Arguments**

`cex` a numeric value, determines the character size in the color plot, the default size is 0.7.

**Value**

`colorTable`  
returns a table of plot colors with the associated color numbers.

**See Also**

`link{characterTable}`, `link{symbolTable}`.

**Examples**

```
## Color Table:  
colorTable()
```

---

colVec	<i>Column and Row Vectors</i>
--------	-------------------------------

---

**Description**

Creates a column or row vector from a numeric vector.

**Usage**

```
colVec(x)  
rowVec(x)
```

**Arguments**

`x` a numeric vector.

## Details

The functions `colVec` and `rowVec` transform a vector into a column and row vector, respectively. A column vector is a matrix object with one column, and a row vector is a matrix object with one row.

## Examples

```
## Create a numeric Vector:
x = rnorm(5)

## Column and Row Vectors:
colVec(x)
rowVec(x)
```

---

correlationTest	<i>Correlation Tests</i>
-----------------	--------------------------

---

## Description

Tests if two series are correlated.

## Usage

```
correlationTest(x, y, method = c("pearson", "kendall", "spearman"),
  title = NULL, description = NULL)
```

```
pearsonTest(x, y, title = NULL, description = NULL)
kendallTest(x, y, title = NULL, description = NULL)
spearmanTest(x, y, title = NULL, description = NULL)
```

## Arguments

<code>x, y</code>	numeric vectors of data values.
<code>method</code>	a character string naming which test should be applied.
<code>title</code>	an optional title string, if not specified the inputs data name is deparsed.
<code>description</code>	optional description string, or a vector of character strings.

## Details

The function `correlationTest` tests for association between paired samples allowing to compute Pearson's product moment correlation coefficient, Kendall's tau, or Spearman's rho.

**Value**

In contrast to R's output report from S3 objects of class "htest" a different output report is produced. The classical tests presented here return an S4 object of class "fHTEST". The object contains the following slots:

@call	the function call.
@data	the data as specified by the input argument(s).
@test	a list whose elements contain the results from the statistical test. The information provided is similar to a list object of class "htest".
@title	a character string with the name of the test. This can be overwritten specifying a user defined input argument.
@description	a character string with an optional user defined description. By default just the current date when the test was applied will be returned.

The slot @test returns an object of class "list" containing (at least) the following elements:

statistic	the value(s) of the test statistic.
p.value	the p-value(s) of the test.
parameters	a numeric value or vector of parameters.
estimate	a numeric value or vector of sample estimates.
conf.int	a numeric two row vector or matrix of 95
method	a character string indicating what type of test was performed.
data.name	a character string giving the name(s) of the data.

**Note**

Some of the test implementations are selected from R's ctest package.

**Author(s)**

R-core team for hypothesis tests implemented from R's package ctest.

**References**

- Conover, W. J. (1971); *Practical nonparametric statistics*, New York: John Wiley & Sons.  
 Lehmann E.L. (1986); *Testing Statistical Hypotheses*, John Wiley and Sons, New York.

**See Also**

[locationTest](#), [scaleTest](#), [varianceTest](#).

**Examples**

```
## x, y -
x = rnorm(50)
y = rnorm(50)

## correlationTest -
correlationTest(x, y, "pearson")
correlationTest(x, y, "kendall")
spearmanTest(x, y)
```

decor

*Decor Functions***Description**

Functions for decorating plots.

The plot utility functions are:

decor	simple decoration function,
hgrid	creates horizontal grid lines,
vgrid	creates vertical grid lines,
boxL	creates a L-shaped box,
box_	creates a bottom line box,
copyright	adds Rmetrics copyright to a plot.

**Usage**

```
decor()

hgrid(ny = NULL, ...)
vgrid(nx = NULL, ...)

boxL(col = "white")
box_(col = c("white", "black"))

copyright()
```

**Arguments**

col	the color of the background, "black" and foreground "white" lines of the box.
nx, ny	number of cells of the grid in x or y direction. When NULL, as per default, the grid aligns with the tick marks on the corresponding default axis (i.e., tickmarks as computed by axTicks).
...	additional arguments passed to the grid() function.

**Examples**

```
## Test Plot Function:
plot(x = rnorm(100), type = "l", col = "red",
     xlab = "", ylab = "Variates", las = 1)
title("Normal Deviates", adj = 0)
hgrid()
boxL()
copyright()
```

---

distCheck

*Distribution Check*


---

**Description**

Tests properties of an R implementation of a distribution, i.e. of all four of its “dpqr” functions.

**Usage**

```
distCheck(fun = "norm", n = 1000, robust = TRUE, subdivisions = 100, ...)
```

**Arguments**

fun	a character string denoting the name of the distribution.
n	an integer specifying the number of random variates to be generated.
robust	logical flag, should robust estimates be used? By default TRUE.
subdivisions	integer specifying the numbers of subdivisions in integration.
...	the distributional parameters.

**Examples**

```
distCheck("norm", mean = 1, sd = 1)

distCheck("lnorm", meanlog = 0.5, sdlog = 2, robust=FALSE)
## here, true E(X) = exp(mu + 1/2 sigma^2) = exp(.5 + 2) = exp(2.5) = 12.182
## and      Var(X) = exp(2*mu + sigma^2)*(exp(sigma^2) - 1) =      7954.67
```

---

DistributionFits

*Parameter Fit of a Distribution*


---

**Description**

A collection and description of moment and maximum likelihood estimators to fit the parameters of a distribution.

The functions are:

nFit            MLE parameter fit for a normal distribution,  
 tFit            MLE parameter fit for a Student t-distribution,  
 stableFit      MLE and Quantile Method stable parameter fit.

### Usage

```
nFit(x, doplot = TRUE, span = "auto", title = NULL, description = NULL, ...)
```

```
tFit(x, df = 4, doplot = TRUE, span = "auto", trace = FALSE, title = NULL,  

      description = NULL, ...)
```

```
stableFit(x, alpha = 1.75, beta = 0, gamma = 1, delta = 0,  

          type = c("q", "mle"), doplot = TRUE, control = list(),  

          trace = FALSE, title = NULL, description = NULL)
```

```
## S4 method for signature 'FDISTFIT'  

show(object)
```

### Arguments

control            [stableFit] -  
 a list of control parameters, see function nlminb.

alpha, beta, gamma, delta  
 [stable] -  
 The parameters are alpha, beta, gamma, and delta:  
 value of the index parameter alpha with alpha = (0, 2]; skewness parameter  
 beta, in the range [-1, 1]; scale parameter gamma; and shift parameter delta.

description        a character string which allows for a brief description.

df                  the number of degrees of freedom for the Student distribution,  $df > 2$ , maybe  
 non-integer. By default a value of 4 is assumed.

object             [show] -  
 an S4 class object as returned from the fitting functions.

doplot             a logical flag. Should a plot be displayed?

span                x-coordinates for the plot, by default 100 values automatically selected and rang-  
 ing between the 0.001, and 0.999 quantiles. Alternatively, you can specify the  
 range by an expression like `span=seq(min, max, times = n)`, where, min and  
 max are the left and right endpoints of the range, and n gives the number of the  
 intermediate points.

title              a character string which allows for a project title.

trace               a logical flag. Should the parameter estimation process be traced?

type                a character string which allows to select the method for parameter estimation:  
 "mle", the maximum log likelihood approach, or "qm", McCulloch's quantile  
 method.

x                   a numeric vector.

...                 parameters to be parsed.

## Details

### Stable Parameter Estimation:

Estimation techniques based on the quantiles of an empirical sample were first suggested by Fama and Roll [1971]. However their technique was limited to symmetric distributions and suffered from a small asymptotic bias. McCulloch [1986] developed a technique that uses five quantiles from a sample to estimate alpha and beta without asymptotic bias. Unfortunately, the estimators provided by McCulloch have restriction  $\alpha > 0.6$ .

## Value

The functions `tFit`, `hypFit` and `nigFit` return a list with the following components:

<code>estimate</code>	the point at which the maximum value of the log likelihood function is obtained.
<code>minimum</code>	the value of the estimated maximum, i.e. the value of the log likelihood function.
<code>code</code>	an integer indicating why the optimization process terminated.
<code>gradient</code>	the gradient at the estimated maximum.

Remark: The parameter estimation for the stable distribution via the maximum Log-Likelihood approach may take a quite long time.

## Examples

```
## nFit -
# Simulate random normal variates N(0.5, 2.0):
set.seed(1953)
s = rnorm(n = 1000, 0.5, 2)

## nigFit -
# Fit Parameters:
nFit(s, doplot = TRUE)
```

---

fBasics-deprecated      *Deprecated Functions in Package fBasics*

---

## Description

These functions are provided for compatibility with older versions of the package only, and may be defunct as soon as of the next release.

There are none currently. `dstable` etc now are defunct, as they have been available from **stabledist** since early 2011.

## See Also

[Deprecated, Defunct](#)

fBasicsData

*fBasics Data Sets***Description**

The following data sets are part of this package:

Capitalization	Market capitalization of domestic companies,
cars2	Data for various car models,
DowJones30	Down Jones 30 stocks,
HedgeFund	Hennessee Hedge Fund Indices,
msft.dat	Daily Microsoft OHLC prices and volume,
nyse	NYSE composite Index,
PensionFund	Swiss Pension Fund LPP-2005,
swissEconomy	Swiss Economic Data,
SWXLP	Swiss Pension Fund LPP-2000,
usdthb	Tick data of USD to THB.

**Details****Capitalization:**

Capitalization contains market capitalization of domestic companies from 2003 - 2008 in USD millions.

**cars2:**

cars2 contains the price, country, reliability, mileage, type, weight, engine displacement and net horsepower of various car models.

**DowJones30:**

DowJones30 contains daily observations from the Dow Jones 30 Index series. Each of the thirty columns represents the closing price of a stock in the Index.

**HedgeFund:**

HedgeFund contains monthly percentual returns of various hedge fund strategies from Hennessee Group LLC.

**msft.dat:**

msft.dat contains daily prices (open, high, low and close) and volumes for the Microsoft stocks.

**nyse:**

nyse contains daily records of the NYSE Composite Index.

**PensionFund:**

PensionFund is a daily data set of the Swiss pension fund benchmark LPP-2005. The data set ranges from 2005-11-01 to 2007-04-11. The columns are named: SBI, SPI, SII, LMI, MPI, ALT, LPP25, LPP40, LPP60

**swissEconomy:**

swissEconomy contains the GDP per capita, exports, imports, interest rates, inflation, unemployment and population for the years 1964 - 1999 for Switzerland.



**SWXLP:**

SWXLP is a daily data set of the Swiss pension fund benchmark LPP-2000. The data set ranges from 2000-01-03 to 2007-05-08. The columns are named: SBI, SPI, SII, LP25, LP40, LP60.

**usdthb:**

usdthb Tick data of US Dollar (USD) in Thailand Bhat (THB) collected from Reuters. Format: YYYYMMDDhhmm. Column variables: delay time, contributor, bid and ask prices, and quality flag. It covers the Asia FX crisis in June 1997.

**References****Capitalization:**

*World Federation of Stock Exchanges*, <http://www.world-exchanges.org/statistics>.

**cars2:**

Derived from the car90 dataset within the rpart package. The car90 dataset is based on the car.all dataset in S-PLUS. Original data comes from: April 1990, *Consumer Reports Magazine*, pages 235-255, 281-285 and 287-288.

**DowJones30**

<http://www.yahoo.com>.

**HedgeFund:**

<http://www.hennessiegroup.com/indices/returns/year/2005.html>.

**msft.dat:**

<http://www.yahoo.com>.

**nyse:**

<http://www.nyse.com>.

**PensionFund:**

SBI, SPI, SII: SIX (Swiss Exchange Zurich); LPP25, LPP40, LPP60: Banque Pictet Geneva; LMI, MPI, ALT: Recalculated from the indices and benchmarks

**swissEconomy:**

<http://www.oecd.org/> and <http://www.imf.org/>.

**SWXLP:**

SBI, SPI, SII: SIX (Swiss Exchange Zurich); LPP25, LPP40, LPP60: Banque Pictet Geneva

**usdthb:**

Reuters Select Feed Terminal (1997).

**Examples**

```
## Plot DowJones30 Example Data Set
series <- as.timeSeries(DowJones30)
head(series)
plot(series[,1:6], type = "l")
```

fHTEST

*Tests Class Representation and Utilities***Description**

Class representation, methods and utility functions for objects of class 'fHTEST'.

The class representation and methods are:

fHTEST	Representation for an S4 object of class "fHTEST",
show	S4 print method.

**Usage**

```
## S4 method for signature 'fHTEST'
show(object)
```

**Arguments**

object	[show] - an S4 object of class "fHTEST".
--------	---

**Value**

In contrast to R's output report from S3 objects of class "htest" a different output report is produced. The tests return an S4 object of class "fHTEST". The object contains the following slots:

@call	the function call.
@data	the data as specified by the input argument(s).
@test	a list whose elements contain the results from the statistical test. The information provided is similar to a list object of class "htest".
@title	a character string with the name of the test. This can be overwritten specifying a user defined input argument.
@description	a character string with an optional user defined description. By default just the current date when the test was applied will be returned.

The slot @test returns an object of class "list" containing the following elements:

statistic	the value(s) of the test statistic.
p.value	the p-value(s) of the test.
parameters	a numeric value or vector of parameters.
estimate	a numeric value or vector of sample estimates.
conf.int	a numeric two row vector or matrix of 95
method	a character string indicating what type of test was performed.
data.name	a character string giving the name(s) of the data.

**Examples**

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

---

 getS4

*General S4 Class Extractor Functions*


---

**Description**

A collection and description of functions to extract slots from S4 class objects.

The extractor functions are:

getModel	Extracts the model slot from a S4 object,
getTitle	Extracts the title slot from a S4 object,
getDescription	Extracts the description slot from a S4 object,
getSlot	Extracts a specified slot from a S4 object,
getArgs	Shows the arguments of a S4 function.

Since R version 2.14.0, a generic `getCall()` is part of R; for earlier versions, we had provided a simple version for S4 objects.

**Usage**

```
getModel(object)
getTitle(object)
getDescription(object)

getSlot(object, slotName)

getArgs(f, signature)
```

**Arguments**

f	a generic function or the character-string name of one.
object	an object of class S4.
signature	the signature of classes to match to the arguments of f
slotName	a character string, the name of the slot to be extracted from the S4 object.

**Value**

```
getModel
getTitle
```

```
getDescription
getSlot
return the content of the slot.

getArgs returns the names of the arguments.
```

### Examples

```
## Example S4 Representation:
# Hypothesis Testing with Control Settings
setClass("hypTest",
  representation(
    call = "call",
    data = "numeric",
    test = "list",
    description = "character")
)

## Shapiro Wilk Normality Test
swTest = function(x, description = "") {
  ans = shapiro.test(x)
  class(ans) = "list"
  new("hypTest",
    call = match.call(),
    data = x,
    test = ans,
    description = description)
}
test = swTest(x = rnorm(500), description = "500 RVs")

## Extractor Functions:
isS4(test)
getCall(test)
getDescription(test)

## get arguments
args(returns)
getArgs(returns)
getArgs("returns")
getArgs(returns, "timeSeries")
getArgs("returns", "timeSeries")
```

### Description

Calculates moments of the generalized hyperbolic distribution function.

**Usage**

```
dgh(x, alpha = 1, beta = 0, delta = 1, mu = 0, lambda = -1/2, log = FALSE)
pgh(q, alpha = 1, beta = 0, delta = 1, mu = 0, lambda = -1/2)
qgh(p, alpha = 1, beta = 0, delta = 1, mu = 0, lambda = -1/2)
rgh(n, alpha = 1, beta = 0, delta = 1, mu = 0, lambda = -1/2)
```

**Arguments**

alpha, beta, delta, mu, lambda	numeric values. alpha is the first shape parameter; beta is the second shape parameter in the range $(0, \alpha)$ ; delta is the scale parameter, must be zero or positive; mu is the location parameter, by default 0; and lambda defines the subclass, by default -1/2. These are the meanings of the parameters in the first parameterization pm=1 which is the default parameterization. In the second parameterization, pm=2 alpha and beta take the meaning of the shape parameters (usually named) zeta and rho. In the third parameterization, pm=3 alpha and beta take the meaning of the shape parameters (usually named) xi and chi. In the fourth parameterization, pm=4 alpha and beta take the meaning of the shape parameters (usually named) a.bar and b.bar.
log	a logical flag by default FALSE. Should labels and a main title drawn to the plot?
n	number of observations.
p	a numeric vector of probabilities.
x, q	a numeric vector of quantiles.

**Details**

The generator rgh is based on the GH algorithm given by Scott (2004).

**Value**

All values for the \*gh functions are numeric vectors: d\* returns the density, p\* returns the distribution function, q\* returns the quantile function, and r\* generates random deviates.

All values have attributes named "param" listing the values of the distributional parameters.

**Author(s)**

David Scott for code implemented from R's contributed package HyperbolicDist.

**References**

- Atkinson, A.C. (1982); *The simulation of generalized inverse Gaussian and hyperbolic random variables*, SIAM J. Sci. Stat. Comput. 3, 502–515.
- Barndorff-Nielsen O. (1977); *Exponentially decreasing distributions for the logarithm of particle size*, Proc. Roy. Soc. Lond., A353, 401–419.
- Barndorff-Nielsen O., Blaesild, P. (1983); *Hyperbolic distributions*. In *Encyclopedia of Statistical Sciences*, Eds., Johnson N.L., Kotz S. and Read C.B., Vol. 3, pp. 700–707. New York: Wiley.
- Raible S. (2000); *Levy Processes in Finance: Theory, Numerics and Empirical Facts*, PhD Thesis, University of Freiburg, Germany, 161 pages.

**Examples**

```
## rgh -
set.seed(1953)
r = rgh(5000, alpha = 1, beta = 0.3, delta = 1)
plot(r, type = "l", col = "steelblue",
     main = "gh: alpha=1 beta=0.3 delta=1")

## dgh -
# Plot empirical density and compare with true density:
hist(r, n = 25, probability = TRUE, border = "white", col = "steelblue")
x = seq(-5, 5, 0.25)
lines(x, dgh(x, alpha = 1, beta = 0.3, delta = 1))

## pgh -
# Plot df and compare with true df:
plot(sort(r), (1:5000/5000), main = "Probability", col = "steelblue")
lines(x, pgh(x, alpha = 1, beta = 0.3, delta = 1))

## qgh -
# Compute Quantiles:
qgh(pgh(seq(-5, 5, 1), alpha = 1, beta = 0.3, delta = 1),
     alpha = 1, beta = 0.3, delta = 1)
```

---

ghFit

*GH Distribution Fit*


---

**Description**

Estimates the distributional parameters for a generalized hyperbolic distribution.

**Usage**

```
ghFit(x, alpha = 1, beta = 0, delta = 1, mu = 0, lambda = -1/2,
      scale = TRUE, doplot = TRUE, span = "auto", trace = TRUE,
      title = NULL, description = NULL, ...)
```

**Arguments**

**x** a numeric vector.

**alpha, beta, delta, mu, lambda** The parameters are alpha, beta, delta, mu, and lambda: shape parameter alpha; skewness parameter beta,  $\text{abs}(\text{beta})$  is in the range  $(0, \alpha)$ ; scale parameter delta, delta must be zero or positive; location parameter mu, by default 0; and lambda parameter lambda, by default  $-1/2$ .

**scale** a logical flag, by default TRUE. Should the time series be scaled by its standard deviation to achieve a more stable optimization?

**doplot** a logical flag. Should a plot be displayed?

span	x-coordinates for the plot, by default 100 values automatically selected and ranging between the 0.001, and 0.999 quantiles. Alternatively, you can specify the range by an expression like <code>span=seq(min, max, times = n)</code> , where, <code>min</code> and <code>max</code> are the left and right endpoints of the range, and <code>n</code> gives the number of the intermediate points.
trace	a logical flag. Should the parameter estimation process be traced?
title	a character string which allows for a project title.
description	a character string which allows for a brief description.
...	parameters to be parsed.

### Details

The function `nlm` is used to minimize the "negative" maximum log-likelihood function. `nlm` carries out a minimization using a Newton-type algorithm.

### Value

returns a list with the following components:

estimate	the point at which the maximum value of the log likelihood function is obtained.
minimum	the value of the estimated maximum, i.e. the value of the log likelihood function.
code	an integer indicating why the optimization process terminated. 1: relative gradient is close to zero, current iterate is probably solution; 2: successive iterates within tolerance, current iterate is probably solution; 3: last global step failed to locate a point lower than estimate. Either estimate is an approximate local minimum of the function or <code>steptol</code> is too small; 4: iteration limit exceeded; 5: maximum step size <code>stepmax</code> exceeded five consecutive times. Either the function is unbounded below, becomes asymptotic to a finite value from above in some direction or <code>stepmax</code> is too small.
gradient	the gradient at the estimated maximum.
steps	number of function calls.

### Examples

```
## ghFit -
# Simulate Random Variates:
set.seed(1953)
s = rgh(n = 1000, alpha = 1.5, beta = 0.3, delta = 0.5, mu = -1.0)

## ghFit -
# Fit Parameters:
ghFit(s, alpha = 1, beta = 0, delta = 1, mu = mean(s), doplot = TRUE)
```

---

 ghMode

*Generalized Hyperbolic Mode*


---

**Description**

Computes the mode of the generalized hyperbolic function.

**Usage**

```
ghMode(alpha = 1, beta = 0, delta = 1, mu = 0, lambda = -1/2)
```

**Arguments**

alpha, beta, delta, mu, lambda

shape parameter alpha; skewness parameter beta,  $\text{abs}(\text{beta})$  is in the range  $(0, \text{alpha})$ ; scale parameter delta, delta must be zero or positive; location parameter mu, by default 0. These is the meaning of the parameters in the first parameterization pm=1 which is the default parameterization selection. In the second parameterization, pm=2 alpha and beta take the meaning of the shape parameters (usually named) zeta and rho. In the third parameterization, pm=3 alpha and beta take the meaning of the shape parameters (usually named) xi and chi. In the fourth parameterization, pm=4 alpha and beta take the meaning of the shape parameters (usually named)  $\bar{a}$  and  $\bar{b}$ .

**Value**

returns the mode for the generalized hyperbolic distribution. A numeric value.

**References**

Atkinson, A.C. (1982); *The simulation of generalized inverse Gaussian and hyperbolic random variables*, SIAM J. Sci. Stat. Comput. 3, 502–515.

Barndorff-Nielsen O. (1977); *Exponentially decreasing distributions for the logarithm of particle size*, Proc. Roy. Soc. Lond., A353, 401–419.

Barndorff-Nielsen O., Blaesild, P. (1983); *Hyperbolic distributions*. In *Encyclopedia of Statistical Sciences*, Eds., Johnson N.L., Kotz S. and Read C.B., Vol. 3, pp. 700–707. New York: Wiley.

Raible S. (2000); *Levy Processes in Finance: Theory, Numerics and Empirical Facts*, PhD Thesis, University of Freiburg, Germany, 161 pages.

**Examples**

```
## ghMode -
  ghMode()
```



**Description**

Calculates moments of the generalized hyperbolic distribution function

**Usage**

```
ghMean(alpha=1, beta=0, delta=1, mu=0, lambda=-1/2)
ghVar(alpha=1, beta=0, delta=1, mu=0, lambda=-1/2)
ghSkew(alpha=1, beta=0, delta=1, mu=0, lambda=-1/2)
ghKurt(alpha=1, beta=0, delta=1, mu=0, lambda=-1/2)

ghMoments(order, type = c("raw", "central", "mu"),
           alpha = 1, beta=0, delta=1, mu=0, lambda=-1/2)
```

**Arguments**

alpha, beta, delta, mu, lambda  
 numeric values. alpha is the first shape parameter; beta is the second shape parameter in the range  $(0, \alpha)$ ; delta is the scale parameter, must be zero or positive; mu is the location parameter, by default 0; and lambda defines the subclass, by default -1/2.

order  
 an integer value, the order of the moment.

type  
 a character value, "raw" returns the moments about zero, "central" returns the central moments about the mean, and "mu" returns the moments about the location parameter mu.

**Value**

a numerical value.

**Author(s)**

Diethelm Wuertz.

**References**

Scott, D. J., Wuertz, D. and Tran, T. T. (2008) *Moments of the Generalized Hyperbolic Distribution*. Preprint.

## Examples

```
## ghMean -
  ghMean(alpha=1.1, beta=0.1, delta=0.8, mu=-0.3, lambda=1)

## ghKurt -
  ghKurt(alpha=1.1, beta=0.1, delta=0.8, mu=-0.3, lambda=1)

## ghMoments -
  ghMoments(4,
    alpha=1.1, beta=0.1, delta=0.8, mu=-0.3, lambda=1)
  ghMoments(4, "central",
    alpha=1.1, beta=0.1, delta=0.8, mu=-0.3, lambda=1)
```

---

 ghRobMoments

*Robust Moments for the GH*


---

## Description

Computes the first four robust moments for the generalized hyperbolic distribution..

## Usage

```
ghMED(alpha = 1, beta = 0, delta = 1, mu = 0, lambda = -1/2)
ghIQR(alpha= 1, beta = 0, delta = 1, mu = 0, lambda = -1/2)
ghSKEW(alpha = 1, beta = 0, delta = 1, mu = 0, lambda = -1/2)
ghKURT(alpha = 1, beta = 0, delta = 1, mu = 0, lambda = -1/2)
```

## Arguments

alpha, beta, delta, mu, lambda

numeric values. alpha is the first shape parameter; beta is the second shape parameter in the range (0, alpha); delta is the scale parameter, must be zero or positive; mu is the location parameter, by default 0; and lambda defines the subclass, by default -1/2. These are the meanings of the parameters in the first parameterization pm=1 which is the default parameterization. In the second parameterization, pm=2 alpha and beta take the meaning of the shape parameters (usually named) zeta and rho. In the third parameterization, pm=3 alpha and beta take the meaning of the shape parameters (usually named) xi and chi. In the fourth parameterization, pm=4 alpha and beta take the meaning of the shape parameters (usually named) a.bar and b.bar.

## Value

All values for the \*gh functions are numeric vectors: d\* returns the density, p\* returns the distribution function, q\* returns the quantile function, and r\* generates random deviates.

All values have attributes named "param" listing the values of the distributional parameters.

**Author(s)**

Diethelm Wuertz.

**Examples**

```
## ghMED -  
# Median:  
ghMED(alpha = 1, beta = 0, delta = 1, mu = 0, lambda = -1/2)  
  
## ghIQR -  
# Inter-quartile Range:  
ghIQR(alpha = 1, beta = 0, delta = 1, mu = 0, lambda = -1/2)  
  
## ghSKEW -  
# Robust Skewness:  
ghSKEW(alpha = 1, beta = 0, delta = 1, mu = 0, lambda = -1/2)  
  
## ghKURT -  
# Robust Kurtosis:  
ghKURT(alpha = 1, beta = 0, delta = 1, mu = 0, lambda = -1/2)
```

---

ghSlider

*Generalized Hyperbolic Distribution Slider*

---

**Description**

Displays interactively the dependence of the generalized hyperbolic distribution on its parameters.

**Usage**

```
ghSlider()
```

**Value**

a tcl/tk based graphical user interface.

This is a nice display for educational purposes to investigate the densities and probabilities of the generalized hyperbolic distribution.

**Examples**

```
## ghSlider -  
# ghSlider()
```

ght

*Generalized Hyperbolic Student-t***Description**

Density, distribution function, quantile function and random generation for the hyperbolic distribution.

**Usage**

```
dght(x, beta = 0.1, delta = 1, mu = 0, nu = 10, log = FALSE)
pght(q, beta = 0.1, delta = 1, mu = 0, nu = 10)
qght(p, beta = 0.1, delta = 1, mu = 0, nu = 10)
rght(n, beta = 0.1, delta = 1, mu = 0, nu = 10)
```

**Arguments**

beta, delta, mu numeric values. beta is the skewness parameter in the range  $(0, \alpha)$ ; delta is the scale parameter, must be zero or positive; mu is the location parameter, by default 0. These are the parameters in the first parameterization.

nu a numeric value, the number of degrees of freedom. Note, alpha takes the limit of  $\text{abs}(\text{beta})$ , and  $\text{lambda} = -\text{nu}/2$ .

x, q a numeric vector of quantiles.

p a numeric vector of probabilities.

n number of observations.

log a logical, if TRUE, probabilities p are given as  $\log(p)$ .

**Value**

All values for the \*ght functions are numeric vectors: d\* returns the density, p\* returns the distribution function, q\* returns the quantile function, and r\* generates random deviates.

All values have attributes named "param" listing the values of the distributional parameters.

**References**

- Atkinson, A.C. (1982); *The simulation of generalized inverse Gaussian and hyperbolic random variables*, SIAM J. Sci. Stat. Comput. 3, 502–515.
- Barndorff-Nielsen O. (1977); *Exponentially decreasing distributions for the logarithm of particle size*, Proc. Roy. Soc. Lond., A353, 401–419.
- Barndorff-Nielsen O., Blaesild, P. (1983); *Hyperbolic distributions*. In *Encyclopedia of Statistical Sciences*, Eds., Johnson N.L., Kotz S. and Read C.B., Vol. 3, pp. 700–707. New York: Wiley.
- Raible S. (2000); *Levy Processes in Finance: Theory, Numerics and Empirical Facts*, PhD Thesis, University of Freiburg, Germany, 161 pages.

**Examples**

```
## ght -
#
```

---

 ghtFit

*GHT Distribution Fit*


---

**Description**

Estimates the distributional parameters for a generalized hyperbolic Student-t distribution.

**Usage**

```
ghtFit(x, beta = 0.1, delta = 1, mu = 0, nu = 10,
       scale = TRUE, doplot = TRUE, span = "auto", trace = TRUE,
       title = NULL, description = NULL, ...)
```

**Arguments**

beta, delta, mu	numeric values. beta is the skewness parameter in the range $(0, \alpha)$ ; delta is the scale parameter, must be zero or positive; mu is the location parameter, by default 0. These are the parameters in the first parameterization.
nu	defines the number of degrees of freedom. Note, alpha takes the limit of $\text{abs}(\text{beta})$ , and $\text{lambda} = -\text{nu}/2$ .
x	a numeric vector.
scale	a logical flag, by default TRUE. Should the time series be scaled by its standard deviation to achieve a more stable optimization?
doplot	a logical flag. Should a plot be displayed?
span	x-coordinates for the plot, by default 100 values automatically selected and ranging between the 0.001, and 0.999 quantiles. Alternatively, you can specify the range by an expression like <code>span=seq(min, max, times = n)</code> , where, min and max are the left and right endpoints of the range, and n gives the number of the intermediate points.
trace	a logical flag. Should the parameter estimation process be traced?
title	a character string which allows for a project title.
description	a character string which allows for a brief description.
...	parameters to be parsed.

**Details**

The function `nlm` is used to minimize the "negative" maximum log-likelihood function. `nlm` carries out a minimization using a Newton-type algorithm.

**Value**

returns a list with the following components:

estimate	the point at which the maximum value of the log likelihood function is obtained.
minimum	the value of the estimated maximum, i.e. the value of the log likelihood function.
code	an integer indicating why the optimization process terminated. 1: relative gradient is close to zero, current iterate is probably solution; 2: successive iterates within tolerance, current iterate is probably solution; 3: last global step failed to locate a point lower than estimate. Either estimate is an approximate local minimum of the function or steptol is too small; 4: iteration limit exceeded; 5: maximum step size stepmax exceeded five consecutive times. Either the function is unbounded below, becomes asymptotic to a finite value from above in some direction or stepmax is too small.
gradient	the gradient at the estimated maximum.
steps	number of function calls.

**Examples**

```
## ghtFit -
# Simulate Random Variates:
set.seed(1953)

## ghtFit -
# Fit Parameters:
```

---

ghtMode	<i>Generalized Hyperbolic Student-t Mode</i>
---------	--

---

**Description**

Computes the mode of the generalized hyperbolic Student-t distribution.

**Usage**

```
ghtMode(beta = 0.1, delta = 1, mu = 0, nu = 10)
```

**Arguments**

beta, delta, mu	numeric values. beta is the skewness parameter in the range (0, alpha); delta is the scale parameter, must be zero or positive; mu is the location parameter, by default 0. These are the parameters in the first parameterization.
nu	a numeric value, the number of degrees of freedom. Note, alpha takes the limit of abs(beta), and lambda=-nu/2.

**Value**

returns the mode for the generalized hyperbolic Student-t distribution. A numeric value.

**References**

- Atkinson, A.C. (1982); *The simulation of generalized inverse Gaussian and hyperbolic random variables*, SIAM J. Sci. Stat. Comput. 3, 502–515.
- Barndorff-Nielsen O. (1977); *Exponentially decreasing distributions for the logarithm of particle size*, Proc. Roy. Soc. Lond., A353, 401–419.
- Barndorff-Nielsen O., Blaesild, P. (1983); *Hyperbolic distributions*. In *Encyclopedia of Statistical Sciences*, Eds., Johnson N.L., Kotz S. and Read C.B., Vol. 3, pp. 700–707. New York: Wiley.
- Raible S. (2000); *Levy Processes in Finance: Theory, Numerics and Empirical Facts*, PhD Thesis, University of Freiburg, Germany, 161 pages.

**Examples**

```
## ghtMode -
  ghtMode()
```

---

 ghtMoments

*Generalized Hyperbolic Student-t Moments*


---

**Description**

Calculates moments of the generalized hyperbolic Student-t distribution function.

**Usage**

```
ghtMean(beta=0.1, delta=1, mu=0, nu=10)
ghtVar(beta=0.1, delta=1, mu=0, nu=10)
ghtSkew(beta=0.1, delta=1, mu=0, nu=10)
ghtKurt(beta=0.1, delta=1, mu=0, nu=10)

ghtMoments(order, type = c("raw", "central", "mu"),
  beta=0.1, delta=1, mu=0, nu=10)
```

**Arguments**

- beta, delta, mu numeric values. beta is the skewness parameter in the range (0, alpha); delta is the scale parameter, must be zero or positive; mu is the location parameter, by default 0. These are the parameters in the first parameterization.
- nu a numeric value, the number of degrees of freedom. Note, alpha takes the limit of abs(beta), and lambda=-nu/2.
- order an integer value, the order of the moment.
- type a character value, "raw" returns the moments about zero, "central" returns the central moments about the mean, and "mu" returns the moments about the location parameter mu.

**Value**

a numerical value.

**Author(s)**

Diethelm Wuertz.

**References**

Scott, D.J., Wuertz, D. and Tran, T.T. (2008) *Moments of the Generalized Hyperbolic Distribution*. Preprint.

**Examples**

```
## ghtMean -
  ghtMean(beta=0.2, delta=1.2, mu=-0.5, nu=4)

## ghtKurt -
  ghtKurt(beta=0.2, delta=1.2, mu=-0.5, nu=4)

## ghtMoments -
  ghtMoments(4,
    beta=0.2, delta=1.2, mu=-0.5, nu=4)
  ghtMoments(4, "central",
    beta=0.2, delta=1.2, mu=-0.5, nu=4)
```

---

 ghtRobMoments

*Robust Moments for the GHT*


---

**Description**

Computes the first four robust moments for the generalized hyperbolic Student-t.

**Usage**

```
ghtMED(beta = 0.1, delta = 1, mu = 0, nu = 10)
ghtIQR(beta = 0.1, delta = 1, mu = 0, nu = 10)
ghtSKEW(beta = 0.1, delta = 1, mu = 0, nu = 10)
ghtKURT(beta = 0.1, delta = 1, mu = 0, nu = 10)
```

**Arguments**

beta, delta, mu numeric values. beta is the skewness parameter in the range  $(0, \alpha)$ ; delta is the scale parameter, must be zero or positive; mu is the location parameter, by default 0. These are the parameters in the first parameterization.

nu a numeric value, the number of degrees of freedom. Note, alpha takes the limit of  $\text{abs}(\beta)$ , and  $\lambda = -\nu/2$ .



**Value**

All values for the \*ght functions are numeric vectors: d\* returns the density, p\* returns the distribution function, q\* returns the quantile function, and r\* generates random deviates.

All values have attributes named "param" listing the values of the distributional parameters.

**Author(s)**

Diethelm Wuertz.

**Examples**

```
## ghtMED -  
# Median:  
ghtMED(beta = 0.1, delta = 1, mu = 0, nu = 10)  
  
## ghtIQR -  
# Inter-quartile Range:  
ghtIQR(beta = 0.1, delta = 1, mu = 0, nu = 10)  
  
## ghtSKEW -  
# Robust Skewness:  
ghtSKEW(beta = 0.1, delta = 1, mu = 0, nu = 10)  
  
## ghtKURT -  
# Robust Kurtosis:  
ghtKURT(beta = 0.1, delta = 1, mu = 0, nu = 10)
```

---

gld

*Generalized Lambda Distribution*

---

**Description**

Density, distribution function, quantile function and random generation for the generalized lambda distribution.

**Usage**

```
dgld(x, lambda1 = 0, lambda2 = -1, lambda3 = -1/8, lambda4 = -1/8, log = FALSE)  
pgld(q, lambda1 = 0, lambda2 = -1, lambda3 = -1/8, lambda4 = -1/8)  
qgld(p, lambda1 = 0, lambda2 = -1, lambda3 = -1/8, lambda4 = -1/8)  
rgld(n, lambda1 = 0, lambda2 = -1, lambda3 = -1/8, lambda4 = -1/8)
```

**Arguments**

lambda1, lambda2, lambda3, lambda4  
 are numeric values where lambda1 is the location parameter, lambda2 is the location parameter, lambda3 is the first shape parameter, and lambda4 is the second shape parameter.

n  
 number of observations.

p  
 a numeric vector of probabilities.

x, q  
 a numeric vector of quantiles.

log  
 a logical, if TRUE, probabilities p are given as log(p).

**Value**

All values for the \*gld functions are numeric vectors: d\* returns the density, p\* returns the distribution function, q\* returns the quantile function, and r\* generates random deviates.

All values have attributes named "param" listing the values of the distributional parameters.

**Author(s)**

Chong Gu for code implemented from R's contributed package gld.

**Examples**

```
## rgld -
set.seed(1953)
r = rgld(500,
  lambda1=0, lambda2=-1, lambda3=-1/8, lambda4=-1/8)
plot(r, type = "l", col = "steelblue",
  main = "gld: lambda1=0 lambda2=-1 lambda3/4=-1/8")

## dgld -
# Plot empirical density and compare with true density:
hist(r, n = 25, probability = TRUE, border = "white",
  col = "steelblue")
x = seq(-5, 5, 0.25)
lines(x, dgld(x,
  lambda1=0, lambda2=-1, lambda3=-1/8, lambda4=-1/8))

## pgld -
# Plot df and compare with true df:
plot(sort(r), ((1:500)-0.5)/500, main = "Probability",
  col = "steelblue")
lines(x, pgld(x,
  lambda1=0, lambda2=-1, lambda3=-1/8, lambda4=-1/8))

## qgld -
# Compute Quantiles:
qgld(pgld(seq(-5, 5, 1),
  lambda1=0, lambda2=-1, lambda3=-1/8, lambda4=-1/8),
  lambda1=0, lambda2=-1, lambda3=-1/8, lambda4=-1/8)
```

---

gldFit	<i>GH Distribution Fit</i>
--------	----------------------------

---

**Description**

Estimates the distributional parameters for a generalized lambda distribution.

**Usage**

```
gldFit(x, lambda1 = 0, lambda2 = -1, lambda3 = -1/8, lambda4 = -1/8,
       method = c("mle", "mps", "gof", "hist", "rob"),
       scale = NA, doplot = TRUE, add = FALSE, span = "auto", trace = TRUE,
       title = NULL, description = NULL, ...)
```

**Arguments**

x	a numeric vector.
lambda1, lambda2, lambda3, lambda4	are numeric values where lambda1 is the location parameter, lambda2 is the location parameter, lambda3 is the first shape parameter, and lambda4 is the second shape parameter.
method	a character string, the estimation approach to fit the distributional parameters, see details.
scale	not used.
doplot	a logical flag. Should a plot be displayed?
add	a logical flag. Should a new fit added to an existing plot?
span	x-coordinates for the plot, by default 100 values automatically selected and ranging between the 0.001, and 0.999 quantiles. Alternatively, you can specify the range by an expression like <code>span=seq(min, max, times = n)</code> , where, <code>min</code> and <code>max</code> are the left and right endpoints of the range, and <code>n</code> gives the number of the intermediate points.
trace	a logical flag. Should the parameter estimation process be traced?
title	a character string which allows for a project title.
description	a character string which allows for a brief description.
...	parameters to be parsed.

**Details**

The function `nlminb` is used to minimize the objective function. The following approaches have been implemented:

"mle", maximum log likelihood estimation.

"mps", maximum product spacing estimation.

"gof", goodness of fit approaches, `type="ad"` Anderson-Darling, `type="cvm"` Cramer-vonMise, `type="ks"` Kolmogorov-Smirnov.

"hist", histogram binning approaches, "fd" Freedman-Diaconis binning, "scott", Scott histogram binning, "sturges", Sturges histogram binning.

"rob", robust moment matching.

### Value

returns a list with the following components:

estimate	the point at which the maximum value of the log likelihood function is obtained.
minimum	the value of the estimated maximum, i.e. the value of the log likelihood function.
code	an integer indicating why the optimization process terminated. 1: relative gradient is close to zero, current iterate is probably solution; 2: successive iterates within tolerance, current iterate is probably solution; 3: last global step failed to locate a point lower than estimate. Either estimate is an approximate local minimum of the function or <code>steptol</code> is too small; 4: iteration limit exceeded; 5: maximum step size <code>stepmax</code> exceeded five consecutive times. Either the function is unbounded below, becomes asymptotic to a finite value from above in some direction or <code>stepmax</code> is too small.
gradient	the gradient at the estimated maximum.
steps	number of function calls.

### Examples

```
## gldFit -
# Simulate Random Variates:
set.seed(1953)
s = rgld(n = 1000, lambda1=0, lambda2=-1, lambda3=-1/8, lambda4=-1/8)

## gldFit -
# Fit Parameters:
gldFit(s, lambda1=0, lambda2=-1, lambda3=-1/8, lambda4=-1/8,
      doplot = TRUE, trace = TRUE)
```

---

gldMode

*Generalized Lambda Distribution Mode*

---

### Description

Computes the mode of the generalized lambda distribution.

### Usage

```
gldMode(lambda1 = 0, lambda2 = -1, lambda3 = -1/8, lambda4 = -1/8)
```

**Arguments**

lambda1, lambda2, lambda3, lambda4

are numeric values where lambda1 is the location parameter, lambda2 is the location parameter, lambda3 is the first shape parameter, and lambda4 is the second shape parameter.

**Author(s)**

Implemented by Diethelm Wuertz

---

gldRobMoments

*Robust Moments for the GLD*

---

**Description**

Computes the first four robust moments for the Generalized Lambda Distribution.

**Usage**

```
gldMED(lambda1 = 0, lambda2 = -1, lambda3 = -1/8, lambda4 = -1/8)
gldIQR(lambda1 = 0, lambda2 = -1, lambda3 = -1/8, lambda4 = -1/8)
gldSKEW(lambda1 = 0, lambda2 = -1, lambda3 = -1/8, lambda4 = -1/8)
gldKURT(lambda1 = 0, lambda2 = -1, lambda3 = -1/8, lambda4 = -1/8)
```

**Arguments**

lambda1, lambda2, lambda3, lambda4

are numeric values where lambda1 is the location parameter, lambda2 is the location parameter, lambda3 is the first shape parameter, and lambda4 is the second shape parameter.

**Value**

All values for the \*gld functions are numeric vectors: d\* returns the density, p\* returns the distribution function, q\* returns the quantile function, and r\* generates random deviates.

All values have attributes named "param" listing the values of the distributional parameters.

**Author(s)**

Diethelm Wuertz.

**Examples**

```
## gldMED -
# Median:
gldMED(lambda1 = 0, lambda2 = -1, lambda3 = -1/8, lambda4 = -1/8)

## gldIQR -
# Inter-quartile Range:
gldIQR(lambda1 = 0, lambda2 = -1, lambda3 = -1/8, lambda4 = -1/8)

## gldSKEW -
# Robust Skewness:
gldSKEW(lambda1 = 0, lambda2 = -1, lambda3 = -1/8, lambda4 = -1/8)

## gldKURT -
# Robust Kurtosis:
gldKURT(lambda1 = 0, lambda2 = -1, lambda3 = -1/8, lambda4 = -1/8)
```

---

gridVector

*Grid Vector Coordinates*


---

**Description**

Creates from two vectors rectangular grid coordinates..

**Usage**

```
gridVector(x, y = NULL)
```

**Arguments**

*x*, *y* two numeric vectors of length *m* and *n* which span the rectangular grid of size *m* times *n*. If *y* takes the default value, NULL, then *y=x*.

**Value**

returns a list with two entries named \$X and \$Y, giving the coordinates which span the bivariate grid.

**See Also**

[expand.grid](#).

**Examples**

```
## gridVector -
gridVector((0:10)/10)
gridVector((0:10)/10, (0:10)/10)
```

**Description**

Functions which compute the Heaviside and related functions. These include the sign function, the delta function, the boxcar function, and the ramp function.

The functions are:

Heaviside	Computes Heaviside unit step function,
Sign	Just another signum function,
Delta	Computes delta function,
Boxcar	Computes boxcar function,
Ramp	Computes ramp function.

**Usage**

```
Heaviside(x, a = 0)
Sign(x, a = 0)
Delta(x, a = 0)
Boxcar(x, a = 0.5)
Ramp(x, a = 0)
```

**Arguments**

a	a numeric value, the location of the break.
x	a numeric vector.

**Details**

The Heaviside step function `Heaviside` is 1 for  $x > a$ , 1/2 for  $x = a$ , and 0 for  $x < a$ .

The Sign function `Sign` is 1 for  $x > a$ , 0 for  $x = a$ , and -1 for  $x < a$ .

The delta function `Delta` is defined as:  $\text{Delta}(x) = d/dx H(x-a)$ .

The boxcar function `Boxcar` is defined as:  $\text{Boxcar}(x) = H(x+a) - H(x-a)$ .

The ramp function is defined as:  $\text{Ramp}(x) = (x-a) * H(x-a)$ .

**Value**

returns the function values of the selected function.

**Note**

The Heaviside function is used in the implementation of the skew Normal, Student-t, and Generalized Error distributions, distributions functions which play an important role in modelling GARCH processes.

**References**

Weisstein W. (2004); <http://mathworld.wolfram.com/HeavisideStepFunction.html>, Mathworld.

**See Also**

GarchDistribution, GarchDistributionFits.

**Examples**

```
## Heaviside -
x = sort(round(c(-1, -0.5, 0, 0.5, 1, 5*rnorm(5)), 2))
h = Heaviside(x)

## Sign -
s = Sign(x)

## Delta -
d = Delta(x)

## Boxcar -
Pi = Boxcar(x)

## Ramp -
r = Ramp(x)
cbind(x = x, Step = h, Signum = s, Delta = d, Pi = Pi, R = r)
```

---

hilbert

*Hilbert Matrix*

---

**Description**

Creates a Hilbert matrix.

**Usage**

```
hilbert(n)
```

**Arguments**

n an integer value, the dimension of the square matrix.



**Details**

In linear algebra, a Hilbert matrix is a matrix with the unit fraction elements.

The Hilbert matrices are canonical examples of ill-conditioned matrices, making them notoriously difficult to use in numerical computation. For example, the 2-norm condition number of a 5x5 Hilbert matrix above is about 4.8e5.

The Hilbert matrix is symmetric and positive definite.

**Value**

hilbert generates a Hilbert matrix of order n.

**References**

Hilbert D., *Collected papers*, vol. II, article 21.

Beckermann B, (2000); *The condition number of real Vandermonde, Krylov and positive definite Hankel matrices*, Numerische Mathematik 85, 553–577, 2000.

Choi, M.D., (1983); *Tricks or Treats with the Hilbert Matrix*, American Mathematical Monthly 90, 301–312, 1983.

Todd, J., (1954); *The Condition Number of the Finite Segment of the Hilbert Matrix*, National Bureau of Standards, Applied Mathematics Series 39, 109–116.

Wilf, H.S., (1970); *Finite Sections of Some Classical Inequalities*, Heidelberg, Springer.

**Examples**

```
## Create a Hilbert Matrix:
H = hilbert(5)
H
```

---

HistogramPlot

*Histogram and Density Plots*


---

**Description**

Returns a histogram, a density, or a logarithmic density plot.

List of Functions:

histPlot	Returns a tailored histogram plot,
densityPlot	Returns a tailored kernel density estimate plot,
logDensityPlot	Returns a tailored log kernel density estimate plot.

**Usage**

```
histPlot(x, labels = TRUE, col = "steelblue", fit = TRUE,
```

```

    title = TRUE, grid = TRUE, rug = TRUE, skip = FALSE, ...)
densityPlot(x, labels = TRUE, col = "steelblue", fit = TRUE, hist = TRUE,
    title = TRUE, grid = TRUE, rug = TRUE, skip = FALSE, ...)
logDensityPlot(x, labels = TRUE, col = "steelblue", robust = TRUE,
    title = TRUE, grid = TRUE, rug = TRUE, skip = FALSE, ...)

```

### Arguments

col	the color for the series. In the univariate case use just a color name like the default, col="steelblue", in the multivariate case we recommend to select the colors from a color palette, e.g. col=heat.colors(ncol(x)).
fit	a logical flag, should a fit added to the Plot?
grid	a logical flag, should a grid be added to the plot? By default TRUE. To plot a horizontal lines only use grid="h" and for vertical lines use grid="v", respectively.
hist	a logical flag, by default TRUE. Should a histogram to be underlaid to the plot?
labels	a logical flag, should the plot be returned with default labels and decorated in an automated way? By default TRUE.
rug	a logical flag, by default TRUE. Should a rug representation of the data added to the plot?
skip	a logical flag, should zeros be skipped in the return Series?
robust	a logical flag, by default TRUE. Should a robust fit added to the plot?
title	a logical flag, by default TRUE. Should a default title added to the plot?
x	an object of class "timeSeries" or any other object which can be transformed by the function as.timeSeries into an object of class timeSeries. The latter case, other then timeSeries objects, is more or less untested.
...	optional arguments to be passed.

### Value

displays a time series plot.

### Examples

```

## data -
SPI <- LPP2005REC[, "SPI"]
plot(SPI, type = "l", col = "steelblue", main = "SP500")
abline(h = 0, col = "grey")

## histPlot -
histPlot(SPI)

## densityPlot -
densityPlot(SPI)

```

hyp

*Hyperbolic Distribution***Description**

Density, distribution function, quantile function and random generation for the hyperbolic distribution.

**Usage**

```
dhyp(x, alpha = 1, beta = 0, delta = 1, mu = 0,
     pm = c("1", "2", "3", "4"), log = FALSE)
phyp(q, alpha = 1, beta = 0, delta = 1, mu = 0,
     pm = c("1", "2", "3", "4"), ...)
qhyp(p, alpha = 1, beta = 0, delta = 1, mu = 0,
     pm = c("1", "2", "3", "4"), ...)
rhyp(n, alpha = 1, beta = 0, delta = 1, mu = 0,
     pm = c("1", "2", "3", "4"))
```

**Arguments**

alpha, beta, delta, mu

shape parameter alpha; skewness parameter beta,  $\text{abs}(\text{beta})$  is in the range  $(0, \text{alpha})$ ; scale parameter delta, delta must be zero or positive; location parameter mu, by default 0. These is the meaning of the parameters in the first parameterization  $\text{pm}=1$  which is the default parameterization selection. In the second parameterization,  $\text{pm}=2$  alpha and beta take the meaning of the shape parameters (usually named) zeta and rho. In the third parameterization,  $\text{pm}=3$  alpha and beta take the meaning of the shape parameters (usually named)  $\bar{x}$  and  $\bar{\chi}$ . In the fourth parameterization,  $\text{pm}=4$  alpha and beta take the meaning of the shape parameters (usually named)  $\bar{a}$  and  $\bar{b}$ .

n                    number of observations.

p                    a numeric vector of probabilities.

pm                   an integer value between 1 and 4 for the selection of the parameterization. The default takes the first parameterization.

x, q                 a numeric vector of quantiles.

log                  a logical, if TRUE, probabilities p are given as  $\log(p)$ .

...                  arguments to be passed to the function integrate.

**Details**

The generator rhyp is based on the HYP algorithm given by Atkinson (1982).

**Value**

All values for the \*hyp functions are numeric vectors: d\* returns the density, p\* returns the distribution function, q\* returns the quantile function, and r\* generates random deviates.

All values have attributes named "param" listing the values of the distributional parameters.

**Author(s)**

David Scott for code implemented from R's contributed package HyperbolicDist.

**References**

Atkinson, A.C. (1982); *The simulation of generalized inverse Gaussian and hyperbolic random variables*, SIAM J. Sci. Stat. Comput. 3, 502–515.

Barndorff-Nielsen O. (1977); *Exponentially decreasing distributions for the logarithm of particle size*, Proc. Roy. Soc. Lond., A353, 401–419.

Barndorff-Nielsen O., Blaesild, P. (1983); *Hyperbolic distributions*. In *Encyclopedia of Statistical Sciences*, Eds., Johnson N.L., Kotz S. and Read C.B., Vol. 3, pp. 700–707. New York: Wiley.

Raible S. (2000); *Levy Processes in Finance: Theory, Numerics and Empirical Facts*, PhD Thesis, University of Freiburg, Germany, 161 pages.

**Examples**

```
## hyp -
  set.seed(1953)
  r = rhyp(5000, alpha = 1, beta = 0.3, delta = 1)
  plot(r, type = "l", col = "steelblue",
       main = "hyp: alpha=1 beta=0.3 delta=1")

## hyp -
  # Plot empirical density and compare with true density:
  hist(r, n = 25, probability = TRUE, border = "white", col = "steelblue")
  x = seq(-5, 5, 0.25)
  lines(x, dhyp(x, alpha = 1, beta = 0.3, delta = 1))

## hyp -
  # Plot df and compare with true df:
  plot(sort(r), (1:5000/5000), main = "Probability", col = "steelblue")
  lines(x, phyp(x, alpha = 1, beta = 0.3, delta = 1))

## hyp -
  # Compute Quantiles:
  qhyp(phyp(seq(-5, 5, 1), alpha = 1, beta = 0.3, delta = 1),
       alpha = 1, beta = 0.3, delta = 1)
```

**Description**

Estimates the parameters of a hyperbolic distribution.

**Usage**

```
hypFit(x, alpha = 1, beta = 0, delta = 1, mu = 0,
       scale = TRUE, doplot = TRUE, span = "auto", trace = TRUE,
       title = NULL, description = NULL, ...)
```

**Arguments**

alpha, beta, delta, mu	alpha is a shape parameter by default 1, beta is a skewness parameter by default 0, note $\text{abs}(\text{beta})$ is in the range (0, alpha), delta is a scale parameter by default 1, note, delta must be zero or positive, and mu is a location parameter, by default 0. These is the meaning of the parameters in the first parameterization pm=1 which is the default parameterization selection. In the second parameterization, pm=2 alpha and beta take the meaning of the shape parameters (usually named) zeta and rho. In the third parameterization, pm=3 alpha and beta take the meaning of the shape parameters (usually named) xi and chi. In the fourth parameterization, pm=4 alpha and beta take the meaning of the shape parameters (usually named) a.bar and b.bar.
description	a character string which allows for a brief description.
doplot	a logical flag. Should a plot be displayed?
scale	a logical flag, by default TRUE. Should the time series be scaled by its standard deviation to achieve a more stable optimization?
span	x-coordinates for the plot, by default 100 values automatically selected and ranging between the 0.001, and 0.999 quantiles. Alternatively, you can specify the range by an expression like <code>span=seq(min, max, times = n)</code> , where, min and max are the left and right endpoints of the range, and n gives the number of the intermediate points.
title	a character string which allows for a project title.
trace	a logical flag. Should the parameter estimation process be traced?
x	a numeric vector.
...	parameters to be parsed.

**Details**

The function `nlm` is used to minimize the "negative" maximum log-likelihood function. `nlm` carries out a minimization using a Newton-type algorithm.

**Value**

The functions tFit, hypFit and nigFit return a list with the following components:

estimate	the point at which the maximum value of the log likelihood function is obtained.
minimum	the value of the estimated maximum, i.e. the value of the log likelihood function.
code	an integer indicating why the optimization process terminated. 1: relative gradient is close to zero, current iterate is probably solution; 2: successive iterates within tolerance, current iterate is probably solution; 3: last global step failed to locate a point lower than estimate. Either estimate is an approximate local minimum of the function or steptol is too small; 4: iteration limit exceeded; 5: maximum step size stepmax exceeded five consecutive times. Either the function is unbounded below, becomes asymptotic to a finite value from above in some direction or stepmax is too small.
gradient	the gradient at the estimated maximum.
steps	number of function calls.

**Examples**

```
## rhyp -
# Simulate Random Variates:
set.seed(1953)
s = rhyp(n = 1000, alpha = 1.5, beta = 0.3, delta = 0.5, mu = -1.0)

## hypFit -
# Fit Parameters:
hypFit(s, alpha = 1, beta = 0, delta = 1, mu = mean(s), doplot = TRUE)
```

---

hypMode

*Hyperbolic Mode*

---

**Description**

Computes the mode of the hyperbolic function.

**Usage**

```
hypMode(alpha = 1, beta = 0, delta = 1, mu = 0, pm = c(1, 2, 3, 4))
```

**Arguments**

alpha, beta, delta, mu

shape parameter alpha; skewness parameter beta,  $abs(beta)$  is in the range  $(0, alpha)$ ; scale parameter delta, delta must be zero or positive; location parameter mu, by default 0. These is the meaning of the parameters in the first parameterization pm=1 which is the default parameterization selection. In the

second parameterization,  $pm=2$  alpha and beta take the meaning of the shape parameters (usually named) zeta and rho. In the third parameterization,  $pm=3$  alpha and beta take the meaning of the shape parameters (usually named) xi and chi. In the fourth parameterization,  $pm=4$  alpha and beta take the meaning of the shape parameters (usually named)  $\bar{a}$  and  $\bar{b}$ .

$pm$  an integer value between 1 and 4 for the selection of the parameterization. The default takes the first parameterization.

### Value

returns the mode in the appropriate parameterization for the hyperbolic distribution. A numeric value.

### Author(s)

David Scott for code implemented from R's contributed package HyperbolicDist.

### References

Atkinson, A.C. (1982); *The simulation of generalized inverse Gaussian and hyperbolic random variables*, SIAM J. Sci. Stat. Comput. 3, 502–515.

Barndorff-Nielsen O. (1977); *Exponentially decreasing distributions for the logarithm of particle size*, Proc. Roy. Soc. Lond., A353, 401–419.

Barndorff-Nielsen O., Blaesild, P. (1983); *Hyperbolic distributions*. In *Encyclopedia of Statistical Sciences*, Eds., Johnson N.L., Kotz S. and Read C.B., Vol. 3, pp. 700–707. New York: Wiley.

Raible S. (2000); *Levy Processes in Finance: Theory, Numerics and Empirical Facts*, PhD Thesis, University of Freiburg, Germany, 161 pages.

### Examples

```
## hypMode -
  hypMode()
```

---

hypMoments

*Hyperbolic Distribution Moments*

---

### Description

Calculates moments of the hyperbbolic distribution function

**Usage**

```
hypMean(alpha=1, beta=0, delta=1, mu=0)
hypVar(alpha=1, beta=0, delta=1, mu=0)
hypSkew(alpha=1, beta=0, delta=1, mu=0)
hypKurt(alpha=1, beta=0, delta=1, mu=0)

hypMoments(order, type = c("raw", "central", "mu"),
           alpha=1, beta=0, delta=1, mu=0)
```

**Arguments**

alpha, beta, delta, mu  
 numeric values. alpha is the first shape parameter; beta is the second shape parameter in the range (0, alpha); delta is the scale parameter, must be zero or positive; mu is the location parameter, by default 0.

order  
 an integer value, the order of the moment.

type  
 a character value, "raw" returns the moments about zero, "central" returns the central moments about the mean, and "mu" returns the moments about the location parameter mu.

**Value**

a numerical value.

**Author(s)**

Diethelm Wuertz.

**References**

Scott, D. J., Wuertz, D. and Tran, T. T. (2008) *Moments of the Generalized Hyperbolic Distribution*. Preprint.

**Examples**

```
## hypMean -
hypMean(alpha=1.1, beta=0.1, delta=0.8, mu=-0.3)

## ghKurt -
hypKurt(alpha=1.1, beta=0.1, delta=0.8, mu=-0.3)

## hypMoments -
hypMoments(4, alpha=1.1, beta=0.1, delta=0.8, mu=-0.3)
hypMoments(4, "central", alpha=1.1, beta=0.1, delta=0.8, mu=-0.3)
```



**Description**

Computes the first four robust moments for the hyperbolic distribution.

**Usage**

```
hypMED(alpha = 1, beta = 0, delta = 1, mu = 0)
hypIQR(alpha = 1, beta = 0, delta = 1, mu = 0)
hypSKEW(alpha = 1, beta = 0, delta = 1, mu = 0)
hypKURT(alpha = 1, beta = 0, delta = 1, mu = 0)
```

**Arguments**

alpha, beta, delta, mu

shape parameter alpha; skewness parameter beta,  $\text{abs}(\text{beta})$  is in the range  $(0, \alpha)$ ; scale parameter delta, delta must be zero or positive; location parameter mu, by default 0. These is the meaning of the parameters in the first parameterization pm=1 which is the default parameterization selection. In the second parameterization, pm=2 alpha and beta take the meaning of the shape parameters (usually named) zeta and rho. In the third parameterization, pm=3 alpha and beta take the meaning of the shape parameters (usually named) xi and chi. In the fourth parameterization, pm=4 alpha and beta take the meaning of the shape parameters (usually named)  $\bar{a}$  and  $\bar{b}$ .

**Value**

All values for the \*hyp functions are numeric vectors: d\* returns the density, p\* returns the distribution function, q\* returns the quantile function, and r\* generates random deviates.

All values have attributes named "param" listing the values of the distributional parameters.

**Author(s)**

Diethelm Wuertz.

**Examples**

```
## hypMED -
# Median:
hypMED(alpha = 1, beta = 0, delta = 1, mu = 0)

## hypIQR -
# Inter-quartile Range:
hypIQR(alpha = 1, beta = 0, delta = 1, mu = 0)

## hypSKEW -
```

```

# Robust Skewness:
hypSKEW(alpha = 1, beta = 0, delta = 1, mu = 0)

## hypKURT -
# Robust Kurtosis:
hypKURT(alpha = 1, beta = 0, delta = 1, mu = 0)

```

---

hypSlider	<i>Hyperbolic Distribution Slider</i>
-----------	---------------------------------------

---

### Description

Displays interactively the dependence of the hyperbolic distribution on its parameters.

### Usage

```
hypSlider()
```

### Value

a tcl/tk based graphical user interface.

This is a nice display for educational purposes to investigate the densities and probabilities of the hyperbolic distribution.

### Examples

```
## hypSlider -
#
```

---

Ids	<i>Set and Retrieve Column/Row Names</i>
-----	--

---

### Description

Sets and retrieves column and row names. The functions are for compatibility with SPlus.

### Usage

```
colIds(x, ...)
rowIds(x, ...)
```

### Arguments

x	a numeric matrix.
...	arguments to be passed.

## Details

Usual in R the functions `colnames`, and `rownames` are used to retrieve and set the names of matrices. The functions `rowIds` and `colIds`, are S-Plus like synonyms.

## Examples

```
## pascal -
# Create Pascal Matrix:
P = pascal(3)
P

## rownames -
rownames(P) <- letters[1:3]
P

## colIds<- -
colIds(P) <- as.character(1:3)
P
```

---

 interactivePlot

*Interactive Plot Utility*


---

## Description

Plots with emphasis on interactive plots.

## Usage

```
interactivePlot(x, choices = paste("Plot", 1:9),
  plotFUN = paste("plot.", 1:9, sep = "."), which = "all", ...)
```

## Arguments

<code>choices</code>	a vector of character strings for the choice menu. By Default "Plot 1" ... "Plot 9" allowing for 9 plots at maximum.
<code>plotFUN</code>	a vector of character strings naming the plot functions. By Default "plot.1" ... "plot.9" allowing for 9 plots at maximum.
<code>which</code>	plot selection, which graph should be displayed? If "which" is a character string named "ask" the user is interactively asked which to plot, if a logical vector of length N, those plots which are set TRUE are displayed, if a character string named "all" all plots are displayed.
<code>x</code>	an object to be plotted.
<code>...</code>	additional arguments passed to the FUN or plot function.

## Examples

```
## Test Plot Function:
testPlot = function(x, which = "all", ...) {
  # Plot Function and Addons:
  plot.1 <- function(x, ...) plot(x, ...)
  plot.2 <- function(x, ...) acf(x, ...)
  plot.3 <- function(x, ...) hist(x, ...)
  plot.4 <- function(x, ...) qqnorm(x, ...)
  # Plot:
  interactivePlot(x,
    choices = c("Series Plot", "ACF", "Histogram", "QQ Plot"),
    plotFUN = c("plot.1", "plot.2", "plot.3", "plot.4"),
    which = which, ...)
  # Return Value:
  invisible()
}
# Plot:
par(mfrow = c(2, 2), cex = 0.7)
testPlot(rnorm(500))

# Try:
# par(mfrow = c(1,1))
# testPlot(rnorm(500), which = "ask")
```

---

 inv

*The Inverse of a Matrix*


---

## Description

Returns the inverse of a matrix.

## Usage

```
inv(x)
```

## Arguments

x                    a numeric matrix.

## Value

returns the inverse matrix.

## Note

The function `inv` is a synonyme to the function `solve`.

## References

Golub, van Loan, (1996); *Matrix Computations*, 3rd edition. Johns Hopkins University Press.

**Examples**

```
## Create Pascal Matrix:
P = pascal(5)
P

## Compute the Inverse Matrix:
inv(P)

## Check:
inv(P)

## Alternatives:
chol2inv(chol(P))
solve(P)
```

---

krigeInterp

*Bivariate Krige Interpolation*


---

**Description**

Bivariate Krige Interpolation.

**Usage**

```
krigeInterp(x, y = NULL, z = NULL, gridPoints = 21,
            xo = seq(min(x), max(x), length = gridPoints),
            yo = seq(min(y), max(y), length = gridPoints),
            extrap = FALSE, polDegree = 6)
```

**Arguments**

<code>x, y, z</code>	the arguments <code>x</code> and <code>y</code> are two numeric vectors of grid points, and <code>z</code> is a numeric matrix or any other rectangular object which can be transformed by the function <code>as.matrix</code> into a matrix object.
<code>gridPoints</code>	an integer value specifying the number of grid points in <code>x</code> and <code>y</code> direction.
<code>xo, yo</code>	two numeric vectors of data points spanning the grid.
<code>extrap</code>	a logical, if <code>TRUE</code> then the data points are extrapolated.
<code>polDegree</code>	the polynomial krige degree, an integer ranging between 1 and 6.

**Value**

`krigeInterp` returns a list with at least three entries, `x`, `y` and `z`. Note, that the returned values, can be directly used by the [persp](#) and [contour](#) 3D plotting methods.

**Note**

The function `krigeInterp()` requires the R package **spatial**.

**See Also**

[akimaInterp](#), [linearInterp](#).

**Examples**

```
## The akima library is not auto-installed because of a different licence.
## krigeInterp - Kriging:
set.seed(1953)
x = runif(999) - 0.5
y = runif(999) - 0.5
z = cos(2*pi*(x^2+y^2))
ans = krigeInterp(x, y, z, extrap = FALSE)
persp(ans, theta = -40, phi = 30, col = "steelblue",
      xlab = "x", ylab = "y", zlab = "z")
contour(ans)
```

---

kron

*Kronecker Product*

---

**Description**

Returns the Kronecker product.

**Usage**

```
kron(x, y)
```

**Arguments**

x, y                    two numeric matrixes.

**Details**

The *Kronecker product* can be computed using the operator `%%` or alternatively using the function `kron` for SPlus compatibility.

**Note**

`kron` is a synonyme to `%%`.

**References**

Golub, van Loan, (1996); *Matrix Computations*, 3rd edition. Johns Hopkins University Press.

**Examples**

```
## Create Pascal Matrix:
P = pascal(3)
P

## Return the Kronecker Product
kron(P, diag(3))
P
```

ks2Test

*Two Sample Kolmogorov–Smirnov Test***Description**

Tests if two series are distributional equivalent.

**Usage**

```
ks2Test(x, y, title = NULL, description = NULL)
```

**Arguments**

x, y	numeric vectors of data values.
title	an optional title string, if not specified the inputs data name is deparsed.
description	optional description string, or a vector of character strings.

**Details**

The test `ks2Test` performs a Kolmogorov–Smirnov two sample test that the two data samples `x` and `y` come from the same distribution, not necessarily a normal distribution. That means that it is not specified what that common distribution is.

**Value**

In contrast to R's output report from S3 objects of class "htest" a different output report is produced. The classical tests presented here return an S4 object of class "fHTEST". The object contains the following slots:

@call	the function call.
@data	the data as specified by the input argument(s).
@test	a list whose elements contain the results from the statistical test. The information provided is similar to a list object of class "htest".
@title	a character string with the name of the test. This can be overwritten specifying a user defined input argument.

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

The slot @test returns an object of class "list" containing (at least) the following elements:

statistic	the value(s) of the test statistic.
p.value	the p-value(s) of the test.
parameters	a numeric value or vector of parameters.
estimate	a numeric value or vector of sample estimates.
conf.int	a numeric two row vector or matrix of 95
method	a character string indicating what type of test was performed.
data.name	a character string giving the name(s) of the data.

### Author(s)

R-core team for hypothesis tests implemented from R's package ctest.

### References

Conover, W. J. (1971); *Practical nonparametric statistics*, New York: John Wiley & Sons.

Lehmann E.L. (1986); *Testing Statistical Hypotheses*, John Wiley and Sons, New York.

### Examples

```
## rnorm -
# Generate Series:
x = rnorm(50)
y = rnorm(50)

## ks2Test -
ks2Test(x, y)
```

---

lcg

*Generator for Portable Random Innovations*


---

### Description

Functions to generate portable random innovations. The functions run under R and S-Plus and generate the same sequence of random numbers. Supported are uniform, normal and Student-t distributed random numbers.

The functions are:

set.lcgseed	Set initial random seed,
get.lcgseed	Get the current value of the random seed,
runif.lcg	Uniform linear congruational generator,
rnorm.lcg	Normal linear congruational generator,
rt.lcg	Student-t linear congruational generator.



**Usage**

```
set.lcgseed(seed = 4711)
get.lcgseed()

runif.lcg(n, min = 0, max = 1)
rnorm.lcg(n, mean = 0, sd = 1)
rt.lcg(n, df)
```

**Arguments**

df	number of degrees of freedom, a positive integer, maybe non-integer.
mean, sd	means and standard deviation of the normal distributed innovations.
min, max	lower and upper limits of the uniform distributed innovations.
seed	an integer value, the random number seed.
n	an integer, the number of random innovations to be generated.

**Details**

A simple portable random number generator for use in R and SPlus. We recommend to use this generator only for comparisons of calculations in R and Splus.

The generator is a linear congruential generator with parameters LCG( $a=13445$ ,  $c=0$ ,  $m=2^{31}-1$ ,  $X=0$ ). It is a simple random number generator which passes the bitwise randomness test.

**Value**

A vector of generated random innovations. The value of the current seed is stored in the variable `lcg.seed`.

**References**

Altman, N.S. (1988); *Bitwise Behavior of Random Number Generators*, SIAM J. Sci. Stat. Comput., 9(5), September, 941–949.

**Examples**

```
## set.lcgseed -
  set.lcgseed(seed = 65890)

## runif.lcg - rnorm.lcg - rt.lcg -
  cbind(runif.lcg(10), rnorm.lcg(10), rt.lcg(10, df = 4))

## get.lcgseed -
  get.lcgseed()

## Note, to overwrite rnorm, use
  # rnorm = rnorm.lcg
  # Going back to rnorm
  # rm(rnorm)
```

---

linearInterp	<i>Bivariate Linear Interpolation</i>
--------------	---------------------------------------

---

**Description**

Bivariate Linear Interpolation. Two options are available gridded and pointwise interpolation.

**Usage**

```
linearInterp(x, y = NULL, z = NULL, gridPoints = 21,
             xo = seq(min(x), max(x), length = gridPoints),
             yo = seq(min(y), max(y), length = gridPoints))
```

```
linearInterpp(x, y = NULL, z = NULL, xo, yo)
```

**Arguments**

x, y, z	for <code>linearInterp</code> the arguments x and y are two numeric vectors of grid points, and z is a numeric matrix or any other rectangular object which can be transformed by the function <code>as.matrix</code> into a matrix object. For <code>linearInterpp</code> we consider either three numeric vectors of equal length or if y and z are NULL, a list with entries x, y, z, or named data frame with x in the first, y in the second, and z in the third column.
gridPoints	an integer value specifying the number of grid points in x and y direction.
xo, yo	for <code>linearInterp</code> two numeric vectors of data points spanning the grid, and for <code>linearInterpp</code> two numeric vectors of data points building pairs for pointwise interpolation.

**Value**

`linearInterp`

returns a list with at least three entries, x, y and z. Note, that the returned values, can be directly used by the `persp` and `contour` 3D plotting methods.

`linearInterpp`

returns a `data.frame` with columns "x", "y", and "z".

**See Also**

[akimaInterp](#), and [krigeInterp](#).

## Examples

```
## linearInterp -  
# Linear Interpolation:  
if (requireNamespace("interp")) {  
  set.seed(1953)  
  x = runif(999) - 0.5  
  y = runif(999) - 0.5  
  z = cos(2*pi*(x^2+y^2))  
  ans = linearInterp(x, y, z, gridPoints = 41)  
  persp(ans, theta = -40, phi = 30, col = "steelblue",  
        xlab = "x", ylab = "y", zlab = "z")  
  contour(ans)  
}
```

---

listDescription	<i>Description File Listing</i>
-----------------	---------------------------------

---

## Description

Lists the content of a description file.

## Usage

```
listDescription(package, character.only = FALSE)
```

## Arguments

`package` a literal character or character string denoting the name of the package to be listed.

`character.only` a logical indicating whether 'package' can be assumed to be character strings.

## Value

prints the description file.

## See Also

[listFunctions](#), [listIndex](#).

## Examples

```
## listDescription -  
listDescription("fBasics")
```

---

listFunctions	<i>Functions Listing</i>
---------------	--------------------------

---

**Description**

Lists and counts functions from packages.

**Usage**

```
listFunctions(package, character.only = FALSE)
countFunctions(package, character.only = FALSE)
```

**Arguments**

`package` a literal character or a character string denoting the name of the package to be listed.

`character.only` a logical indicating whether 'package' can be assumed to be character strings.

**Value**

prints a list and counts of functions.

**See Also**

[listFunctions](#), [listIndex](#).

**Examples**

```
## listFunctions -
listFunctions("fBasics")

## countFunctions -
countFunctions("fBasics")
```

---

listIndex	<i>Index File Listing</i>
-----------	---------------------------

---

**Description**

Lists the content of an index file.

**Usage**

```
listIndex(package, character.only = FALSE)
```

**Arguments**

`package` a literal character string or a character string denoting the name of the package to be listed.

`character.only` a logical indicating whether 'package' can be assumed to be character strings.

**Value**

prints the index file.

**See Also**

[listDescription](#), [listIndex](#).

**Examples**

```
## listIndex -
listIndex("fBasics")
```

---

locationTest	<i>Two Sample Location Tests</i>
--------------	----------------------------------

---

**Description**

Tests if two series differ in their distributional location parameter.

**Usage**

```
locationTest(x, y, method = c("t", "kw2"),
  title = NULL, description = NULL)
```

**Arguments**

`x, y` numeric vectors of data values.

`method` a character string naming which test should be applied.

`title` an optional title string, if not specified the inputs data name is deparsed.

`description` optional description string, or a vector of character strings.

**Details**

The `method="t"` can be used to determine if the two sample means are equal for unpaired data sets. Two variants are used, assuming equal or unequal variances.

The `method="kw2"` performs a Kruskal-Wallis rank sum test of the null hypothesis that the central tendencies or medians of two samples are the same. The alternative is that they differ. Note, that it is not assumed that the two samples are drawn from the same distribution. It is also worth to know that the test assumes that the variables under consideration have underlying continuous distributions.

**Value**

In contrast to R's output report from S3 objects of class "htest" a different output report is produced. The classical tests presented here return an S4 object of class "fHTEST". The object contains the following slots:

@call	the function call.
@data	the data as specified by the input argument(s).
@test	a list whose elements contain the results from the statistical test. The information provided is similar to a list object of class "htest".
@title	a character string with the name of the test. This can be overwritten specifying a user defined input argument.
@description	a character string with an optional user defined description. By default just the current date when the test was applied will be returned.

The slot @test returns an object of class "list" containing (at least) the following elements:

statistic	the value(s) of the test statistic.
p.value	the p-value(s) of the test.
parameters	a numeric value or vector of parameters.
estimate	a numeric value or vector of sample estimates.
conf.int	a numeric two row vector or matrix of 95
method	a character string indicating what type of test was performed.
data.name	a character string giving the name(s) of the data.

**Note**

Some of the test implementations are selected from R's ctest package.

**Author(s)**

R-core team for hypothesis tests implemented from R's package ctest.

**References**

Conover, W. J. (1971); *Practical nonparametric statistics*, New York: John Wiley & Sons.  
 Lehmann E.L. (1986); *Testing Statistical Hypotheses*, John Wiley and Sons, New York.

**Examples**

```
## rnorm -
# Generate Series:
x = rnorm(50)
y = rnorm(50)

## locationTest -
locationTest(x, y, "t")
locationTest(x, y, "kw2")
```

---

maxdd	<i>Drawdown Statistics</i>
-------	----------------------------

---

**Description**

This is a collection and description of functions which compute drawdown statistics. Included are density, distribution function, and random generation for the maximum drawdown distribution. In addition the expectation of drawdowns for Brownian motion can be computed.

The functions are:

dmaxdd	the Density function,
pmaxdd	the Distribution function,
rmaxdd	the random number generator,
maxddStats	the expectation of drawdowns.

**Usage**

```
dmaxdd(x, sd = 1, horizon = 100, N = 1000)
pmaxdd(q, sd = 1, horizon = 100, N = 1000)
rmaxdd(n, mean = 0, sd = 1, horizon = 100)

maxddStats(mean = 0, sd = 1, horizon = 1000)
```

**Arguments**

x, q	a numeric vector of quantiles.
n	an integer value, the number of observations.
mean, sd	two numeric values, the mean and standard deviation.
horizon	an integer value, the (run time) horizon of the investor.
N	an integer value, the precession index for summations. Before you change this value please inspect Magdon-Ismail et. al. (2003).

**Value**

**dmaxdd**  
returns for a trendless Brownian process mean=0 and standard deviation "sd" the density from the probability that the maximum drawdown "D" is larger or equal to "h" in the interval [0,T], where "T" denotes the time horizon of the investor.

**pmaxdd**  
returns for a trendless Brownian process mean=0 and standard deviation "sd" the the probability that the maximum drawdown "D" is larger or equal to "h" in the interval [0,T], where "T" denotes the time horizon of the investor.

`rmaxdd`  
returns for a Brownian Motion process with mean  $\mu$  and standard deviation  $\sigma$  random variates of maximum drawdowns.

`maxddStats`  
returns the expectation Value  $E[D]$  of maximum drawdowns of Brownian Motion for a given drift mean, variance  $\sigma^2$ , and runtime horizon of the Brownian Motion process.

### Note

Currently, for the functions `dmaxdd` and `pmaxdd` only the trend or driftless case is implemented.

### References

Magdon-Ismael M., Atiya A.F., Pratap A., Abu-Mostafa Y.S. (2003); *On the Maximum Drawdown of a Brownian Motion*, Preprint, CalTech, Pasadena USA, p. 24.

### Examples

```
## rmaxdd -
# Set a random seed
set.seed(1953)
# horizon of the investor, time T
horizon = 1000
# number of MC samples, N -> infinity
samples = 1000
# Range of expected Drawdowns
xlim = c(0, 5)*sqrt(horizon)
# Plot Histogram of Simulated Max Drawdowns:
r = rmaxdd(n = samples, mean = 0, sd = 1, horizon = horizon)
hist(x = r, n = 40, probability = TRUE, xlim = xlim,
     col = "steelblue4", border = "white", main = "Max. Drawdown Density")
points(r, rep(0, samples), pch = 20, col = "orange", cex = 0.7)

## dmaxdd -
x = seq(0, xlim[2], length = 200)
d = dmaxdd(x = x, sd = 1, horizon = horizon, N = 1000)
lines(x, d, lwd = 2)

## pmaxdd -
# Count Frequencies of Drawdowns Greater or Equal to "h":
n = 50
x = seq(0, xlim[2], length = n)
g = rep(0, times = n)
for (i in 1:n) g[i] = length (r[r > x[i]]) / samples
plot(x, g, type ="h", lwd = 3,
     xlab = "q", main = "Max. Drawdown Probability")
# Compare with True Probability "G_D(h)":
x = seq(0, xlim[2], length = 5*n)
p = pmaxdd(q = x, sd = 1, horizon = horizon, N = 5000)
lines(x, p, lwd = 2, col="steelblue4")

## maxddStats -
```



```
# Compute expectation Value E[D]:
maxddStats(mean = -0.5, sd = 1, horizon = 10^(1:4))
maxddStats(mean = 0.0, sd = 1, horizon = 10^(1:4))
maxddStats(mean = 0.5, sd = 1, horizon = 10^(1:4))
```

---

nig

*Normal Inverse Gaussian Distribution*


---

### Description

Density, distribution function, quantile function and random generation for the normal inverse Gaussian distribution.

### Usage

```
dnig(x, alpha = 1, beta = 0, delta = 1, mu = 0, log = FALSE)
pnig(q, alpha = 1, beta = 0, delta = 1, mu = 0)
qnig(p, alpha = 1, beta = 0, delta = 1, mu = 0)
rnig(n, alpha = 1, beta = 0, delta = 1, mu = 0)
```

### Arguments

alpha, beta, delta, mu	shape parameter alpha; skewness parameter beta, $abs(beta)$ is in the range $(0, alpha)$ ; scale parameter delta, delta must be zero or positive; location parameter mu, by default 0. These are the parameters in the first parameterization.
log	a logical flag by default FALSE. Should labels and a main title drawn to the plot?
n	number of observations.
p	a numeric vector of probabilities.
x, q	a numeric vector of quantiles.

### Details

The random deviates are calculated with the method described by Raible (2000).

### Value

All values for the \*nig functions are numeric vectors: d\* returns the density, p\* returns the distribution function, q\* returns the quantile function, and r\* generates random deviates.

All values have attributes named "param" listing the values of the distributional parameters.

### Author(s)

David Scott for code implemented from R's contributed package HyperbolicDist.

## References

- Atkinson, A.C. (1982); *The simulation of generalized inverse Gaussian and hyperbolic random variables*, SIAM J. Sci. Stat. Comput. 3, 502–515.
- Barndorff-Nielsen O. (1977); *Exponentially decreasing distributions for the logarithm of particle size*, Proc. Roy. Soc. Lond., A353, 401–419.
- Barndorff-Nielsen O., Blaesild, P. (1983); *Hyperbolic distributions*. In *Encyclopedia of Statistical Sciences*, Eds., Johnson N.L., Kotz S. and Read C.B., Vol. 3, pp. 700–707. New York: Wiley.
- Raible S. (2000); *Levy Processes in Finance: Theory, Numerics and Empirical Facts*, PhD Thesis, University of Freiburg, Germany, 161 pages.

## Examples

```
## nig -
  set.seed(1953)
  r = rnig(5000, alpha = 1, beta = 0.3, delta = 1)
  plot(r, type = "l", col = "steelblue",
       main = "nig: alpha=1 beta=0.3 delta=1")

## nig -
  # Plot empirical density and compare with true density:
  hist(r, n = 25, probability = TRUE, border = "white", col = "steelblue")
  x = seq(-5, 5, 0.25)
  lines(x, dnig(x, alpha = 1, beta = 0.3, delta = 1))

## nig -
  # Plot df and compare with true df:
  plot(sort(r), (1:5000/5000), main = "Probability", col = "steelblue")
  lines(x, pnig(x, alpha = 1, beta = 0.3, delta = 1))

## nig -
  # Compute Quantiles:
  qnig(pnig(seq(-5, 5, 1), alpha = 1, beta = 0.3, delta = 1),
       alpha = 1, beta = 0.3, delta = 1)
```

---

nigFit

*Fit of a Normal Inverse Gaussian Distribution*


---

## Description

Estimates the parameters of a normal inverse Gaussian distribution.

## Usage

```
nigFit(x, alpha = 1, beta = 0, delta = 1, mu = 0,
       method = c("mle", "gmm", "mps", "vmgs"), scale = TRUE, doplot = TRUE,
       span = "auto", trace = TRUE, title = NULL, description = NULL, ...)
```

**Arguments**

alpha, beta, delta, mu	The parameters are alpha, beta, delta, and mu: shape parameter alpha; skewness parameter beta, $\text{abs}(\text{beta})$ is in the range (0, alpha); scale parameter delta, delta must be zero or positive; location parameter mu, by default 0. These is the meaning of the parameters in the first parameterization pm=1 which is the default parameterization selection. In the second parameterization, pm=2 alpha and beta take the meaning of the shape parameters (usually named) zeta and rho. In the third parameterization, pm=3 alpha and beta take the meaning of the shape parameters (usually named) xi and chi. In the fourth parameterization, pm=4 alpha and beta take the meaning of the shape parameters (usually named) $\bar{a}$ and $\bar{b}$ .
description	a character string which allows for a brief description.
doplot	a logical flag. Should a plot be displayed?
method	a character string. Either "mle", Maximum Likelihood Estimation, the default, "gmm" Generalized Method of Moments Estimation, "mps" Maximum Product Spacings Estimation, or "vmgs" Minimum Variance Product Spacings Estimation.
scale	a logical flag, by default TRUE. Should the time series be scaled by its standard deviation to achieve a more stable optimization?
span	x-coordinates for the plot, by default 100 values automatically selected and ranging between the 0.001, and 0.999 quantiles. Alternatively, you can specify the range by an expression like <code>span=seq(min, max, times = n)</code> , where, min and max are the left and right endpoints of the range, and n gives the number of the intermediate points.
title	a character string which allows for a project title.
trace	a logical flag. Should the parameter estimation process be traced?
x	a numeric vector.
...	parameters to be parsed.

**Value**

The functions `tFit`, `hypFit` and `nigFit` return a list with the following components:

estimate	the point at which the maximum value of the log likelihood function is obtained.
minimum	the value of the estimated maximum, i.e. the value of the log likelihood function.
code	an integer indicating why the optimization process terminated. 1: relative gradient is close to zero, current iterate is probably solution; 2: successive iterates within tolerance, current iterate is probably solution; 3: last global step failed to locate a point lower than estimate. Either estimate is an approximate local minimum of the function or <code>steptol</code> is too small; 4: iteration limit exceeded; 5: maximum step size <code>stepmax</code> exceeded five consecutive times. Either the function is unbounded below, becomes asymptotic to a finite value from above in some direction or <code>stepmax</code> is too small.

gradient        the gradient at the estimated maximum.  
 steps            number of function calls.

### Examples

```
## nigFit -
# Simulate Random Variates:
set.seed(1953)
s = rnig(n = 1000, alpha = 1.5, beta = 0.3, delta = 0.5, mu = -1.0)

## nigFit -
# Fit Parameters:
nigFit(s, alpha = 1, beta = 0, delta = 1, mu = mean(s), doplot = TRUE)
```

---

nigMode

*Normal Inverse Gaussian Mode*

---

### Description

Computes the mode of the norm inverse Gaussian distribution.

### Usage

```
nigMode(alpha = 1, beta = 0, delta = 1, mu = 0)
```

### Arguments

alpha, beta, delta, mu

shape parameter alpha; skewness parameter beta,  $\text{abs}(\text{beta})$  is in the range  $(0, \alpha)$ ; scale parameter delta, delta must be zero or positive; location parameter mu, by default 0. These are the parameters in the first parameterization.

### Value

returns the mode for the normal inverse Gaussian distribution. A numeric value.

### References

- Atkinson, A.C. (1982); *The simulation of generalized inverse Gaussian and hyperbolic random variables*, SIAM J. Sci. Stat. Comput. 3, 502–515.
- Barndorff-Nielsen O. (1977); *Exponentially decreasing distributions for the logarithm of particle size*, Proc. Roy. Soc. Lond., A353, 401–419.
- Barndorff-Nielsen O., Blaesild, P. (1983); *Hyperbolic distributions*. In *Encyclopedia of Statistical Sciences*, Eds., Johnson N.L., Kotz S. and Read C.B., Vol. 3, pp. 700–707. New York: Wiley.
- Raible S. (2000); *Levy Processes in Finance: Theory, Numerics and Empirical Facts*, PhD Thesis, University of Freiburg, Germany, 161 pages.

## Examples

```
## nigMode -  
nigMode()
```

---

nigMoments

*Moments for the Normal Inverse Gaussian*

---

## Description

Computes the first four moments for the normal inverse Gaussian distribution.

## Usage

```
nigMean(alpha = 1, beta = 0, delta = 1, mu = 0)  
nigVar(alpha = 1, beta = 0, delta = 1, mu = 0)  
nigSkew(alpha = 1, beta = 0, delta = 1, mu = 0)  
nigKurt(alpha = 1, beta = 0, delta = 1, mu = 0)
```

## Arguments

alpha, beta, delta, mu

are numeric values where alpha is the location parameter, beta is the location parameter, delta is the first shape parameter, and mu is the second shape parameter.

## Value

All values for the \*nig functions are numeric vectors: d\* returns the density, p\* returns the distribution function, q\* returns the quantile function, and r\* generates random deviates.

All values have attributes named "param" listing the values of the distributional parameters.

## Author(s)

Diethelm Wuertz.

## References

Scott, D. J., Wuertz, D. and Tran, T. T. (2008) *Moments of the Generalized Hyperbolic Distribution*. Preprint.

**Examples**

```
## nigMean -
# Median:
nigMean(alpha = 1, beta = 0, delta = 1, mu = 0)

## nigVar -
# Inter-quartile Range:
nigVar(alpha = 1, beta = 0, delta = 1, mu = 0)

## nigSKEW -
# Robust Skewness:
nigSkew(alpha = 1, beta = 0, delta = 1, mu = 0)

## nigKurt -
# Robust Kurtosis:
nigKurt(alpha = 1, beta = 0, delta = 1, mu = 0)
```

---

nigRobMoments

*Robust Moments for the NIG*


---

**Description**

Computes the first four robust moments for the Normal Inverse Gaussian Distribution.

**Usage**

```
nigMED(alpha = 1, beta = 0, delta = 1, mu = 0)
nigIQR(alpha = 1, beta = 0, delta = 1, mu = 0)
nigSKEW(alpha = 1, beta = 0, delta = 1, mu = 0)
nigKURT(alpha = 1, beta = 0, delta = 1, mu = 0)
```

**Arguments**

alpha, beta, delta, mu

are numeric values where alpha is the location parameter, beta is the location parameter, delta is the first shape parameter, and mu is the second shape parameter.

**Value**

All values for the \*nig functions are numeric vectors: d\* returns the density, p\* returns the distribution function, q\* returns the quantile function, and r\* generates random deviates.

All values have attributes named "param" listing the values of the distributional parameters.

**Author(s)**

Diethelm Wuertz.

**Examples**

```
## nigMED -  
# Median:  
nigMED(alpha = 1, beta = 0, delta = 1, mu = 0)  
  
## nigIQR -  
# Inter-quartile Range:  
nigIQR(alpha = 1, beta = 0, delta = 1, mu = 0)  
  
## nigSKEW -  
# Robust Skewness:  
nigSKEW(alpha = 1, beta = 0, delta = 1, mu = 0)  
  
## nigKURT -  
# Robust Kurtosis:  
nigKURT(alpha = 1, beta = 0, delta = 1, mu = 0)
```

---

nigShapeTriangle	<i>NIG Shape Triangle</i>
------------------	---------------------------

---

**Description**

Plots the normal inverse Gaussian Shape Triangle.

**Usage**

```
nigShapeTriangle(object, add = FALSE, labels = TRUE, ...)
```

**Arguments**

object	an object of class "fDISTFIT" as returned by the function nigFit.
add	a logical value. Should another point added to the NIG shape triangle? By default FALSE, a new plot will be created.
labels	a logical flag by default TRUE. Should the logarithm of the density be returned?
...	arguments to be passed to the function integrate.

**Value**

displays the parameters of fitted distributions in the NIG shape triangle.

**Author(s)**

David Scott for code implemented from R's contributed package HyperbolicDist.

## References

Atkinson, A.C. (1982); *The simulation of generalized inverse Gaussian and hyperbolic random variables*, SIAM J. Sci. Stat. Comput. 3, 502–515.

Barndorff-Nielsen O. (1977); *Exponentially decreasing distributions for the logarithm of particle size*, Proc. Roy. Soc. Lond., A353, 401–419.

Barndorff-Nielsen O., Blaesild, P. (1983); *Hyperbolic distributions*. In *Encyclopedia of Statistical Sciences*, Eds., Johnson N.L., Kotz S. and Read C.B., Vol. 3, pp. 700–707. New York: Wiley.

Raible S. (2000); *Levy Processes in Finance: Theory, Numerics and Empirical Facts*, PhD Thesis, University of Freiburg, Germany, 161 pages.

## Examples

```
## nigShapeTriangle -
#
```

---

nigSlider

*nigerbolic Distribution Slider*

---

## Description

Displays interactively the dependence of the nigerbolic distribution on its parameters.

## Usage

```
nigSlider()
```

## Value

a tcl/tk based graphical user interface.

This is a nice display for educational purposes to investigate the densities and probabilities of the invetrse Gaussian distribution.

## Examples

```
## nigSlider -
# nigSlider()
```



---

norm

*Matrix Norm*

---

### Description

Returns the norm of a matrix.

### Usage

```
norm2(x, p = 2)
```

### Arguments

**x** a numeric matrix.

**p** an integer value, 1, 2 or Inf.  $p=1$  - The maximum absolute column sum norm which is defined as the maximum of the sum of the absolute valued elements of columns of the matrix.  $p=2$  - The spectral norm is "the norm" of a matrix  $X$ . This value is computed as the square root of the maximum eigenvalue of  $CX$  where  $C$  is the conjugate transpose.  $p=Inf$  - The maximum absolute row sum norm is defined as the maximum of the sum of the absolute valued elements of rows of the matrix.

### Details

The function `norm2` computes the norm of a matrix. Three choices are possible:

$p=1$  - The maximum absolute column sum norm which is defined as the maximum of the sum of the absolute valued elements of columns of the matrix.

$p=2$  - The spectral norm is "the norm" of a matrix  $X$ . This value is computed as the square root of the maximum eigenvalue of  $CX$  where  $C$  is the conjugate transpose.

$p=Inf$  - The maximum absolute row sum norm is defined as the maximum of the sum of the absolute valued elements of rows of the matrix.

Note, the function `fBasics::norm()` has become obsolete, since `base::norm()` has become available in the R base environment. To avoid conflicts with `norm()` we have renamed in the `fBasics` package as `norm2`.

### Value

returns the value of the norm of the matrix.

### References

Golub, van Loan, (1996); *Matrix Computations*, 3rd edition. Johns Hopkins University Press.

**Examples**

```
## Create Pascal Matrix:
P <- pascal(5)
P

## Return the Norm of the Matrix:
norm2(P)
```

---

NormalityTests            *Normality Tests*

---

**Description**

A collection and description of functions of one sample tests for testing normality of financial return series.

The functions for testing normality are:

ksnormTest	Kolmogorov-Smirnov normality test,
shapiroTest	Shapiro-Wilk's test for normality,
jarqueberaTest	Jarque-Bera test for normality,
dagoTest	D'Agostino normality test.

Functions for high precision Jarque Bera LM and ALM tests:

jbTest    Performs finite sample adjusted JB LM and ALM test.

Additional functions for testing normality from the 'nortest' package:

adTest	Anderson-Darling normality test,
cvmTest	Cramer-von Mises normality test,
lillieTest	Lilliefors (Kolmogorov-Smirnov) normality test,
pchiTest	Pearson chi-square normality test,
sfTest	Shapiro-Francia normality test.

For SPlus/Finmetrics Compatibility:

normalTest    test suite for some normality tests.

**Usage**

```
ksnormTest(x, title = NULL, description = NULL)

jbTest(x, title = NULL, description = NULL)
shapiroTest(x, title = NULL, description = NULL)
```

```

normalTest(x, method = c("sw", "jb"), na.rm = FALSE)

jarqueberaTest(x, title = NULL, description = NULL)
dagoTest(x, title = NULL, description = NULL)

adTest(x, title = NULL, description = NULL)
cvmTest(x, title = NULL, description = NULL)
lillieTest(x, title = NULL, description = NULL)
pchiTest(x, title = NULL, description = NULL)
sfTest(x, title = NULL, description = NULL)

```

### Arguments

description	optional description string, or a vector of character strings.
method	[normalTest] - indicates four different methods for the normality test, "ks" for the Kolmogorov-Smirnov one-sample test, "sw" for the Shapiro-Wilk test, "jb" for the Jarque-Bera Test, and "da" for the D'Agostino Test. The default value is "ks".
na.rm	[normalTest] - a logical value. Should missing values removed before computing the tests? The default value is FALSE.
title	an optional title string, if not specified the inputs data name is deparsed.
x	a numeric vector of data values or a S4 object of class <code>timeSeries</code> .

### Details

The hypothesis tests may be of interest for many financial and economic applications, especially for the investigation of univariate time series returns.

#### Normal Tests:

Several tests for testing if the records from a data set are normally distributed are available. The input to all these functions may be just a vector `x` or a univariate time series object `x` of class `timeSeries`.

First there exists a wrapper function which allows to call one from two normal tests either the Shapiro-Wilks test or the Jarque-Bera test. This wrapper was introduced for compatibility with S-Plus' `FinMetrics` package.

Also available are the Kolmogorov-Smirnov one sample test and the D'Agostino normality test.

The remaining five normal tests are the Anderson-Darling test, the Cramer-von Mises test, the Lilliefors (Kolmogorov-Smirnov) test, the Pearson chi-square test, and the Shapiro-Francia test. They are calling functions from R's contributed package `norTest`. The difference to the original test functions implemented in R and from contributed R packages is that the `Rmetrics` functions accept time series objects as input and give a more detailed output report.

The Anderson-Darling test is used to test if a sample of data came from a population with a specific distribution, here the normal distribution. The `adTest` goodness-of-fit test can be considered as a modification of the Kolmogorov-Smirnov test which gives more weight to the tails than does the `ksnormTest`.

**Value**

In contrast to R's output report from S3 objects of class "htest" a different output report is produced. The tests here return an S4 object of class "fHTEST". The object contains the following slots:

@call	the function call.
@data	the data as specified by the input argument(s).
@test	a list whose elements contain the results from the statistical test. The information provided is similar to a list object of class "htest".
@title	a character string with the name of the test. This can be overwritten specifying a user defined input argument.
@description	a character string with an optional user defined description. By default just the current date when the test was applied will be returned.

The slot @test returns an object of class "list" containing the following (otionally empty) elements:

statistic	the value(s) of the test statistic.
p.value	the p-value(s) of the test.
parameters	a numeric value or vector of parameters.
estimate	a numeric value or vector of sample estimates.
conf.int	a numeric two row vector or matrix of 95
method	a character string indicating what type of test was performed.
data.name	a character string giving the name(s) of the data.

The meaning of the elements of the @test slot is the following:

`ksnormTest`

returns the values for the 'D' statistic and p-values for the three alternatives 'two-sided', 'less' and 'greater'.

`shapiroTest`

returns the values for the 'W' statistic and the p-value.

`jarqueberaTest`

`jbTest`

returns the values for the 'Chi-squared' statistic with 2 degrees of freedom, and the asymptotic p-value. `jbTest` is the finite sample version of the Jarque Bera Lagrange multiplier, LM, and adjusted Lagrange multiplier test, ALM.

`dagoTest`

returns the values for the 'Chi-squared', the 'Z3' (Skewness) and 'Z4' (Kurtosis) statistic together with the corresponding p values.

`adTest`

returns the value for the 'A' statistic and the p-value.

`cvmTest`

returns the value for the 'W' statistic and the p-value.

`lillieTest`

returns the value for the 'D' statistic and the p-value.

`pchiTest`

returns the value for the 'P' statistic and the p-values for the adjusted and not adjusted test cases. In addition the number of classes is printed, taking the default value due to Moore (1986) computed from the expression  $n.classes = \text{ceiling}(2 * (n^{(2/5)}))$ , where n is the number of observations.

`sfTest`

returns the value for the 'W' statistic and the p-value.

### Note

Some of the test implementations are selected from R's `cstest` and `nortest` packages.

### Author(s)

R-core team for the tests from R's `cstest` package,  
 Adrian Trapletti for the runs test from R's `tseries` package,  
 Juergen Gross for the normal tests from R's `nortest` package,  
 James Filliben for the Fortran program producing the runs report,  
 Diethelm Wuertz and Helmut Katzgraber for the finite sample JB tests,  
 Diethelm Wuertz for the Rmetrics R-port.  
 Earlier versions of these functions were based on Fortran code of Paul Johnson.

### References

- Anderson T.W., Darling D.A. (1954); *A Test of Goodness of Fit*, JASA 49:765–69.
- Conover, W. J. (1971); *Practical nonparametric statistics*, New York: John Wiley & Sons.
- D'Agostino R.B., Pearson E.S. (1973); *Tests for Departure from Normality*, Biometrika 60, 613–22.
- D'Agostino R.B., Rosman B. (1974); *The Power of Geary's Test of Normality*, Biometrika 61, 181–84.
- Durbin J. (1961); *Some Methods of Constructing Exact Tests*, Biometrika 48, 41–55.
- Durbin, J. (1973); *Distribution Theory Based on the Sample Distribution Function*, SIAM, Philadelphia.
- Geary R.C. (1947); *Testing for Normality*; Biometrika 36, 68–97.
- Lehmann E.L. (1986); *Testing Statistical Hypotheses*, John Wiley and Sons, New York.
- Linnet K. (1988); *Testing Normality of Transformed Data*, Applied Statistics 32, 180–186.
- Moore, D.S. (1986); *Tests of the chi-squared type*, In: D'Agostino, R.B. and Stephens, M.A., eds., *Goodness-of-Fit Techniques*, Marcel Dekker, New York.
- Shapiro S.S., Francia R.S. (1972); *An Approximate Analysis of Variance Test for Normality*, JASA 67, 215–216.
- Shapiro S.S., Wilk M.B., Chen V. (1968); *A Comparative Study of Various Tests for Normality*, JASA 63, 1343–72.
- Thode H.C. (2002); *Testing for Normality*, Marcel Dekker, New York.

Weiss M.S. (1978); *Modification of the Kolmogorov-Smirnov Statistic for Use with Correlated Data*, JASA 73, 872–75.

Wuertz D., Katzgraber H.G. (2005); *Precise finite-sample quantiles of the Jarque-Bera adjusted Lagrange multiplier test*, ETHZ Preprint.

## Examples

```
## Series:
x = rnorm(100)

## ksnormTests -
# Kolmogorov - Smirnov One-Sampel Test
ksnormTest(x)

## shapiroTest - Shapiro-Wilk Test
shapiroTest(x)

## jarqueberaTest -
# Jarque - Bera Test
# jarqueberaTest(x)
# jbTest(x)
```

---

normRobMoments

*Robust Moments for the NORM*

---

## Description

Computes the first four robust moments for the Normal Distribution.

## Usage

```
normMED(mean = 0, sd = 1)
normIQR(mean = 0, sd = 1)
normSKEW(mean = 0, sd = 1)
normKURT(mean = 0, sd = 1)
```

## Arguments

mean	location parameter
sd	scale parameter

## Value

All values for the \*norm functions are numeric vectors: d\* returns the density, p\* returns the distribution function, q\* returns the quantile function, and r\* generates random deviates.

All values have attributes named "param" listing the values of the distributional parameters.

**Author(s)**

Diethelm Wuertz.

**Examples**

```
## normMED -  
# Median:  
normMED(mean = 0, sd = 1)  
  
## normIQR -  
# Inter-quartile Range:  
normIQR(mean = 0, sd = 1)  
  
## normSKEW -  
# Robust Skewness:  
normSKEW(mean = 0, sd = 1)  
  
## normKURT -  
# Robust Kurtosis:  
normKURT(mean = 0, sd = 1)
```

---

pascal

*Pascal Matrix*

---

**Description**

Creates a Pascal matrix.

**Usage**

```
pascal(n)
```

**Arguments**

n                    an integer value, the dimension of the square matrix.

**Details**

The function `pascal` generates a Pascal matrix of order  $n$  which is a symmetric, positive, definite matrix with integer entries made up from Pascal's triangle. The determinant of a Pascal matrix is 1. The inverse of a Pascal matrix has integer entries. If  $\lambda$  is an eigenvalue of a Pascal matrix, then  $1/\lambda$  is also an eigenvalue of the matrix. Pascal matrices are ill-conditioned.

**References**

- Call G.S., Velleman D.J., (1993); *Pascal's matrices*, American Mathematical Monthly 100, 372–376.
- Edelman A., Strang G., (2004); *Pascal Matrices*, American Mathematical Monthly 111, 361–385.

**Examples**

```
## Create Pascal Matrix:
P = pascal(5)
P

## Determinant
det(pascal(5))
det(pascal(10))
det(pascal(15))
det(pascal(20))
```

---

pdl

*Polynomial Distributed Lags*

---

**Description**

Returns a regressor matrix for polynomial distributed lags.

**Usage**

```
pdl(x, d = 2, q = 3, trim = FALSE)
```

**Arguments**

x	a numeric vector.
d	an integer specifying the order of the polynomial.
q	an integer specifying the number of lags to use in creating polynomial distributed lags. This must be greater than d.
trim	a logical flag; if TRUE, the missing values at the beginning of the returned matrix will be trimmed.

**See Also**

[tslag](#).

**Examples**

```
## pdl -
#
```



---

positiveDefinite	<i>Positive Definite Matrixes</i>
------------------	-----------------------------------

---

**Description**

Checks if a matrix is positive definite and/or forces a matrix to be positive definite.

**Usage**

```
isPositiveDefinite(x)  
makePositiveDefinite(x)
```

**Arguments**

x                    a square numeric matrix.

**Details**

The function `isPositiveDefinite` checks if a square matrix is positive definite.

The function `makePositiveDefinite` forces a matrix to be positive definite.

**Author(s)**

Korbinian Strimmer.

**Examples**

```
## isPositiveDefinite -  
# the 3x3 Pascal Matrix is positive definite  
isPositiveDefinite(pascal(3))
```

---

print	<i>Print Control</i>
-------	----------------------

---

**Description**

Unlists and prints a control object.

**Usage**

```
## S3 method for class 'control'  
print(x, ...)
```

**Arguments**

x                    the object to be printed.  
...                   arguments to be passed.

**Value**

```
print.control
prints control.
```

**Examples**

```
## print -
control = list(n = 211, seed = 54, name = "generator")
print(control)
class(control) = "control"
print(control)
```

---

QuantileQuantilePlots *Quantile-Quantile Plots*

---

**Description**

Returns quantile-quantile plots for the normal, the normal inverse Gaussian, the generalized hyperbolic Student-t and the generalized lambda distribution.

List of Functions:

qqnormPlot	Returns a tailored Normal quantile-quantile plot,
qqnigPlot	Returns a tailored NIG quantile-quantile plot,
qqghtPlot	Returns a tailored GHT quantile-quantile plot,
qqgldPlot	Returns a tailored GLD quantile-quantile plot.

**Usage**

```
qqnormPlot(x, labels = TRUE, col = "steelblue", pch = 19,
  title = TRUE, mtext = TRUE, grid = FALSE, rug = TRUE,
  scale = TRUE, ...)
qqnigPlot(x, labels = TRUE, col = "steelblue", pch = 19,
  title = TRUE, mtext = TRUE, grid = FALSE, rug = TRUE,
  scale = TRUE, ...)
qqghtPlot(x, labels = TRUE, col = "steelblue", pch = 19,
  title = TRUE, mtext = TRUE, grid = FALSE, rug = TRUE,
  scale = TRUE, ...)
qqgldPlot(x, labels = TRUE, col = "steelblue", pch = 19,
  title = TRUE, mtext = TRUE, grid = FALSE, rug = TRUE,
  scale = TRUE, ...)
```

**Arguments**

x	an object of class "timeSeries" or any other object which can be transformed by the function <code>as.timeSeries</code> into an object of class <code>timeSeries</code> . The latter case, other than <code>timeSeries</code> objects, is more or less untested.
labels	a logical flag, should the plot be returned with default labels and decorated in an automated way? By default TRUE.
col	the color for the series. In the univariate case use just a color name like the default, <code>col="steelblue"</code> , in the multivariate case we recommend to select the colors from a color palette, e.g. <code>col=heat.colors(ncol(x))</code> .
pch	an integer value, by default 19. Which plot character should be used in the plot?
title	a logical flag, by default TRUE. Should a default title added to the plot?
mtext	a logical flag, by default TRUE. Should a marginal text be printed on the third site of the graph?
grid	a logical flag, should a grid be added to the plot? By default TRUE. To plot a horizontal lines only use <code>grid="h"</code> and for vertical lines use <code>grid="v"</code> , respectively.
rug	a logical flag, by default TRUE. Should a rug representation of the data added to the plot?
scale	a logical flag, by default TRUE. Should the time series be scaled for the investigation?
...	optional arguments to be passed.

**Value**

displays a quantile-quantile plot.

**Author(s)**

Diethelm Wuertz for the Rmetrics R-port.

**Examples**

```
## data -
SPI <- LPP2005REC[, "SPI"]
plot(SPI, type = "l", col = "steelblue", main = "SP500")
abline(h = 0, col = "grey")

## qqPlot -
qqnormPlot(SPI)
```

---

`returnSeriesGUI`*Return Series Plots*

---

**Description**

A graphical user interface to display financial time series plots.

List of Functions:

`returnSeriesGUI` Opens a GUI for return series plots.

**Usage**

```
returnSeriesGUI(x)
```

**Arguments**

`x` an object of class "timeSeries" or any other object which can be transformed by the function `as.timeSeries` into an object of class `timeSeries`. The latter case, other than `timeSeries` objects, is more or less untested.

**Value**

```
returnSeriesGUI
```

For the `returnSeriesGUI` function, beside the graphical user interface no values are returned.

**Author(s)**

Diethelm Wuertz for the Rmetrics R-port.

**Examples**

```
##
```

---

`rk`*The Rank of a Matrix*

---

**Description**

Returns the rank of a matrix.

**Usage**

```
rk(x, method = c("qr", "chol"))
```

**Arguments**

`x` a numeric matrix.

`method` a character string. For `method = "qr"` the rank is computed as `qr(x)$rank`, or alternatively for `method="chol"` the rank is computed as `attr(chol(x, pivot=TRUE), "rank")`.

**Details**

The function `rk` computes the rank of a matrix which is the dimension of the range of the matrix corresponding to the number of linearly independent rows or columns of the matrix, or to the number of nonzero singular values.

The rank of a matrix is also named `inear map`.

**References**

Golub, van Loan, (1996); *Matrix Computations*, 3rd edition. Johns Hopkins University Press.

**Examples**

```
## Create Pascal Matrix:
P = pascal(5)
P

## Compute the Rank:
rk(P)
rk(P, "chol")
```

---

rowStats

*Row Statistics*


---

**Description**

Functions to compute row statistical properties of financial and economic time series data.

The functions are:

<code>rowStats</code>	calculates row statistics,
<code>rowSds</code>	calculates row standard deviations,
<code>rowVars</code>	calculates row variances,
<code>rowSkewness</code>	calculates row skewness,
<code>rowKurtosis</code>	calculates row kurtosis,
<code>rowMaxs</code>	calculates maximum values in each row,
<code>rowMins</code>	calculates minimum values in each row,
<code>rowProds</code>	computes product of all values in each row,
<code>rowQuantiles</code>	computes quantiles of each row.

**Usage**

```
rowStats(x, FUN, ...)
```

```
rowSds(x, ...)  
rowVars(x, ...)  
rowSkewness(x, ...)  
rowKurtosis(x, ...)  
rowMaxs(x, ...)  
rowMins(x, ...)  
rowProds(x, ...)  
rowQuantiles(x, prob = 0.05, ...)  
  
rowStdevs(x, ...)  
rowAvgs(x, ...)
```

**Arguments**

FUN	a function name. The statistical function to be applied.
prob	a numeric value, the probability with value in [0,1].
x	a rectangular object which can be transformed into a matrix by the function <code>as.matrix</code> .
...	arguments to be passed.

**Value**

the functions return a numeric vector of the statistics.

**See Also**

`link{colStats}`.

**Examples**

```
## Simulated Return Data in Matrix Form:  
x = matrix(rnorm(10*10), nrow = 10)  
  
## rowStats -  
rowStats(x, FUN = mean)  
  
## rowMaxs -  
rowMaxs(x)
```

---

sampleLMoments	<i>Sample L-Moments</i>
----------------	-------------------------

---

**Description**

Computes L-moments from an empirical sample data set.

**Usage**

```
sampleLmoments(x, rmax=4)
```

**Arguments**

`x` are numeric vector, the sample values.  
`rmax` an integer value, the number of L-moments to be returned.

**Value**

All values for the \*sample functions are numeric vectors: `d*` returns the density, `p*` returns the distribution function, `q*` returns the quantile function, and `r*` generates random deviates.

All values have attributes named "param" listing the values of the distributional parameters.

**Author(s)**

Diethelm Wuertz.

**Examples**

```
## Sample:  
x = rt(100, 4)  
  
## sampleLmoments -  
sampleLmoments(x)
```

---

sampleRobMoments	<i>Robust Moments for the GLD</i>
------------------	-----------------------------------

---

**Description**

Computes the first four robust moments for the Normal Inverse Gaussian Distribution.

**Usage**

```
sampleMED(x)
sampleIQR(x)
sampleSKEW(x)
sampleKURT(x)
```

**Arguments**

`x` are numeric vector, the sample values.

**Value**

All values for the \*sample functions are numeric vectors: `d*` returns the density, `p*` returns the distribution function, `q*` returns the quantile function, and `r*` generates random deviates.

All values have attributes named "param" listing the values of the distributional parameters.

**Author(s)**

Diethelm Wuertz.

**Examples**

```
## Sample:
x = rt(100, 4)

## sampleMED -
# Median:
sampleMED(x)

## sampleIQR -
# Inter-quartile Range:
sampleIQR(x)

## sampleSKEW -
# Robust Skewness:
sampleSKEW(x)

## sampleKURT -
# Robust Kurtosis:
sampleKURT(x)
```

---

scaleTest

*Two Sample Scale Tests*

---

**Description**

Tests if two series differ in their distributional scale parameter.



**Usage**

```
scaleTest(x, y, method = c("ansari", "mood"),
          title = NULL, description = NULL)
```

**Arguments**

<code>x, y</code>	numeric vectors of data values.
<code>method</code>	a character string naming which test should be applied.
<code>title</code>	an optional title string, if not specified the inputs data name is deparsed.
<code>description</code>	optional description string, or a vector of character strings.

**Details**

The `method="ansari"` performs the Ansari–Bradley two–sample test for a difference in scale parameters. The test returns for any sizes of the series `x` and `y` the exact `p` value together with its asymptotic limit.

The `method="mood"`, is another test which performs a two–sample test for a difference in scale parameters. The underlying model is that the two samples are drawn from  $f(x-l)$  and  $f((x-l)/s)/s$ , respectively, where  $l$  is a common location parameter and  $s$  is a scale parameter. The null hypothesis is  $s=l$ .

**Value**

In contrast to R's output report from S3 objects of class "htest" a different output report is produced. The classical tests presented here return an S4 object of class "fHTEST". The object contains the following slots:

<code>@call</code>	the function call.
<code>@data</code>	the data as specified by the input argument(s).
<code>@test</code>	a list whose elements contain the results from the statistical test. The information provided is similar to a list object of class "htest".
<code>@title</code>	a character string with the name of the test. This can be overwritten specifying a user defined input argument.
<code>@description</code>	a character string with an optional user defined description. By default just the current date when the test was applied will be returned.

The slot `@test` returns an object of class "list" containing (at least) the following elements:

<code>statistic</code>	the value(s) of the test statistic.
<code>p.value</code>	the p-value(s) of the test.
<code>parameters</code>	a numeric value or vector of parameters.
<code>estimate</code>	a numeric value or vector of sample estimates.
<code>conf.int</code>	a numeric two row vector or matrix of 95
<code>method</code>	a character string indicating what type of test was performed.
<code>data.name</code>	a character string giving the name(s) of the data.

**Note**

Some of the test implementations are selected from R's ctest package.

**Author(s)**

R-core team for hypothesis tests implemented from R's package ctest.

**References**

- Conover, W. J. (1971); *Practical nonparametric statistics*, New York: John Wiley & Sons.  
 Lehmann E.L. (1986); *Testing Statistical Hypotheses*, John Wiley and Sons, New York.  
 Moore, D.S. (1986); *Tests of the chi-squared type*, In: D'Agostino, R.B. and Stephens, M.A., eds.,  
 Goodness-of-Fit Techniques, Marcel Dekker, New York.

**Examples**

```
## rnorm -
# Generate Series:
x = rnorm(50)
y = rnorm(50)

## scaleTest -
scaleTest(x, y, "ansari")
scaleTest(x, y, "mood")
```

---

ScalingLawPlot

*Scaling Law Behaviour*


---

**Description**

Evaluates the scaling exponent of a financial return series and plots the scaling law.

**Usage**

```
scalinglawPlot(x, span = ceiling(log(length(x)/252)/log(2)), doplot = TRUE,
  labels = TRUE, trace = TRUE, ...)
```

**Arguments**

- |        |   |
|--------|---|
| doplot | a logical value. Should a plot be displayed?  |
| labels | a logical value. Whether or not x- and y-axes should be automatically labeled and a default main title should be added to the plot. By default TRUE.  |
| span   | an integer value, determines for the qqgaussPlot the plot range, by default 5, and for the scalingPlot a reasonable number of of points for the scaling range, by default daily data with 252 business days per year are assumed. |
| trace  | a logical value. Should the computation be traced?  |

x an uni- or multivariate return series of class `timeSeries` or any other object which can be transformed by the function `as.timeSeries()` into an object of class `timeSeries`.

... arguments to be passed.

## Details

### Scaling Behavior:

The function `scalingPlot` plots the scaling law of financial time series under aggregation and returns an estimate for the scaling exponent. The scaling behavior is a very striking effect of the foreign exchange market and also other markets expressing a regular structure for the volatility. Considering the average absolute return over individual data periods one finds a scaling power law which relates the mean volatility over given time intervals to the size of these intervals. The power law is in many cases valid over several orders of magnitude in time. Its exponent usually deviates significantly from a Gaussian random walk model which implies  $1/2$ .

## Value

returns a list with the following components: `Intercept`, `Exponent` the scaling exponent, and `InverseExponent` its inverse value.

## Author(s)

Diethelm Wuertz for the Rmetrics R-port.

## References

Taylor S.J. (1986); *Modeling Financial Time Series*, John Wiley and Sons, Chichester.

## Examples

```
## data -
  SPI <- LPP2005REC[, "SPI"]
  plot(SPI, type = "l", col = "steelblue", main = "SP500")
  abline(h = 0, col = "grey")

## teffectPlot -
  # Scaling Law Effect:
  scalinglawPlot(SPI)
```

sgh

*Standardized Generalized Hyperbolic Distribution***Description**

Density, distribution function, quantile function and random generation for the standardized generalized hyperbolic distribution.

**Usage**

```
dsgh(x, zeta = 1, rho = 0, lambda = 1, log = FALSE)
psgh(q, zeta = 1, rho = 0, lambda = 1)
qsgh(p, zeta = 1, rho = 0, lambda = 1)
rsgh(n, zeta = 1, rho = 0, lambda = 1)
```

**Arguments**

zeta, rho, lambda	shape parameter zeta is positive, skewness parameter rho is in the range (-1, 1).
log	a logical flag by default FALSE. If TRUE, log values are returned.
n	number of observations.
p	a numeric vector of probabilities.
x, q	a numeric vector of quantiles.

**Details**

The generator rsgh is based on the GH algorithm given by Scott (2004).

**Value**

All values for the \*sgh functions are numeric vectors: d\* returns the density, p\* returns the distribution function, q\* returns the quantile function, and r\* generates random deviates.

All values have attributes named "param" listing the values of the distributional parameters.

**Author(s)**

Diethelm Wuertz.

**Examples**

```
## rsgh -
set.seed(1953)
r = rsgh(5000, zeta = 1, rho = 0.5, lambda = 1)
plot(r, type = "l", col = "steelblue",
      main = "gh: zeta=1 rho=0.5 lambda=1")
```

```
## dsgh -
# Plot empirical density and compare with true density:
hist(r, n = 50, probability = TRUE, border = "white", col = "steelblue",
     ylim = c(0, 0.6))
x = seq(-5, 5, length = 501)
lines(x, dsgh(x, zeta = 1, rho = 0.5, lambda = 1))

## psgh -
# Plot df and compare with true df:
plot(sort(r), (1:5000/5000), main = "Probability", col = "steelblue")
lines(x, psgh(x, zeta = 1, rho = 0.5, lambda = 1))

## qsgh -
# Compute Quantiles:
round(qsgh(psgh(seq(-5, 5, 1), zeta = 1, rho = 0.5), zeta = 1, rho = 0.5), 4)
```

---

sghFit

*Standardized GH Distribution Fit*


---

## Description

Estimates the distributional parameters for a standardized generalized hyperbolic distribution.

## Usage

```
sghFit(x, zeta = 1, rho = 0, lambda = 1, include.lambda = TRUE,
       scale = TRUE, doplot = TRUE, span = "auto", trace = TRUE,
       title = NULL, description = NULL, ...)
```

## Arguments

x	a numeric vector.
zeta, rho, lambda	shape parameter zeta is positive, skewness parameter rho is in the range (-1, 1). and index parameter lambda, by default 1.
include.lambda	a logical flag, by default TRUE. Should the index parameter lambda included in the parameter estimate?
scale	a logical flag, by default TRUE. Should the time series be scaled by its standard deviation to achieve a more stable optimization?
doplot	a logical flag. Should a plot be displayed?
span	x-coordinates for the plot, by default 100 values automatically selected and ranging between the 0.001, and 0.999 quantiles. Alternatively, you can specify the range by an expression like span=seq(min, max, times = n), where, min and max are the left and right endpoints of the range, and n gives the number of the intermediate points.
trace	a logical flag. Should the parameter estimation process be traced?

title            a character string which allows for a project title.  
 description     a character string which allows for a brief description.  
 ...              parameters to be parsed.

### Value

returns a list with the following components:

estimate        the point at which the maximum value of the log likelihood function is obtained.  
 minimum        the value of the estimated maximum, i.e. the value of the log likelihood function.  
 code            an integer indicating why the optimization process terminated.  
                  1: relative gradient is close to zero, current iterate is probably solution;  
                  2: successive iterates within tolerance, current iterate is probably solution;  
                  3: last global step failed to locate a point lower than estimate. Either estimate  
                  is an approximate local minimum of the function or `steptol` is too small;  
                  4: iteration limit exceeded;  
                  5: maximum step size `stepmax` exceeded five consecutive times. Either the  
                  function is unbounded below, becomes asymptotic to a finite value from above  
                  in some direction or `stepmax` is too small.  
 gradient        the gradient at the estimated maximum.  
 steps            number of function calls.

### Examples

```
## sghFit -
# Simulate Random Variates:
set.seed(1953)
s = rsgh(n = 2000, zeta = 0.7, rho = 0.5, lambda = 0)

## sghFit -
# Fit Parameters:
sghFit(s, zeta = 1, rho = 0, lambda = 1, include.lambda = TRUE,
       doplot = TRUE)
```

---

sght

*Standardized generalized hyperbolic Student-t Distribution*

---

### Description

Density, distribution function, quantile function and random generation for the standardized generalized hyperbolic distribution.

### Usage

```
dsght(x, beta = 0.1, delta = 1, mu = 0, nu = 10, log = FALSE)
psght(q, beta = 0.1, delta = 1, mu = 0, nu = 10)
qsght(p, beta = 0.1, delta = 1, mu = 0, nu = 10)
rsght(n, beta = 0.1, delta = 1, mu = 0, nu = 10)
```

**Arguments**

beta, delta, mu	numeric values. beta is the skewness parameter in the range (0, alpha); delta is the scale parameter, must be zero or positive; mu is the location parameter, by default 0. These are the parameters in the first parameterization.
nu	a numeric value, the number of degrees of freedom. Note, alpha takes the limit of abs(beta), and lambda=-nu/2.
x, q	a numeric vector of quantiles.
p	a numeric vector of probabilities.
n	number of observations.
log	a logical, if TRUE, probabilities p are given as log(p).

**Value**

All values for the \*sght functions are numeric vectors: d\* returns the density, p\* returns the distribution function, q\* returns the quantile function, and r\* generates random deviates.

All values have attributes named "param" listing the values of the distributional parameters.

**Author(s)**

Diethelm Wuertz.

**Examples**

```
## rsght -
set.seed(1953)
r = rsght(5000, beta = 0.1, delta = 1, mu = 0, nu = 10)
plot(r, type = "l", col = "steelblue",
     main = "gh: zeta=1 rho=0.5 lambda=1")

## dsght -
# Plot empirical density and compare with true density:
hist(r, n = 50, probability = TRUE, border = "white", col = "steelblue")
x = seq(-5, 5, length = 501)
lines(x, dsght(x, beta = 0.1, delta = 1, mu = 0, nu = 10))

## psght -
# Plot df and compare with true df:
plot(sort(r), (1:5000/5000), main = "Probability", col = "steelblue")
lines(x, psght(x, beta = 0.1, delta = 1, mu = 0, nu = 10))

## qsght -
# Compute Quantiles:
round(qsght(psght(seq(-5, 5, 1), beta = 0.1, delta = 1, mu = 0, nu = 10),
             beta = 0.1, delta = 1, mu = 0, nu = 10), 4)
```

---

 snig

---

*Standardized Normal Inverse Gaussian Distribution*


---

**Description**

Density, distribution function, quantile function and random generation for the standardized normal inverse Gaussian distribution.

**Usage**

```
dsnig(x, zeta = 1, rho = 0, log = FALSE)
psnig(q, zeta = 1, rho = 0)
qsnig(p, zeta = 1, rho = 0)
rsnig(n, zeta = 1, rho = 0)
```

**Arguments**

zeta, rho	shape parameter zeta is positive, skewness parameter rho is in the range (-1, 1).
log	a logical flag by default FALSE. If TRUE, log values are returned.
n	number of observations.
p	a numeric vector of probabilities.
x, q	a numeric vector of quantiles.

**Details**

The random deviates are calculated with the method described by Raible (2000).

**Value**

All values for the \*snig functions are numeric vectors: d\* returns the density, p\* returns the distribution function, q\* returns the quantile function, and r\* generates random deviates.

All values have attributes named "param" listing the values of the distributional parameters.

**Author(s)**

Diethelm Wuertz.

**Examples**

```
## snig -
  set.seed(1953)
  r = rsnig(5000, zeta = 1, rho = 0.5)
  plot(r, type = "l", col = "steelblue",
       main = "snig: zeta=1 rho=0.5")

## snig -
```



```

# Plot empirical density and compare with true density:
hist(r, n = 50, probability = TRUE, border = "white", col = "steelblue")
x = seq(-5, 5, length = 501)
lines(x, dsnig(x, zeta = 1, rho = 0.5))

## snig -
# Plot df and compare with true df:
plot(sort(r), (1:5000/5000), main = "Probability", col = "steelblue")
lines(x, psnig(x, zeta = 1, rho = 0.5))

## snig -
# Compute Quantiles:
qsnig(psnig(seq(-5, 5, 1), zeta = 1, rho = 0.5), zeta = 1, rho = 0.5)

```

---

snigFit

*Fit of a Standardized NIG Distribution*


---

## Description

Estimates the parameters of a standardized normal inverse Gaussian distribution.

## Usage

```

snigFit(x, zeta = 1, rho = 0, scale = TRUE, doplot = TRUE,
        span = "auto", trace = TRUE, title = NULL, description = NULL, ...)

```

## Arguments

zeta, rho	shape parameter zeta is positive, skewness parameter rho is in the range (-1, 1).
description	a character string which allows for a brief description.
doplot	a logical flag. Should a plot be displayed?
scale	a logical flag, by default TRUE. Should the time series be scaled by its standard deviation to achieve a more stable optimization?
span	x-coordinates for the plot, by default 100 values automatically selected and ranging between the 0.001, and 0.999 quantiles. Alternatively, you can specify the range by an expression like <code>span=seq(min, max, times = n)</code> , where, min and max are the left and right endpoints of the range, and n gives the number of the intermediate points.
title	a character string which allows for a project title.
trace	a logical flag. Should the parameter estimation process be traced?
x	a numeric vector.
...	parameters to be parsed.

**Value**

The function `snigFit` returns a list with the following components:

<code>estimate</code>	the point at which the maximum value of the log likelihood function is obtained.
<code>minimum</code>	the value of the estimated maximum, i.e. the value of the log likelihood function.
<code>code</code>	an integer indicating why the optimization process terminated. 1: relative gradient is close to zero, current iterate is probably solution; 2: successive iterates within tolerance, current iterate is probably solution; 3: last global step failed to locate a point lower than <code>estimate</code> . Either <code>estimate</code> is an approximate local minimum of the function or <code>steptol</code> is too small; 4: iteration limit exceeded; 5: maximum step size <code>stepmax</code> exceeded five consecutive times. Either the function is unbounded below, becomes asymptotic to a finite value from above in some direction or <code>stepmax</code> is too small.
<code>gradient</code>	the gradient at the estimated maximum.
<code>steps</code>	number of function calls.

**Examples**

```
## snigFit -
# Simulate Random Variates:
set.seed(1953)
s = rsnig(n = 2000, zeta = 0.7, rho = 0.5)

## snigFit -
# Fit Parameters:
snigFit(s, zeta = 1, rho = 0, doplot = TRUE)
```

---

ssd

*Spline Smoothed Distribution*


---

**Description**

Density, distribution function, quantile function and random generation from smoothing spline estimates.

**Usage**

```
dssd(x, param, log = FALSE)
pssd(q, param)
qssd(p, param)
rssd(n, param)
```

**Arguments**

param	an object as returned by the function <code>ssdFit</code> .
log	a logical flag by default FALSE. Should labels and a main title drawn to the plot?
n	number of observations.
p	a numeric vector of probabilities.
x, q	a numeric vector of quantiles.

**Value**

All values for the `*ssd` functions are numeric vectors: `d*` returns the density, `p*` returns the distribution function, `q*` returns the quantile function, and `r*` generates random deviates.

All values have attributes named "param" listing the values of the distributional parameters.

**Author(s)**

Diethelm Wuertz, Chong Gu for the underlying `gss` package.

**References**

Gu, C. (2002), *Smoothing Spline ANOVA Models*, New York Springer-Verlag.

Gu, C. and Wang, J. (2003), *Penalized likelihood density estimation: Direct cross-validation and scalable approximation*, *Statistica Sinica*, 13, 811–826.

**Examples**

```
## ssdFit -
  set.seed(1953)
  r = rnorm(500)
  hist(r, breaks = "FD", probability = TRUE,
       col = "steelblue", border = "white")

## ssdFit -
  param = ssdFit(r)

## dssd -
  u = seq(min(r), max(r), len = 301)
  v = dssd(u, param)
  lines(u, v, col = "orange", lwd = 2)
```

---

`ssdFit`*Fit Density Using Smoothing Splines*

---

**Description**

Estimates the parameters of a density function using smoothing splines.

**Usage**

```
ssdFit(x)
```

**Arguments**

`x` a numeric vector.

**Value**

The function `ssdFit`, `hypFit` returns an object of class `ssden`. The returned object can be used to evaluate density, probabilities and quantiles.

**Author(s)**

Diethelm Wuertz, Chong Gu for the underlying `gss` package.

**References**

Gu, C. (2002), *Smoothing Spline ANOVA Models*, New York Springer-Verlag.

Gu, C. and Wang, J. (2003), *Penalized likelihood density estimation: Direct cross-validation and scalable approximation*, *Statistica Sinica*, 13, 811–826.

**Examples**

```
## ssdFit -
  set.seed(1953)
  r = rnorm(500)
  hist(r, breaks = "FD", probability = TRUE,
       col = "steelblue", border = "white")

## ssdFit -
  param = ssdFit(r)

## dssd -
  u = seq(min(r), max(r), len = 301)
  v = dssd(u, param)
  lines(u, v, col = "orange", lwd = 2)
```

---

`StableSlider`*Slider GUI for Stable Distribution*

---

**Description**

The `stableSlider()` function provides interactive displays of density and probabilities of stable distributions.

**Usage**

```
stableSlider(col= "steelblue", col.med = "gray30")
```

**Arguments**

`col`, `col.med` optional arguments for the slider.

**Value**

The `stableSlider()` function displays densities and probabilities of the skew stable distribution, for educational purposes.

**Author(s)**

Diethelm Wuertz for the Rmetrics R-port.

**References**

see those in [dstable](#), in package **stabledist**.

**Examples**

```
if(dev.interactive())
  stableSlider()
```

---

`symbolTable`*Table of Symbols*

---

**Description**

Displays a Table of plot characters and symbols.

**Usage**

```
symbolTable(font = par('font'), cex = 0.7)
```

**Arguments**

`cex` a numeric value, determines the character size, the default size is 0.7.  
`font` an integer value, the number of the font, by default font number 1.

**Value**

`symbolTable`

displays a table with the plot characters and symbols numbered from 0 to 255 and returns invisible the name of the font.

**See Also**

`link{characterTable}`, `link{colorTable}`.

**Examples**

```
## symbolTable -
# Default Symbol Table:
symbolTable()
```

---

TimeSeriesPlots

*Financial Time Series Plots*


---

**Description**

Returns an index/price, a return, or a drawdown plot.

List of Functions:

<code>seriesPlot</code>	Returns a tailored return series plot,
<code>cumulatedPlot</code>	Displays a cumulated series given the returns,
<code>returnPlot</code>	Displays returns given the cumulated series,
<code>drawdownPlot</code>	Displays drawdowns given the return series.

**Usage**

```
seriesPlot(x, labels = TRUE, type = "l", col = "steelblue",
           title = TRUE, grid = TRUE, box = TRUE, rug = TRUE, ...)
cumulatedPlot(x, index = 100, labels = TRUE, type = "l", col = "steelblue",
              title = TRUE, grid = TRUE, box = TRUE, rug = TRUE, ...)
returnPlot(x, labels = TRUE, type = "l", col = "steelblue",
           title = TRUE, grid = TRUE, box = TRUE, rug = TRUE, ...)
drawdownPlot(x, labels = TRUE, type = "l", col = "steelblue",
             title = TRUE, grid = TRUE, box = TRUE, rug = TRUE, ...)
```

**Arguments**

<code>box</code>	a logical flag, should a box be added to the plot? By default TRUE.
<code>col</code>	the color for the series. In the univariate case use just a color name like the default, <code>col="steelblue"</code> , in the multivariate case we recommend to select the colors from a color palette, e.g. <code>col=heat.colors(ncol(x))</code> .
<code>grid</code>	a logical flag, should a grid be added to the plot? By default TRUE.
<code>index</code>	a numeric value, by default 100. The function cumulates column by column the returns and multiplies the result with the index value: <code>index*exp(colCumsums(x))</code> .
<code>labels</code>	a logical flag, should the plot be returned with default labels and decorated in an automated way? By default TRUE.
<code>rug</code>	a logical flag, by default TRUE. Should a rug representation of the data added to the plot?
<code>title</code>	a logical flag, by default TRUE. Should a default title added to the plot?
<code>type</code>	what type of plot should be drawn? By default we use a line plot, <code>type="l"</code> . An alternative plot style which produces nice figures is for example <code>type="h"</code> .
<code>x</code>	an object of class "timeSeries" or any other object which can be transformed by the function <code>as.timeSeries</code> into an object of class <code>timeSeries</code> . The latter case, other than <code>timeSeries</code> objects, is more or less untested.
<code>...</code>	optional arguments to be passed.

**Details**

The plot functions can be used to plot univariate and multivariate time series of class `timeSeries`.

The graphical parameters `type` and `col` can be set by the values specified through the argument list. In the case of multivariate time series `col` can be specified by the values returned by a color palette.

Automated titles including main title, x- and y-labels, grid lines, box style and rug representations can be selected by setting these arguments to TRUE which is the default. If the title flag is unset, then the main title, x-, and y-labels are empty strings. This allows to set user defined labels with the function `title` after the plot is drawn.

Beside `type`, `col`, `main`, `xlab` and `ylab`, all other `par` arguments can be passed to the `plot` function.

If the `labels` flag is unset to FALSE, then no decorations will be added to the plot, and the plot can be fully decorated by the user.

**Value**

displays a time series plot.

**Examples**

```
## seriesPlot -
tS <- as.timeSeries(data(LPP2005REC))
seriesPlot(tS)
```

tr *Trace of a Matrix*

---

**Description**

Returns trace of a matrix.

**Usage**

```
tr(x)
```

**Arguments**

x a numeric matrix.

**Details**

The function `tr` computes the trace of a square matrix which is the sum of the diagonal elements of the matrix under consideration.

**References**

Golub, van Loan, (1996); *Matrix Computations*, 3rd edition. Johns Hopkins University Press.

**Examples**

```
## Create Pascal Matrix:  
P = pascal(3)  
P  
  
## Trace:  
tr(P)
```

---

triang *Upper and Lower Triangular Matrixes*

---

**Description**

Extracts the upper or lower tridiagonal part from a matrix.

**Usage**

```
triang(x)  
Triang(x)
```



**Arguments**

x                    a numeric matrix.

**Details**

The functions `triang` and `Triang` allow to transform a square matrix to a lower or upper triangular form. A triangular matrix is either an upper triangular matrix or lower triangular matrix. For the first case all matrix elements  $a[i, j]$  of matrix  $A$  are zero for  $i > j$ , whereas in the second case we have just the opposite situation. A lower triangular matrix is sometimes also called left triangular. In fact, triangular matrices are so useful that much computational linear algebra begins with factoring or decomposing a general matrix or matrices into triangular form. Some matrix factorization methods are the Cholesky factorization and the LU-factorization. Even including the factorization step, enough later operations are typically avoided to yield an overall time savings. Triangular matrices have the following properties: the inverse of a triangular matrix is a triangular matrix, the product of two triangular matrices is a triangular matrix, the determinant of a triangular matrix is the product of the diagonal elements, the eigenvalues of a triangular matrix are the diagonal elements.

**References**

Higham, N.J., (2002); *Accuracy and Stability of Numerical Algorithms*, 2nd ed., SIAM.

Golub, van Loan, (1996); *Matrix Computations*, 3rd edition. Johns Hopkins University Press.

**Examples**

```
## Create Pascal Matrix:
P = pascal(3)
P

## Create lower triangle matrix
L = triang(P)
L
```

---

tsHessian

*Two sided approximated Hessian*


---

**Description**

Computes two sided (TS) approximated Hessian.

**Usage**

```
tsHessian(x, fun, ...)
```

**Arguments**

x                    arugment to be passed to function  
fun                   function  
...                   additional paramters to be passed to function

**Author(s)**

A function borrowed from Kevin Sheppard's Matlab garch toolbox as implemented by Alexios Ghalanos in his **rgarch** package

---

tslag	<i>Lagged or Leading Vector/Matrix</i>
-------	--

---

**Description**

Creates a lagged or leading vector/matrix of selected order(s).

**Usage**

```
tslag(x, k = 1, trim = FALSE)
```

**Arguments**

k	an integer value, the number of positions the new series is to lag or to lead the input series.
x	a numeric vector or matrix, missing values are allowed.
trim	a logical flag, if TRUE, the missing values at the beginning and/or end of the returned series will be trimmed. The default value is FALSE.

**See Also**

[pdl](#).

**Examples**

```
## tslag -  
#
```

---

varianceTest	<i>Two Sample Variance Tests</i>
--------------	----------------------------------

---

**Description**

Tests if two series differ in their distributional variance parameter.

**Usage**

```
varianceTest(x, y, method = c("varf", "bartlett", "fligner"),  
            title = NULL, description = NULL)
```

**Arguments**

x, y	numeric vectors of data values.
method	a character string naming which test should be applied.
title	an optional title string, if not specified the inputs data name is deparsed.
description	optional description string, or a vector of character strings.

**Details**

The method="varf" can be used to compare variances of two normal samples performing an F test. The null hypothesis is that the ratio of the variances of the populations from which they were drawn is equal to one.

The method="bartlett" performs the Bartlett test of the null hypothesis that the variances in each of the samples are the same. This fact of equal variances across samples is also called *homogeneity of variances*. Note, that Bartlett's test is sensitive to departures from normality. That is, if the samples come from non-normal distributions, then Bartlett's test may simply be testing for non-normality. The Levene test (not yet implemented) is an alternative to the Bartlett test that is less sensitive to departures from normality.

The method="fligner" performs the Fligner-Killeen test of the null that the variances in each of the two samples are the same.

**Value**

In contrast to R's output report from S3 objects of class "htest" a different output report is produced. The classical tests presented here return an S4 object of class "fHTEST". The object contains the following slots:

@call	the function call.
@data	the data as specified by the input argument(s).
@test	a list whose elements contain the results from the statistical test. The information provided is similar to a list object of class "htest".
@title	a character string with the name of the test. This can be overwritten specifying a user defined input argument.
@description	a character string with an optional user defined description. By default just the current date when the test was applied will be returned.

The slot @test returns an object of class "list" containing (at least) the following elements:

statistic	the value(s) of the test statistic.
p.value	the p-value(s) of the test.
parameters	a numeric value or vector of parameters.
estimate	a numeric value or vector of sample estimates.
conf.int	a numeric two row vector or matrix of 95
method	a character string indicating what type of test was performed.
data.name	a character string giving the name(s) of the data.

**Note**

Some of the test implementations are selected from R's ctest package.

**Author(s)**

R-core team for hypothesis tests implemented from R's package ctest.

**References**

Conover, W. J. (1971); *Practical nonparametric statistics*, New York: John Wiley & Sons.

Lehmann E.L. (1986); *Testing Statistical Hypotheses*, John Wiley and Sons, New York.

**Examples**

```
## rnorm -  
# Generate Series:  
x = rnorm(50)  
y = rnorm(50)  
  
## varianceTest -  
varianceTest(x, y, "varf")  
varianceTest(x, y, "bartlett")  
varianceTest(x, y, "fligner")
```

---

vec

*Stacking Vectors and Matrixes*

---

**Description**

Stacks either a lower triangle matrix or a matrix.

**Usage**

```
vec(x)  
vech(x)
```

**Arguments**

x                    a numeric matrix.

**Details**

The function `vec` implements the operator that stacks a matrix as a column vector, to be more precise in a matrix with one column.  $\text{vec}(X) = (X_{11}, X_{21}, \dots, X_{N1}, X_{12}, X_{22}, \dots, X_{NN})$ .

The function `vech` implements the operator that stacks the lower triangle of a  $N \times N$  matrix as an  $N(N+1)/2 \times 1$  vector:  $\text{vech}(X) = (X_{11}, X_{21}, X_{22}, X_{31}, \dots, X_{NN})$ , to be more precise in a matrix with one row.

**Examples**

```
## Create Pascal Matrix:  
P = pascal(3)  
  
## Stack a matrix  
vec(P)  
  
## Stack the lower triangle  
vech(P)
```

# Index

- \* **datasets**
  - fBasicsData, 32
- \* **distribution**
  - distCheck, 29
  - DistributionFits, 29
  - gh, 36
  - ghFit, 38
  - ghMode, 40
  - ghMoments, 41
  - ghRobMoments, 42
  - ghSlider, 43
  - ght, 44
  - ghtFit, 45
  - ghtMode, 46
  - ghtMoments, 47
  - ghtRobMoments, 48
  - gld, 49
  - gldFit, 51
  - gldRobMoments, 53
  - hyp, 59
  - hypFit, 61
  - hypMode, 62
  - hypMoments, 63
  - hypRobMoments, 65
  - hypSlider, 66
  - maxdd, 79
  - nig, 81
  - nigFit, 82
  - nigMode, 84
  - nigMoments, 85
  - nigRobMoments, 86
  - nigShapeTriangle, 87
  - nigSlider, 88
  - normRobMoments, 94
  - sampleLMoments, 103
  - sampleRobMoments, 103
  - sgh, 108
  - sghFit, 109
  - sght, 110
  - snig, 112
  - snigFit, 113
  - ssd, 114
  - ssdFit, 116
  - StableSlider, 117
- \* **hplot**
  - acfPlot, 13
  - decor, 28
  - gridVector, 54
  - interactivePlot, 67
  - ScalingLawPlot, 106
- \* **htest**
  - correlationTest, 26
  - fHTEST, 34
  - ks2Test, 71
  - locationTest, 77
  - NormalityTests, 90
  - scaleTest, 104
  - varianceTest, 122
- \* **math**
  - colVec, 25
  - hilbert, 56
  - Ids, 66
  - inv, 68
  - kron, 70
  - norm, 89
  - pascal, 95
  - pdl, 96
  - positiveDefinite, 97
  - rk, 100
  - tr, 120
  - triang, 120
  - tslag, 122
  - vec, 124
- \* **misc**
  - fBasics-deprecated, 31
- \* **package**
  - fBasics-package, 4
- \* **programming**

- akimaInterp, 15
- baseMethods, 17
- BasicStatistics, 18
- BoxPlot, 19
- characterTable, 20
- colorLocator, 20
- colorPalette, 21
- colorTable, 25
- getS4, 35
- Heaviside, 55
- HistogramPlot, 57
- krigeInterp, 69
- lcg, 72
- linearInterp, 74
- listDescription, 75
- listFunctions, 76
- listIndex, 76
- print, 97
- QuantileQuantilePlots, 98
- ReturnSeriesGUI, 100
- symbolTable, 117
- TimeSeriesPlots, 118
- \* **univar**
  - rowStats, 101
  - .HedgeFund1 (fBasicsData), 32
  - .HedgeFund2 (fBasicsData), 32
- acf, 14
- acfPlot, 13
- adTest (NormalityTests), 90
- akimaInterp, 15, 70, 74
- akimaInterpp (akimaInterp), 15
- baseMethods, 17
- BasicStatistics, 18
- basicStats (BasicStatistics), 18
- box\_ (decor), 28
- Boxcar (Heaviside), 55
- boxL (decor), 28
- boxPercentilePlot (BoxPlot), 19
- BoxPlot, 19
- boxPlot (BoxPlot), 19
- Capitalization (fBasicsData), 32
- cars2 (fBasicsData), 32
- characterTable, 20
- cmPalette (colorPalette), 21
- colIds (Ids), 66
- colIds<- (Ids), 66
- colorLocator, 20
- colorMatrix (colorLocator), 20
- colorPalette, 21, 21
- colorTable, 21, 25
- colVec, 25
- contour, 69
- copyright (decor), 28
- correlationTest, 26
- countFunctions (listFunctions), 76
- cumulatedPlot (TimeSeriesPlots), 118
- cvmTest (NormalityTests), 90
- dagoTest (NormalityTests), 90
- decor, 28
- Defunct, 31
- Delta (Heaviside), 55
- densityPlot (HistogramPlot), 57
- Deprecated, 31
- dgh (gh), 36
- dght (ght), 44
- dgld (gld), 49
- dhyp (hyp), 59
- distCheck, 29
- DistributionFits, 29
- divPalette (colorPalette), 21
- dmaxdd (maxdd), 79
- dnig (nig), 81
- DowJones30 (fBasicsData), 32
- drawdownPlot (TimeSeriesPlots), 118
- dsgh (sgh), 108
- dsght (sght), 110
- dsnig (snig), 112
- dssd (ssd), 114
- dstable, 31, 117
- expand.grid, 54
- fBasics (fBasics-package), 4
- fBasics-deprecated, 31
- fBasics-package, 4
- fBasicsData, 32
- fDISTFIT (DistributionFits), 29
- fDISTFIT-class (DistributionFits), 29
- fHTEST, 34
- fHTEST-class (fHTEST), 34
- focusPalette (colorPalette), 21
- get.lcgseed (lcg), 72
- getArgs (getS4), 35

- getCall (getS4), 35
- getCall, ANY-method (getS4), 35
- getDescription (getS4), 35
- getModel (getS4), 35
- getS4, 35
- getSlot (getS4), 35
- getTitle (getS4), 35
- gh, 36
- ghFit, 38
- ghIQR (ghRobMoments), 42
- ghKURT (ghRobMoments), 42
- ghKurt (ghMoments), 41
- ghMean (ghMoments), 41
- ghMED (ghRobMoments), 42
- ghMode, 40
- ghMoments, 41
- ghRobMoments, 42
- ghSKEW (ghRobMoments), 42
- ghSkew (ghMoments), 41
- ghSlider, 43
- ght, 44
- ghtFit, 45
- ghtIQR (ghtRobMoments), 48
- ghtKURT (ghtRobMoments), 48
- ghtKurt (ghtMoments), 47
- ghtMean (ghtMoments), 47
- ghtMED (ghtRobMoments), 48
- ghtMode, 46
- ghtMoments, 47
- ghtRobMoments, 48
- ghtSKEW (ghtRobMoments), 48
- ghtSkew (ghtMoments), 47
- ghtVar (ghtMoments), 47
- ghVar (ghMoments), 41
- gld, 49
- gldFit, 51
- gldIQR (gldRobMoments), 53
- gldKURT (gldRobMoments), 53
- gldMED (gldRobMoments), 53
- gldMode, 52
- gldRobMoments, 53
- gldSKEW (gldRobMoments), 53
- greyPalette (colorPalette), 21
- gridVector, 54
  
- heatPalette (colorPalette), 21
- Heaviside, 55
- HedgeFund (fBasicsData), 32
- hgrid (decor), 28
  
- hilbert, 56
- HistogramPlot, 57
- histPlot (HistogramPlot), 57
- hyp, 59
- hypFit, 61
- hypIQR (hypRobMoments), 65
- hypKURT (hypRobMoments), 65
- hypKurt (hypMoments), 63
- hypMean (hypMoments), 63
- hypMED (hypRobMoments), 65
- hypMode, 62
- hypMoments, 63
- hypRobMoments, 65
- hypSKEW (hypRobMoments), 65
- hypSkew (hypMoments), 63
- hypSlider, 66
- hypVar (hypMoments), 63
  
- Ids, 66
- interactivePlot, 67
- inv, 68
- isPositiveDefinite (positiveDefinite), 97
  
- jarqueberaTest (NormalityTests), 90
- jbTest (NormalityTests), 90
  
- kendallTest (correlationTest), 26
- krigeInterp, 16, 69, 74
- kron, 70
- ks2Test, 71
- ksnormTest (NormalityTests), 90
  
- lacfPlot (acfPlot), 13
- lcg, 72
- lillieTest (NormalityTests), 90
- linearInterp, 16, 70, 74
- linearInterpp (linearInterp), 74
- listDescription, 75, 77
- listFunctions, 75, 76, 76
- listIndex, 75, 76, 76, 77
- locationTest, 27, 77
- locator, 21
- logDensityPlot (HistogramPlot), 57
  
- makePositiveDefinite (positiveDefinite), 97
- maxdd, 79
- maxddStats (maxdd), 79



- monoPalette (colorPalette), 21
- msft.dat (fBasicsData), 32
- nFit (DistributionFits), 29
- nig, 81
- nigFit, 82
- nigIQR (nigRobMoments), 86
- nigKURT (nigRobMoments), 86
- nigKurt (nigMoments), 85
- nigMean (nigMoments), 85
- nigMED (nigRobMoments), 86
- nigMode, 84
- nigMoments, 85
- nigRobMoments, 86
- nigShapeTriangle, 87
- nigSKEW (nigRobMoments), 86
- nigSkew (nigMoments), 85
- nigSlider, 88
- nigVar (nigMoments), 85
- nlm, 39, 45, 61
- nlminb, 51
- norm, 89
- norm2 (norm), 89
- NormalityTests, 90
- normalTest (NormalityTests), 90
- normIQR (normRobMoments), 94
- normKURT (normRobMoments), 94
- normMED (normRobMoments), 94
- normRobMoments, 94
- normSKEW (normRobMoments), 94
- nyse (fBasicsData), 32
- pacfPlot (acfPlot), 13
- pascal, 95
- pchiTest (NormalityTests), 90
- pdl, 96, 122
- pearsonTest (correlationTest), 26
- PensionFund (fBasicsData), 32
- persp, 69
- pgh (gh), 36
- pght (ght), 44
- pgld (gld), 49
- phyp (hyp), 59
- pmaxdd (maxdd), 79
- pnig (nig), 81
- positiveDefinite, 97
- print, 97
- print.fDISTFIT (DistributionFits), 29
- psgh (sgh), 108
- psght (sght), 110
- psnig (snig), 112
- pssd (ssd), 114
- qgh (gh), 36
- qght (ght), 44
- qgld (gld), 49
- qhyp (hyp), 59
- qnig (nig), 81
- qqghtPlot (QuantileQuantilePlots), 98
- qqgldPlot (QuantileQuantilePlots), 98
- qqnigPlot (QuantileQuantilePlots), 98
- qqnormPlot (QuantileQuantilePlots), 98
- qsg (sgh), 108
- qsgh (sgh), 108
- qsght (sght), 110
- qsnig (snig), 112
- qssd (ssd), 114
- qualiPalette (colorPalette), 21
- QuantileQuantilePlots, 98
- rainbowPalette (colorPalette), 21
- Ramp (Heaviside), 55
- rampPalette (colorPalette), 21
- returnPlot (TimeSeriesPlots), 118
- ReturnSeriesGUI, 100
- returnSeriesGUI (ReturnSeriesGUI), 100
- rgh (gh), 36
- rght (ght), 44
- rgld (gld), 49
- rhyp (hyp), 59
- rk, 100
- rmaxdd (maxdd), 79
- rnig (nig), 81
- rnorm.lcg (lcg), 72
- rowAves (rowStats), 101
- rowIds (Ids), 66
- rowIds<- (Ids), 66
- rowKurtosis (rowStats), 101
- rowMaxs (rowStats), 101
- rowMins (rowStats), 101
- rowProds (rowStats), 101
- rowQuantiles (rowStats), 101
- rowSds (rowStats), 101
- rowSkewness (rowStats), 101
- rowStats, 101
- rowStdevs (rowStats), 101
- rowVars (rowStats), 101
- rowVec (colVec), 25
- rsgh (sgh), 108

- rsght (sght), 110
- rsnig (snig), 112
- rssd (ssd), 114
- rt.lcg (lcg), 72
- runif.lcg (lcg), 72
  
- sampleIQR (sampleRobMoments), 103
- sampleKURT (sampleRobMoments), 103
- sampleLMoments, 103
- sampleLMoments (sampleLMoments), 103
- sampleMED (sampleRobMoments), 103
- sampleRobMoments, 103
- sampleSKEW (sampleRobMoments), 103
- scaleTest, 27, 104
- ScalingLawPlot, 106
- scalinglawPlot (ScalingLawPlot), 106
- seqPalette (colorPalette), 21
- seriesPlot (TimeSeriesPlots), 118
- set.lcgseed (lcg), 72
- sfTest (NormalityTests), 90
- sgh, 108
- sghFit, 109
- sght, 110
- shapiroTest (NormalityTests), 90
- show, fDISTFIT-method  
    (DistributionFits), 29
- show, fHTEST-method (fHTEST), 34
- Sign (Heaviside), 55
- snig, 112
- snigFit, 113
- spearmanTest (correlationTest), 26
- ssd, 114
- ssdFit, 116
- stableFit (DistributionFits), 29
- StableSlider, 117
- stableSlider (StableSlider), 117
- stdev (baseMethods), 17
- swissEconomy (fBasicsData), 32
- SWXLP (fBasicsData), 32
- symbolTable, 117
  
- teffectPlot (acfPlot), 13
- termPlot (baseMethods), 17
- terrainPalette (colorPalette), 21
- tFit (DistributionFits), 29
- TimeSeriesPlots, 118
- timPalette (colorPalette), 21
- topoPalette (colorPalette), 21
- tr, 120
  
- Triang (triang), 120
- triang, 120
- tsHessian, 121
- tslag, 96, 122
  
- usdthb (fBasicsData), 32
  
- varianceTest, 27, 122
- vec, 124
- vech (vec), 124
- vgrid (decor), 28
- volatility (baseMethods), 17