

Package ‘fUtilities’

September 29, 2009

Version 2100.77

Revision 4057

Date 2009-09-28

Title Function Utilities

Author Diethelm Wuertz and many others. See the SOURCE file

Depends R (>= 2.4.0), methods, MASS

Suggests tcltk, RUnit, akima, spatial, foreign

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

Description Environment for teaching “Financial Engineering and Computational Finance”

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

LazyLoad yes

LazyData yes

License GPL (>= 2)

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

Repository CRAN

Date/Publication 2009-09-29 18:44:37

R topics documented:

| | |
|--------------------|----|
| fUtilities-package | 3 |
| akimaInterp | 7 |
| as.matrix.ts | 9 |
| as.POSIXlt | 10 |
| baseMethods | 10 |
| characterTable | 14 |
| colorLocator | 15 |
| colorPalette | 16 |
| colorTable | 19 |
| colStats | 20 |
| colVec | 21 |
| decor | 22 |
| description | 23 |
| distCheck | 23 |
| fHTEST | 24 |
| getS4 | 25 |
| gridVector | 26 |
| Heaviside | 27 |
| hilbert | 29 |
| Ids | 30 |
| interactivePlot | 30 |
| inv | 31 |
| krigeInterp | 32 |
| kron | 33 |
| kurtosis | 34 |
| lcg | 35 |
| linearInterp | 37 |
| listDescription | 38 |
| listFunctions | 39 |
| listIndex | 39 |
| norm | 40 |
| pascal | 41 |
| pdl | 42 |
| positiveDefinite | 43 |
| print | 43 |
| rk | 44 |
| rowStats | 45 |
| skewness | 46 |
| symbolTable | 47 |
| tr | 48 |
| triang | 49 |
| tslag | 50 |
| vec | 51 |

fUtilities-package *Utilities and Tools Package*

Description

Package of basic utilities and general tools for Rmetrics.

Details

Package: fUtilities
 Type: Package
 Version: 261.73.1
 Date: 2008
 License: GPL Version 2 or later
 Copyright: (c) 1999-2008 Diethelm Wuertz and Rmetrics Foundation
 URL: <http://www.rmetrics.org>

Overview of Topics:

1. Basic Function Extensions
2. Column and Row Statistics for Rectangular Objects
3. Skewness and Kurtosis Statistics
4. Bivariate Interpolation and Kriging
5. Code Tables, Color Selection and Palettes
6. Vector/Matrix Arithmetics and Linear Algebra Addons
7. Special Functions Addon

1. Basic Function Extensions

Several functions are added by Rmetrics which are missing in R's basic packages.

The first set of these functions are concerned with R functions which were made generic, so that they could be used to add additional methods:

| | |
|------------|---------------------------|
| align | adds align function, |
| as.POSIXlt | adds POSIXlt function, |
| atoms | adds atoms function, |
| attach | extends attach function, |
| colnames<- | adds colnames assignment, |
| cor | extends cor function, |
| cov | extends cov function, |
| log | extends log function, |
| outlier | adds outlier function, |
| rank | extends rank function, |
| rownames<- | adds rownames assignment, |

| | |
|-------------------------|---------------------------|
| <code>sample</code> | extends sample function, |
| <code>stdev</code> | adds stdev function, |
| <code>termPlot</code> | adds term plot function, |
| <code>var</code> | extends var function, |
| <code>volatility</code> | adds volatility function. |

All these functions have now default methods. Furthermore the `fUtilities` package also provides additional methods. These include:

| | |
|----------------------------|---|
| <code>as.matrix.ts</code> | adds <code>as.matrix.ts</code> method, |
| <code>as.matrix.mts</code> | adds <code>as.matrix.mts</code> method, |
| <code>print.control</code> | adds <code>print.control</code> method. |

2. Statistical Function Extensions

Beside the basic function extensions also statistical function extensions are provided. This concerns the missing "skewness" and "kurtosis" functions in R

| | |
|-----------------------|----------------------------|
| <code>skewness</code> | returns value of skewness, |
| <code>kurtosis</code> | returns value of kurtosis. |

and functions for column and row statistics. The `colStats` and `rowStats` are quite general functions which allow to specify the function to compute the desired statistics by the user. The remaining column and row statistics functions are thought to compute often used time series statistics. This includes the `sum()`, mean, standard deviations, variance, skewness, kurtosis, maximum, minimum, product, and quantile value.

Column Statistics:

| | |
|---------------------------|--|
| <code>colStats</code> | calculates column statistics, |
| <code>colSums</code> | calculates column sums, |
| <code>colMeans</code> | calculates column means, |
| <code>colSds</code> | calculates column standard deviations, |
| <code>colVars</code> | calculates column variances, |
| <code>colSkewness</code> | calculates column skewness, |
| <code>colKurtosis</code> | calculates column kurtosis, |
| <code>colMaxs</code> | calculates maximum values in each column, |
| <code>colMins</code> | calculates minimum values in each column, |
| <code>colProds</code> | computes product of all values in each column, |
| <code>colQuantiles</code> | computes quantiles of each column. |

Row Statistics:

| | |
|--------------------------|-------------------------------------|
| <code>rowStats</code> | calculates row statistics, |
| <code>rowSums</code> | calculates row sums, |
| <code>rowMeans</code> | calculates row means, |
| <code>rowSds</code> | calculates row standard deviations, |
| <code>rowVars</code> | calculates row variances, |
| <code>rowSkewness</code> | calculates row skewness, |

| | |
|--------------|---|
| rowKurtosis | calculates row kurtosis, |
| rowMaxs | calculates maximum values in each row, |
| rowMins | calculates minimum values in each row, |
| rowProds | computes product of all values in each row, |
| rowQuantiles | computes quantiles of each row. |

For hypothesis testing Rmetrics offers a new S4 class and print method:

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

3. Graph and Plot Tools

Character, symbol and color tables are useful tools if one is concerned with graphs and charts:

| | |
|----------------|---|
| characterTable | Table of Numerical Equivalents to Latin Characters, |
| symbolTable | Table of plot characters, plot symbols, |
| colorTable | Table of Color Codes and Plot Colors itself, |
| colorLocator | Plots R's 657 named colors for selection, |
| colorMatrix | Returns matrix of R's color names. |

Many wrapper functions to create color palettes are also added, all following the same naming conventions:

| | |
|----------------|-----------------------------------|
| 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. |

An interactive plot function allows to create easily interactive plots:

| | |
|-----------------|--|
| interactivePlot | a framework for interactive plot displays. |
|-----------------|--|

4. Bivariate Interpolation and Kriging

Functions which allow to interpolate and smooth bivariate irregular data sets including linear interpolation, Akima spline interpolation, and kriging:

| | |
|--------------|---------------------------------------|
| linearInterp | performs linear spline interpolation, |
|--------------|---------------------------------------|

| | |
|-------------|--------------------------------------|
| akimaInterp | performs Akima spline interpolation, |
| krigeInterp | performs krige interpolation. |

4. Vector/Matrix Arithmetics and Linear Algebra Addons

Functions for matrix arithmetics and linear algebra. These functions are often very useful for the manipulation of the data slot of multivariate financial time series.

General Matrix Functions:

| | |
|----------------------|--|
| triang | Extracts the lower tridiagonal part from a matrix, |
| Triang | Extracts the upper tridiagonal part from a matrix, |
| pascal | Creates a Pascal matrix, |
| hilbert | Creates a Hilbert matrix, |
| colVec | Creates a column vector from a vector, |
| rowVec | Creates a row vector from a vector, |
| isPositiveDefinite | Checks if a matrix is positive definite, |
| makePositiveDefinite | Forces a matrix to be positive definite, |
| colIds | Retrieves or sets the colnames of an object, |
| rowIds | Retrieves or sets the rowumn names. |

Linear algebra functions in R's base package include the `%*%` product of two matrices, the `%x%` Kronecker product, the `det` determinant of a matrix, and the `t` transposed matrix.

Rmetrics adds the foloowing functions:

| | |
|------|---|
| inv | Returns the inverse of a matrix, |
| norm | Returns the norm of a matrix, |
| rk | Returns the rank of a matrix, |
| tr | Returns trace of a matrix, |
| vech | Is the operator that stacks the lower triangle, |
| vec | Is the operator that stacks a matrix. |

Note, additional linear algebra functionality is provided in R through the functions `chol` which returns the Cholesky factor matrix, `eigen` which computes eigenvalues and eigenvectors, `svd` which does singular value decomposition, `kappa` which determines the condition number of a matrix, `qr` which performs the QR decomposition of a matrix, `solve` which solves a system of linear equations, together with the functions `backsolve` used when the matrix is upper triangular, and `forwardsolve` used when the matrix is lower triangular.

6. Time Series Generation

For the computation of lagged or leading series the following two functions are provided by Rmetrics:

| | |
|-------|---|
| tslag | Lagged or leading vector/matrix of selected order(s), |
| pdl | Regressor matrix for polynomial distributed lags. |

7. Special Functions Addon

Functions which compute special functions missing in R's base package include:

Heaviside and Related Functions:

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

Generator for Portable Random Innovations:

| | |
|-------------|---|
| 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. |

8. Some utility functions

Finally we like to mention some further utility functions:

`gridVector` creates from two vectors `x` and `y` all grid points.

Author(s)

The `fUtilities` package was originally written by Diethelm Wuertz and is maintained since 2007 and further developed by him and the Rmetrics core team.

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

| | |
|--|---|
| <code>x</code> , <code>y</code> , <code>z</code> | for <code>akimaInterp</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. For <code>akimaInterpp</code> we consider either three numeric vectors of equal length or if <code>y</code> and <code>z</code> are <code>NULL</code> , a list with entries <code>x</code> , <code>y</code> , <code>z</code> , or named data frame with <code>x</code> in the first, <code>y</code> in the second, and <code>z</code> in the third column. |
| <code>gridPoints</code> | an integer value specifying the number of grid points in <code>x</code> and <code>y</code> direction. |
| <code>xo</code> , <code>yo</code> | for <code>akimaInterp</code> two numeric vectors of data points spanning the grid, and for <code>akimaInterpp</code> two numeric vectors of data points building pairs for pointwise interpolation. |
| <code>extrap</code> | a logical, if <code>TRUE</code> 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 kriging 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 Rmetrics user has to load this package from the CRAN server 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

```
## akimaInterp -
# Akima Interpolation:
if (require(akima)) {
  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)
}
```

as.matrix.ts

Convert Time Series to Matrix

Description

Creates a matrix from the given "ts" or "mts" time series objects, a method missing in R's base package.

Usage

```
## S3 method for class 'ts':
as.matrix(x, ...)
## S3 method for class 'mts':
as.matrix(x, ...)
```

Arguments

x 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".

... arguments to be passed.

Examples

```
## as.matrix -
z <- ts(matrix(rnorm(300), 100, 3), start=c(1961, 1), frequency=12)
class(z)
z
as.matrix(z)
```

 as.POSIXlt

Date-time Conversion Function

Description

Adds date-time conversion function "as.POSIXlt" which is missing in R's base package.

Usage

```
as.POSIXlt(x, tz = "")
```

Arguments

| | |
|----|-----------------------------|
| tz | time zone specification. |
| x | the object to be converted. |

Note

IMPORTANT NOTE: as.POSIXlt was introduced as generic function in R 2.7.0. We will keep here the Rmetrics implementation to be downward compatible with previous R/Rmetrics Versions. This has to be adapted.

Examples

```
## as.POSIXlt -
date = as.POSIXlt("2008-01-01")
print(date)
class(date)
```

 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:

| | |
|------------|---------------------------|
| align | adds align function, |
| atoms | adds atoms function, |
| attach | extends attach function, |
| colnames<- | adds colnames assignment, |
| cor | extends cor function, |
| cov | extends cov function, |
| log | extends log function, |

| | |
|------------|---------------------------|
| outlier | adds outlier function, |
| rank | extends rank function, |
| rownames<- | adds rownames assignment, |
| sample | extends sample function, |
| stdev | adds stdev function, |
| termPlot | adds term plot function, |
| var | extends var function, |
| volatility | adds volatility function. |

Usage

```
## Default S3 method:
align(x, y, xout, method = "linear", n = 50, rule = 1,
      f = 0, ties = mean, ...)

## Default S3 method:
atoms(x, ...)

## Default S3 method:
attach(what, pos = 2, name = deparse(substitute(what)),
      warn.conflicts = TRUE)

## Default S3 method:
cor(x, y = NULL, use = "everything",
    method = c("pearson", "kendall", "spearman"))

## Default S3 method:
cov(x, y = NULL, use = "everything",
    method = c("pearson", "kendall", "spearman"))

## Default S3 method:
log(x, base = exp(1))

## Default S3 method:
outlier(x, sd = 5, complement = TRUE, ...)

## Default S3 method:
rank(x, na.last = TRUE,
     ties.method = c("average", "first", "random", "max", "min"))

## Default S3 method:
sample(x, size, replace = FALSE, prob = NULL, ...)

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

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

```
## Default S3 method:
var(x, y = NULL, na.rm = FALSE, use)

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

rownames(x) <- value
colnames(x) <- value
```

Arguments

| | |
|---------|---|
| base | [log] - additional argument to the log function. See for details <code>help(log, package=base)</code> . |
| f | [align] - for method="constant" the value of f takes a number between 0 and 1 inclusive, indicating a compromise between left- and right-continuous step functions. If y0 and y1 are the values to the left and right of the point then the value is $y_0 * (1-f) + y_1 * f$ so that f=0 is right-continuous and f=1 is left-continuous. |
| method | [align] - a character string which specifies the alignment method to be used. Choices are "linear" or "constant". [cov] - a character string indicating which correlation coefficient (or covariance) is to be computed. One of "pearson" (default), "kendall", or "spearman", can be abbreviated. |
| n | [align] - if xout is not specified, alignment takes place at n equally spaced points spanning the interval <code>range(x)</code> . |
| na.last | [rank] - for controlling the treatment of NAs. If TRUE, missing values in the data are put last; if FALSE, they are put first; if NA, they are removed; if "keep" they are kept with rank NA. |
| name | [attach] - alternative way to specify the database to be attached. See for details <code>help(attach, package=base)</code> |
| model | [termPlot] - a fitted model object. |
| object | [volatility] - an object from which to extract the volatility. |
| pos | [attach] - an integer specifying position in <code>search()</code> where to attach the database. See for details <code>help(attach, package=base)</code> . |
| rule | [align] - an integer describing how alignment is to take place outside the interval <code>range(x)</code> . If rule=1 then NA's are returned for such points and if is rule=2, the value at the closest data extreme is used. |

sd, complement

[outlier] -
sd - a numeric value of standard deviations, e.g. 5 means that values larger or smaller than five times the standard deviation of the series will be detected. complement - a logical flag, should the outlier series or its complements be returned?

size, replace, prob

[sample] -
size - is a non-negative integer giving the number of items to choose,
replace - a logical flag. Should sampling be with replacement?
prob - a vector of probability weights for obtaining the elements of the vector being sampled.

ties

[align] -
handles tied x values. Either a function with a single vector argument returning a single number result or the string "ordered".

ties.method

[rank] -
a character string, one of "average", "first", "random", "max", "min" specifying how ties are treated, can be abbreviated, see rank.

value

[colnames][rownames] -
additional arguments to the colnames and rownames functions.

warn.conflicts

[attach] -
a logical value. If TRUE, warnings are printed about conflicts from attaching the database, unless that database contains an object .conflicts.OK. A conflict is a function masking a function, or a non-function masking a non-function. See for details help(attach, package=base).

what

[attach] -
database to be attached. This may currently be a timeSeries object, a data.frame or a list or a R data file created with save or NULL or an environment. See for details help(attach, package=base).

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.

xout

[align] -
a set of values specifying where interpolation is to take place.

y, na.rm, use

[align] -
y-coordinates of the points to be aligned.
[var] -

```

additional arguments to the var function.
[cov] -
additional arguments to the cov function.
...
arguments to be passed.

```

Details

For details we refer to the original help pages.

Examples

```

## Very Simple Outlier Detection:
# outlier -
set.seed(4711)
x = rnorm(1000)
outlier(x, sd = 3, complement = FALSE)

```

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)
colorMatrix(locator = TRUE)
```

Arguments

`locator` a logical flag, activates `locator` for interactive selection of color names, default is `FALSE`.

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=TRUE`, returns matrix with graph coordinates and names of colors selected. `colorsMatrix` quietly returns matrix of names.

See Also

```
link{colorPalette}, link{colorTable}.
```

Examples

```
## colorLocator -
colorLocator()
```

colorPalette *Color Palettes*

Description

Functions to create color palettes.

The functions are:

| | |
|----------------|-----------------------------------|
| 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. |

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 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`, `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 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 cotributed package `PerformanceAnalytics`. 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 plot colors with the associated color numbers.

See Also

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

Examples

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

colStats

*Column Statistics***Description**

A collection and description of functions to compute column statistical properties of financial and economic time series data.

The functions are:

| | |
|--------------|--|
| colStats | calculates column statistics, |
| colSums | calculates column sums, |
| colMeans | calculates column means, |
| colSds | calculates column standard deviations, |
| colVars | calculates column variances, |
| colSkewness | calculates column skewness, |
| colKurtosis | calculates column kurtosis, |
| colMaxs | calculates maximum values in each column, |
| colMins | calculates minimum values in each column, |
| colProds | computes product of all values in each column, |
| colQuantiles | computes quantiles of each column. |

Usage

```
colStats(x, FUN, ...)

colSums(x, ...)
colMeans(x, ...)

colSds(x, ...)
colVars(x, ...)
colSkewness(x, ...)
colKurtosis(x, ...)
colMaxs(x, ...)
colMins(x, ...)
colProds(x, ...)
colQuantiles(x, prob = 0.05, ...)

colStdevs(x, ...)
colAvg(x, ...)

## S3 method for class 'timeSeries':
mean(x, ...)
## S3 method for class 'timeSeries':
var(x, ...)
```

Arguments

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

Value

the functions return a numeric vector of the statistics.

See Also

`link{rowStats}`.

Examples

```
## Simulated Return Data in Matrix Form:
x = matrix(rnorm(252), ncol = 2)

## colStats -
colStats(x, FUN = mean)

## colQuantiles -
colQuantiles(x, prob = 0.10, type = 1)
```

`colVec`*Column and Row Vectors*

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)
```

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()
```

description

Date and User Information

Description

Returns data and user string.

Usage

```
description()
```

Examples

```
## description:
description()
```

distCheck

Distribution Check

Description

Tests properties of a distribution.

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 | a logical flag, should robust estimates be used? By default TRUE. |
| subdivisions | an integer specifying the numbers of subdivisions in integration. |
| ... | the distributional parameters. |

Examples

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

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

```
show.fHTEST(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. |
| 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:

| | |
|----------------|---|
| isS4 | Checks if an object is a S4 object, |
| getCall | Extracts the call slot from a S4 object, |
| 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. |

Usage

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

getSlot(object, slotName)
```

Arguments

| | |
|----------|--|
| object | an object of class S4. |
| slotName | a character string, the name of the slot to be extracted from the S4 object. |

Value

```
getCall
getModel
getTitle
getDescription
getSlot
return the content of the slot.
```

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)
```

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)
```

Heaviside

Heaviside and Related Functions

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
```

 Ids

Set and Retrieve Column/Row Names

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

| | |
|---------|--|
| choices | a vector of character strings for the choice menu. By Default "Plot 1" ... "Plot 9" allowing for 9 plots at maximum. |
| plotFUN | a vector of character strings naming the plot functions. By Default "plot.1" ... "plot.9" allowing for 9 plots at maximum. |
| which | 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. |
| x | an object to be plotted. |
| ... | 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 Kriging Interpolation

Description

Bivariate Kriging 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 kriging 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 loading of the R package `spatial`.

See Also

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

Examples

```
## 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 `%x%` or alternatively using the function `kron` for SPlus compatibility.

Note

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

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
```

kurtosis

Kurtosis

Description

Functions to compute kurtosis.

Usage

```
kurtosis(x, ...)
```

```
## Default S3 method:
kurtosis(x, na.rm = FALSE, method = c("excess", "moment", "fisher"), ...)
```

```
## S3 method for class 'data.frame':
kurtosis(x, ...)
```

```
## S3 method for class 'POSIXct':
kurtosis(x, ...)
```

```
## S3 method for class 'POSIXlt':
kurtosis(x, ...)
```

Arguments

| | |
|---------------------|--|
| <code>na.rm</code> | a logical. Should missing values be removed? |
| <code>method</code> | a character string which specifies the method of computation. These are either "moment", "fisher", or "excess". If "excess" is selected, then the value of the kurtosis is computed by the "moment" method and a value of 3 will be subtracted. The "moment" method is based on the definitions of kurtosis for distributions; these forms should be used when resampling (bootstrap or jackknife). The "fisher" method correspond to the usual "unbiased" definition of sample variance, although in the case of kurtosis exact unbiasedness is not possible. |
| <code>x</code> | a numeric vector or object. |
| <code>...</code> | arguments to be passed. |

Value

`kurtosis`
returns the value of the statistics, a numeric value. An attribute which reports the used method is added.

See Also

`link{skewness}`.

Examples

```
## mean -
## var -
# Mean, Variance:
r = rnorm(100)
mean(r)
var(r)

## kurtosis -
kurtosis(r)
```

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,

| | |
|--------------------------|---|
| <code>get.lcgseed</code> | Get the current value of the random seed, |
| <code>runif.lcg</code> | Uniform linear congruational generator, |
| <code>rnorm.lcg</code> | Normal linear congruational generator, |
| <code>rt.lcg</code> | 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

| | |
|-----------------------|--|
| <code>df</code> | number of degrees of freedom, a positive integer, maybe non-integer. |
| <code>mean, sd</code> | means and standard deviation of the normal distributed innovations. |
| <code>min, max</code> | lower and upper limits of the uniform distributed innovations. |
| <code>seed</code> | an integer value, the random number seed. |
| <code>n</code> | 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 congruational 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 *Bivariate Linear Interpolation*

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

| | |
|-------------------------|--|
| <code>x, y, z</code> | for <code>linearInterp</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 as <code>.matrix</code> into a matrix object. For <code>linearInterpp</code> we consider either three numeric vectors of equal length or if <code>y</code> and <code>z</code> are <code>NULL</code> , a list with entries <code>x, y, z</code> , or named data frame with <code>x</code> in the first, <code>y</code> in the second, and <code>z</code> in the third column. |
| <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> | 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 (require(akima)) {
  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 *Description File Listing*

Description

Lists the content of a description file.

Usage

```
listDescription(package = "Rmetrics", 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("fUtilities")
```

listFunctions *Functions Listing*

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("fUtilities")

## countFunctions -
countFunctions("fUtilities")
```

listIndex *Index File Listing*

Description

Lists the content of an index file.

Usage

```
listIndex(package = "Rmetrics", 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("fUtilities")
```

norm

Matrix Norm

Description

Returns the norm of a matrix.

Usage

```
norm(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 `norm` 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.

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:
norm(P)
```

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 λ 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 *Positive Definite Matrixes*

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 *Print Control*

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)
```

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 <code>method = "qr"</code> the rank is computed as <code>qr(x)\$rank</code> , or alternatively for <code>method="chol"</code> the rank is computed as <code>attr(chol(x, pivot=TRUE), "rank")</code> . |

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:

| | |
|--------------|---|
| rowStats | calculates row statistics, |
| rowSums | calculates row sums, |
| rowMeans | calculates row means, |
| rowSds | calculates row standard deviations, |
| rowVars | calculates row variances, |
| rowSkewness | calculates row skewness, |
| rowKurtosis | calculates row kurtosis, |
| rowMaxs | calculates maximum values in each row, |
| rowMins | calculates minimum values in each row, |
| rowProds | computes product of all values in each row, |
| rowQuantiles | computes quantiles of each row. |

Usage

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

rowSums(x, ...)
rowMeans(x, ...)

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

rowStdevs(x, ...)
```

```
rowAves(x, ...)
```

Arguments

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

skewness

Skewness

Description

Functions to compute skewness.

Usage

```
skewness(x, ...)

## Default S3 method:
skewness(x, na.rm = FALSE, method = c("moment", "fisher"), ...)
## S3 method for class 'data.frame':
skewness(x, ...)
## S3 method for class 'POSIXct':
skewness(x, ...)
## S3 method for class 'POSIXlt':
skewness(x, ...)
```

Arguments

| | |
|---------------------|--|
| <code>na.rm</code> | a logical. Should missing values be removed? |
| <code>method</code> | a character string which specifies the method of computation. These are either "moment" or "fisher" The "moment" method is based on the definitions of skewness for distributions; these forms should be used when resampling (bootstrap or jackknife). The "fisher" method correspond to the usual "unbiased" definition of sample variance, although in the case of skewness exact unbiasedness is not possible. |
| <code>x</code> | a numeric vector or object. |
| <code>...</code> | arguments to be passed. |

Value

skewness

returns the value of the statistics, a numeric value. An attribute which reports the used method is added.

See Also

`link{kurtosis}`.

Examples

```
## mean -
## var -
# Mean, Variance:
r = rnorm(100)
mean(r)
var(r)

## skewness -
skewness(r)
```

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()
```

`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
```

tslag

Lagged or Leading Vector/Matrix

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 -
#
```

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

- *Topic **hplot**
 - decor, 21
 - gridVector, 25
 - interactivePlot, 29
- *Topic **htest**
 - fHTEST, 23
- *Topic **math**
 - colVec, 20
 - hilbert, 28
 - Ids, 29
 - inv, 30
 - kron, 32
 - norm, 39
 - pascal, 40
 - pdl, 41
 - positiveDefinite, 42
 - rk, 43
 - tr, 47
 - triang, 48
 - tslag, 49
 - vec, 50
- *Topic **package**
 - fUtilities-package, 2
- *Topic **programming**
 - akimaInterp, 6
 - as.matrix.ts, 8
 - as.POSIXlt, 9
 - baseMethods, 9
 - characterTable, 13
 - colorLocator, 14
 - colorPalette, 15
 - colorTable, 18
 - description, 22
 - distCheck, 22
 - getS4, 24
 - Heaviside, 26
 - krigeInterp, 31
 - lcg, 34
 - linearInterp, 36
 - listDescription, 37
 - listFunctions, 38
 - listIndex, 38
 - print, 42
 - symbolTable, 46
- *Topic **univar**
 - colStats, 19
 - kurtosis, 33
 - rowStats, 44
 - skewness, 45
- akimaInterp, 6, 32, 36
- akimaInterpp (*akimaInterp*), 6
- align (*baseMethods*), 9
- as.matrix.mts (*as.matrix.ts*), 8
- as.matrix.ts, 8
- as.POSIXlt, 9
- atoms (*baseMethods*), 9
- attach (*baseMethods*), 9
- baseMethods, 9
- box_ (*decor*), 21
- Boxcar (*Heaviside*), 26
- boxL (*decor*), 21
- characterTable, 13
- cmPalette (*colorPalette*), 15
- colAvgs (*colStats*), 19
- colIds (*Ids*), 29
- colIds<- (*Ids*), 29
- colKurtosis (*colStats*), 19
- colMaxs (*colStats*), 19
- colMeans (*colStats*), 19
- colMins (*colStats*), 19
- colnames<- (*baseMethods*), 9
- colorLocator, 14
- colorMatrix (*colorLocator*), 14
- colorPalette, 15
- colorTable, 18
- colProds (*colStats*), 19

- colQuantiles (*colStats*), 19
- colSds (*colStats*), 19
- colSkewness (*colStats*), 19
- colStats, 19
- colStdevs (*colStats*), 19
- colSums (*colStats*), 19
- colVars (*colStats*), 19
- colVec, 20
- copyright (*decor*), 21
- cor (*baseMethods*), 9
- countFunctions (*listFunctions*), 38
- cov (*baseMethods*), 9

- decor, 21
- Delta (*Heaviside*), 26
- description, 22
- distCheck, 22
- divPalette (*colorPalette*), 15

- expand.grid, 26

- fHTEST, 23
- fHTEST-class (*fHTEST*), 23
- focusPalette (*colorPalette*), 15
- fUtilities (*fUtilities-package*), 2
- fUtilities-package, 2

- get.lcgseed (*lcg*), 34
- getCall (*getS4*), 24
- getDescription (*getS4*), 24
- getModel (*getS4*), 24
- getS4, 24
- getSlot (*getS4*), 24
- getTitle (*getS4*), 24
- greyPalette (*colorPalette*), 15
- gridVector, 25

- heatPalette (*colorPalette*), 15
- Heaviside, 26
- hgrid (*decor*), 21
- hilbert, 28

- Ids, 29
- interactivePlot, 29
- inv, 30
- isPositiveDefinite
 (*positiveDefinite*), 42

- krigeInterp, 7, 31, 36
- kron, 32

- kurtosis, 33

- lcg, 34
- linearInterp, 7, 32, 36
- linearInterpp (*linearInterp*), 36
- listDescription, 37, 39
- listFunctions, 37, 38, 38
- listIndex, 37, 38, 38, 39
- log (*baseMethods*), 9

- makePositiveDefinite
 (*positiveDefinite*), 42
- mean.timeSeries (*colStats*), 19
- monoPalette (*colorPalette*), 15

- norm, 39

- outlier (*baseMethods*), 9

- pascal, 40
- pdl, 41, 49
- positiveDefinite, 42
- print, 42

- qualiPalette (*colorPalette*), 15

- rainbowPalette (*colorPalette*), 15
- Ramp (*Heaviside*), 26
- rampPalette (*colorPalette*), 15
- rank (*baseMethods*), 9
- rk, 43
- rnorm.lcg (*lcg*), 34
- rowAvg (rowStats), 44
- rowIds (*Ids*), 29
- rowIds<- (*Ids*), 29
- rowKurtosis (rowStats), 44
- rowMaxs (rowStats), 44
- rowMeans (rowStats), 44
- rowMins (rowStats), 44
- rownames<- (*baseMethods*), 9
- rowProds (rowStats), 44
- rowQuantiles (rowStats), 44
- rowSds (rowStats), 44
- rowSkewness (rowStats), 44
- rowStats, 44
- rowStdevs (rowStats), 44
- rowSums (rowStats), 44
- rowVars (rowStats), 44
- rowVec (*colVec*), 20
- rt.lcg (*lcg*), 34

`runif.lcg(lcg)`, 34

`sample(baseMethods)`, 9

`seqPalette(colorPalette)`, 15

`set.lcgseed(lcg)`, 34

`show, fhTEST-method(fhTEST)`, 23

`show.fhTEST(fhTEST)`, 23

`Sign(Heaviside)`, 26

`skewness`, 45

`stdev(baseMethods)`, 9

`symbolTable`, 46

`termPlot(baseMethods)`, 9

`terrainPalette(colorPalette)`, 15

`timPalette(colorPalette)`, 15

`topoPalette(colorPalette)`, 15

`tr`, 47

`Triang(triang)`, 48

`triang`, 48

`tslag`, 41, 49

`var(baseMethods)`, 9

`var.timeSeries(colStats)`, 19

`vec`, 50

`vech(vec)`, 50

`vgrid(decor)`, 21

`volatility(baseMethods)`, 9