# Package 'FatTailsR'

July 14, 2014

**Title** Power Hyperbolic Functions and Kiener Distributions

**Description** A set of functions to manipulate power hyperbolas, power hyperbolic functions and Kiener distributions of type I, II, III and IV which exhibit left and right fat tails like those that occur in financial markets. These distributions can be used to estimate with a high accuracy market risks and value-at-risk.

**URL** http://www.inmodelia.com/fattailsr-en.html

**Version** 1.0-3

**Date** 2014-07-14

**Author** Patrice Kiener

**Maintainer** Patrice Kiener <fattailsr@inmodelia.com>

**Depends** R (>= 3.1.0)

**Imports** minpack.lm

**Suggests** timeSeries, timeDate

**License** GPL-2

**LazyData** true

**NeedsCompilation** no

**Collate** 'FatTailsR-package.r' 'trigohp.R' 'logishp.R' 'conversion.R' 'kiener1.R' 'kiener2.R' 'kiener3.R' 'kiener4.R' 'regression.R'

**Repository** CRAN

**Date/Publication** 2014-07-14 23:24:37

# R topics documented:



---

`FatTailsR-package` *Package FatTailsR*

---

## Description

A set of functions to manipulate power hyperbolas, power hyperbolic functions and Kiener distributions of type I, II, III and IV which exhibit left and right fat tails like those that occur in financial markets. These distributions can be used to estimate with a high accuracy market risks, value-at-risk and expected shortfalls. Download the pdf cited in the references to get an overview of the theoretical part and several examples on stocks and indices.

## Details

With so many functions, this package could look fat. But it's not! It's rather agile and easy to use! It addresses two topics: power hyperbolic functions and Kiener distributions. The various functions can be assigned to the following groups:

1. Miscellaneous functions for general purpose:
   - `logit`, invlogit.
2. Miscellaneous functions for power hyperbolas:
   - `kashp`, dkashp_dx, ashp.
3. Power hyperbolas and power hyperbolic functions:
   - `exphp`, coshp, sinhp, tanhp, sechp, cosechp, cotanhp.
4. Inverse power hyperbolas and inverse power hyperbolic functions:
   - `loghp`, acoshp, asinhp, atanhp, asechp, acosechp, acotanhp.
5. The logishp function and its variants which combines the logistic function and the power hyperbolas:
   - d, p, q, r, dp, dq, l, dl, ql `logishp`.

6. The Kiener distributions of type I, II, III and IV:
   - d, p, q, r, dp, dq, l, dl, ql `kiener1`,
   - d, p, q, r, dp, dq, l, dl, ql `kiener2`,
   - d, p, q, r, dp, dq, l, dl, ql `kiener3`,
   - d, p, q, r, dp, dq, l, dl, ql `kiener4`.
7. Conversion functions between parameters pertaining to Kiener distributions of type II, III and IV:
   - `aw2k`, aw2d, aw2e, kd2a, kd2w, ke2a, ke2w, ke2d, kd2e, de2k.
8. Regression functions to estimate the distribution parameters of a given dataset with Kiener distributions and Laplace-Gauss normal distribution:
   - `regkienerLX`, `laplacegaussnorm`.

For a quick start on Kiener distributions, jump to the function `regkienerLX` and run the examples. Then, download and read the document in pdf format cited in the references which gives a complete overview on all functions. Finally, explore the other examples.

## References

P. Kiener, Explicit models for bilateral fat-tailed distributions and applications in finance with the package FatTailsR, 8th R/Rmetrics Workshop and Summer School, Paris, 27 June 2014. Download it from: http://www.inmodelia.com/exemples/2014-0627-Rmetrics-Kiener-en.pdf

## Examples

```
require(graphics)
require(minpack.lm)
require(timeSeries)

prices2returns <- function(x) { 100*diff(log(x)) }

### Load the various datasets (1-16)
DS <- list(
"USDCHF"       = prices2returns(as.numeric(USDCHF)) ,
"Microsoft"    = prices2returns(as.numeric(MSFT[,4])) ,
"DAX"          = prices2returns(as.numeric(EuStockMarkets[,1])) ,
"SMI"          = prices2returns(as.numeric(EuStockMarkets[,2])) ,
"CAC"          = prices2returns(as.numeric(EuStockMarkets[,3])) ,
"FTSE"         = prices2returns(as.numeric(EuStockMarkets[,4])) ,
"SBI"          = as.numeric(LPP2005REC[,1]) ,
"SPI"          = as.numeric(LPP2005REC[,2]) ,
"SII"          = as.numeric(LPP2005REC[,3]) ,
"LMI"          = as.numeric(LPP2005REC[,4]) ,
"MPI"          = as.numeric(LPP2005REC[,5]) ,
"ALT"          = as.numeric(LPP2005REC[,6]) ,
"LPP25"        = as.numeric(LPP2005REC[,7]) ,
"LPP40"        = as.numeric(LPP2005REC[,8]) ,
"LPP60"        = as.numeric(LPP2005REC[,9]) ,
"sunspot.year" = prices2returns(as.numeric(sunspot.year)+1000) )
```

```
### Select one dataset number (1-16)
j      <- 5

### and run this block
X      <- DS[[j]]
nameX  <- names(DS)[j]
reg    <- regkienerLX(X)
lgn    <- laplacegaussnorm(X)
lleg   <- c("logit(0.999) = 6.9", "logit(0.99)   = 4.6",
            "logit(0.95)   = 2.9", "logit(0.50)   = 0",
            "logit(0.05)   = -2.9", "logit(0.01)   = -4.6",
            "logit(0.001) = -6.9  ")
pleg   <- c( paste("m =",  reg$coefr4[1]), paste("g  =", reg$coefr4[2]),
               paste("k  =", reg$coefr4[3]), paste("e  =", reg$coefr4[4]) )

## Main plot
op     <- par(mfrow = c(1,1), mgp = c(1.5,0.8,0), mar = c(3,3,2,1))
plot(reg$dfrXP, main = nameX)
legend("top", legend = pleg, cex = 0.9, inset = 0.02 )
lines(reg$dfrEP, col = 2, lwd = 2)
points(reg$dfrQkPk, pch = 3, col = 2, lwd = 2, cex = 1.5)
lines(lgn$dfrXPn, col = 2, lwd = 2)

## Plot F(X) > 0,97
front = c(0.06, 0.39, 0.50, 0.95)
par(fig = front, new = TRUE, mgp = c(1.5, 0.6, 0), las = 0)
plot( reg$dfrXP[which(reg$dfrXP$P > 0.97),] , pch = 1, xlab = "", ylab = "", main = "F(X) > 0,97" )
lines(reg$dfrEP[which(reg$dfrEP$P > 0.97),], type="l", col = 2, lwd = 3 )
lines(lgn$dfrXPn[which(lgn$dfrXPn$Pn > 0.97),], type = "l", col = 7, lwd= 2 )
points(reg$dfrQkPk, pch = 3, col = 2, lwd = 2, cex = 1.5)
points(lgn$dfrQnPn, pch = 3, col = 7, lwd = 2, cex = 1)

## Plot F(X) < 0,03
front = c(0.58, 0.98, 0.06, 0.61)
par(fig = front, new = TRUE, mgp = c(0.5, 0.6, 0), las = 0 )
plot( reg$dfrXP[which(reg$dfrXP$P < 0.03),] , pch = 1, xlab = "", ylab = "", main = "F(X) < 0,03")
lines(reg$dfrEP[which(reg$dfrEP$P < 0.03),], type = "l", col = 2, lwd = 3 )
lines(lgn$dfrXPn[which(lgn$dfrXPn$Pn < 0.03),], type = "l", col= 7, lwd= 2 )
points(reg$dfrQkPk, pch = 3, col = 2, lwd = 2, cex = 1.5)
points(lgn$dfrQnPn, pch = 3, col = 7, lwd = 2, cex = 1)
### End block
```

---

aw2k *Conversion Functions*

---

## Description

Conversion functions between parameters a, k, w, d, e used in Kiener distributions of type II, III and IV.

## Usage

```
aw2k(a, w)

aw2d(a, w)

aw2e(a, w)

kd2a(k, d)

kd2w(k, d)

ke2a(k, e)

ke2w(k, e)

ke2d(k, e)

kd2e(k, d)

de2k(d, e)
```

## Arguments

| | |
|---|---|
| a | a numeric value. |
| k | a numeric value. |
| w | a numeric value. |
| d | a numeric value. |
| e | a numeric value. |

## Details

a (alpha) is the left tail parameter, w (omega) is the right tail parameter, d (delta) is the distorsion parameter, e (epsilon) is the eccentricity parameter.

k (kappa) is the harmonic mean of a and w and describes a global tail parameter.

They are defined by:

$$aw2k(a, w) = 2/(1/a + 1/w) = k$$

$$aw2d(a, w) = (-1/a + 1/w)/2 = d$$

$$aw2e(a, w) = (a - w)/(a + w) = e$$

$$kd2a(k, d) = 1/(1/k - d) = a$$

$$kd2w(k, d) = 1/(1/k + d) = w$$

$$ke2a(k, e) = k/(1 - e) = a$$

$$ke2w(k, e) = k/(1 + e) = w$$

$$ke2d(k, e) = e/k = d$$

$$kd2e(k, d) = d * k = e$$

$$de2k(k, e) = e/d = k$$

aw2k is equivalent to the harmonic mean.

## See Also

The asymmetric Kiener distributions of type II, III and IV: kiener2, kiener3, kiener4

## Examples

```
aw2k(4, 6); aw2d(4, 6); aw2e(4, 6)
outer(1:6, 1:6, aw2k)
```

---

exphp *Power Hyperbolas and Power Hyperbolic Functions*

---

## Description

These functions define the power hyperbola exphp and the associated power hyperbolic cosine, sine, tangent, secant, cosecant, cotangent. They are similar to the traditional hyperbolic functions with term x receiving a nonlinear transformation via the function kashp.

## Usage

```
exphp(x, k = 1)

coshp(x, k = 1)

sinhp(x, k = 1)

tanhp(x, k = 1)

sechp(x, k = 1)

cosechp(x, k = 1)

cotanhp(x, k = 1)
```

## Arguments

| | |
|---|---|
| x | a numeric value, vector or matrix. |
| k | a numeric value, preferably strictly positive. |

## Details

exphp function is defined for x in (-Inf, +Inf) by:

$$exphp(x, k) = exp(kashp(x, k)) = exp(k * asinh(x/2/k))$$

coshp function is defined for x in (-Inf, +Inf) by:

$$coshp(x, k) = cosh(kashp(x, k))$$

sinhp function is defined for x in (-Inf, +Inf) by:

$$sinhp(x, k) = sinh(kashp(x, k))$$

tanhp function is defined for x in (-Inf, +Inf) by:

$$tanhp(x, k) = tanh(kashp(x, k))$$

sechp function is defined for x in (-Inf, +Inf) by:

$$sechp(x, k) = 1/coshp(x, k)$$

cosechp function is defined for x in (-Inf, 0) U (0, +Inf) by:

$$cosechp(x, k) = 1/sinhp(x, k)$$

cotanhp function is defined for x in (-Inf, 0) U (0, +Inf) by:

$$cotanhp(x, k) = 1/tanhp(x, k)$$

The undesired case k = 0 returns 0 for sinhp and tanhp, 1 for exphp, coshp and sechp, Inf for cosechp and cotanhp.

If k is a vector of length > 1, then the use of the function `outer` is recommanded.

## See Also

The nonlinear transformation `kashp`, the inverse power hyperbolas and the inverse power hyperbolic functions `loghp`.

## Examples

```
### Example 1
x  <- (-2:3)*3
exphp(x, k = 4)
coshp(x, k = 4)
sinhp(x, k = 4)
tanhp(x, k = 4)

### Example 2
mat  <- matrix(x , ncol = 2)
exphp(mat, k = 4)
coshp(mat, k = 4)
sinhp(mat, k = 4)
```

```
tanhp(mat, k = 4)

### Example 3 outer + plot(exphp, coshp, sinhp, tanhp)
xmin  <- -10
xd    <- 0.5
x     <- seq(xmin, -xmin, xd) ; names(x) <- x
k     <- c(0.6, 1, 1.5, 2, 3.2, 10) ; names(k) <- k
olty  <- c(2, 1, 2, 1, 2, 1, 1)
olwd  <- c(1, 1, 2, 2, 3, 4, 2)
ocol  <- c(2, 2, 4, 4, 3, 3, 1)
op    <- par(mfrow = c(2,2), mgp = c(1.5,0.8,0), mar = c(3,3,2,1))

## exphp(x, k)
Texphp <- ts(cbind(outer(-x, k, exphp), "exp(-x/2)" = exp(-x/2)),
             start = xmin, deltat = xd)
plot(Texphp, plot.type = "single", ylim = c(0,20),
       lty = olty, lwd = olwd, col = ocol, xaxs = "i", yaxs = "i", xlab = "",
       ylab = "", main = "exphp(-x, k)" )
legend("topright", title = expression(kappa), legend = colnames(Texphp),
       inset = 0.02, lty = olty, lwd = olwd, col = ocol, cex = 0.7 )

## coshp(x, k)
Tcoshp <- ts(cbind(outer(x, k, coshp), "cosh(x/2)" = cosh(x/2)),
             start = xmin, deltat = xd)
plot(Tcoshp, plot.type = "single", ylim = c(0,20),
       lty = olty, lwd = olwd, col = ocol, xaxs = "i", yaxs = "i",
       xlab = "", ylab = "", main = "coshp(x, k)" )
legend("top", title = expression(kappa), legend = colnames(Tcoshp),
       inset = 0.02, lty = olty, lwd = olwd, col = ocol, cex = 0.7 )

## sinhp(x, k)
Tsinhp <- ts(cbind(outer(x, k, sinhp), "sinh(x/2)" = sinh(x/2)),
             start = xmin, deltat=xd)
plot(Tsinhp, plot.type = "single", ylim = c(-10,10),
       lty = olty, lwd = olwd, col = ocol, xaxs = "i", yaxs = "i",
       xlab = "", ylab = "", main = "sinhp(x, k)" )
legend("topleft", title = expression(kappa), legend = colnames(Tsinhp),
       inset = 0.02, lty = olty, lwd=  olwd, col = ocol, cex = 0.7 )

## tanhp(x, k)
Ttanhp <- ts(cbind(outer(x, k, tanhp), "tanh(x/2)" = tanh(x/2)),
             start = xmin, deltat = xd)
plot(Ttanhp, plot.type = "single", ylim = c(-1,1),
       lty = olty, lwd = olwd, col = ocol, xaxs = "i", yaxs = "i", xlab = "",
       ylab = "", main = "tanhp(x, k)" )
legend("topleft", title = expression(kappa), legend = colnames(Ttanhp),
       inset = 0.02, lty = olty, lwd = olwd, col = ocol, cex = 0.7 )
### End Example 3
```

---

kashp *Kashp Function*

---

## Description

kashp, which stands for kappa times arc-sine-hyperbola-power is the nonlinear transformation of x at the heart of power hyperbolas, power hyperbolic functions and symmetric Kiener distributions. dkashp_dx is its derivative with respect to x. ashp is provided for convenience.

## Usage

```
kashp(x, k = 1)

dkashp_dx(x, k = 1)

ashp(x, k = 1)
```

## Arguments

| | |
|---|---|
| x | a numeric value, vector or matrix. |
| k | a numeric value or vector, preferably strictly positive. |

## Details

ashp function is defined for x in (-Inf, +Inf) by:

$$ashp(x, k) = asinh(x/2/k)$$

kashp function is defined for x in (-Inf, +Inf) by:

$$kashp(x, k) = k * asinh(x/2/k)$$

dkashp_dx function is defined for x in (-Inf, +Inf) by:

$$dkashp_dx(x, k) = 1/sqrt(x * x/k/k + 4) = 1/2/cosh(ashp(x, k))$$

If k is a vector, then the use of the function outer is recommanded.

The undesired case k=0 returns 0 for kashp and dkashp_dx, 1 for exphp, -Inf, NaN, +Inf for ashp.

## See Also

The power hyperbolas and the power hyperbolic functions exphp.

## Examples

```
require(graphics)

### Example 1
x    <- (-2:3)*3 ; names(x) <- x
mat  <- matrix(x, ncol=2)
kashp(x, k=2)
kashp(mat, k=2)
k    <- c(-2, 0, 1, 2, 4, 10) ; names(k) <- k
outer(x, k, kashp)
```

```
outer(x, k, exphp)

### Example 2
xmin   <- -10
xd     <- 0.5
x      <- seq(xmin, -xmin, xd) ; names(x) <- x
k      <- c(0.6, 1, 1.5, 2, 3.2, 10) ; names(k) <- k
olty   <- c(2, 1, 2, 1, 2, 1, 1)
olwd   <- c(1, 1, 2, 2, 3, 4, 2)
ocol   <- c(2, 2, 4, 4, 3, 3, 1)
op     <- par(mfrow = c(2,2), mgp = c(1.5,0.8,0), mar = c(3,3,2,1))

Tkashp <- ts(cbind(outer(x, k, kashp), "x/2" = x/2), start = xmin, deltat = xd)
plot(Tkashp, plot.type = "single", ylim = c(-5, +5),
       lty = olty, lwd = olwd, col = ocol, xaxs = "i", yaxs = "i", xlab = "",
       ylab = "", main = "kashp(x, k)" )
legend("topleft", title = expression(kappa), legend = colnames(Tkashp),
       inset = 0.02, lty = olty, lwd = olwd, col = ocol, cex = 0.7 )

Tdkashp <- ts(cbind(outer(x, k, dkashp_dx)), start = xmin, deltat = xd)
plot(Tdkashp, plot.type = "single", ylim = c(0, 0.8),
       lty = olty, lwd = olwd, col = ocol, xaxs = "i", yaxs = "i", xlab = "",
       ylab = "", main="dkashp_dx(x, k)" )
legend("topleft", title = expression(kappa), legend = colnames(Tdkashp),
       inset = 0.02, lty = olty, lwd = olwd, col = ocol, cex = 0.7 )

Tashp <- ts(cbind(outer(x, k, ashp), "x/2" = x/2), start = xmin, deltat = xd)
plot(Tashp, plot.type = "single", ylim = c(-5, +5),
       lty = olty, lwd = olwd, col = ocol, xaxs = "i", yaxs = "i", xlab = "",
       ylab = "", main = "ashp(x, k)" )
legend("topleft", title = expression(kappa), legend = colnames(Tashp),
       inset = 0.02, lty = olty, lwd = olwd, col = ocol, cex = 0.7 )
### End example 2
```

---

kiener1 *Symmetric Kiener Distribution (Type I)*

---

## Description

Density, distribution function, quantile function, random generation and additional formulae for the Kiener distribution of type I. This distribution is similar to the power hyperbola logistic distribution but with additional parameters for location (m) and scale (g).

## Usage

```
dkiener1(x, m = 0, g = 1, k = 3.2, log = FALSE)

pkiener1(q, m = 0, g = 1, k = 3.2, lower.tail = TRUE, log.p = FALSE)

qkiener1(p, m = 0, g = 1, k = 3.2, lower.tail = TRUE, log.p = FALSE)
```

```
rkiener1(n, m = 0, g = 1, k = 3.2)

dpkiener1(p, m = 0, g = 1, k = 3.2, log = FALSE)

dqkiener1(p, m = 0, g = 1, k = 3.2, log = FALSE)

lkiener1(x, m = 0, g = 1, k = 3.2)

dlkiener1(lp, m = 0, g = 1, k = 3.2, log = FALSE)

qlkiener1(lp, m = 0, g = 1, k = 3.2, lower.tail = TRUE)
```

## Arguments

| | |
|---|---|
| x | vector of quantiles. |
| q | vector of quantiles. |
| m | numeric. The median. |
| g | numeric. The scale parameter, preferably strictly positive. |
| k | numeric. The tail parameter, preferably strictly positive. |
| p | vector of probabilities. |
| lp | vector of logit of probabilities. |
| n | number of observations. If length(n) > 1, the length is taken to be the number required. |
| log | logical; if TRUE, densities are given in log scale. |
| log.p | logical; if TRUE, probabilities p are given as log(p). |
| lower.tail | logical; if TRUE (default), probabilities are P[X = x] otherwise, P[X > x]. |

## Details

Kiener distributions of type I describe distributions with symmetric left and right fat tails with tail parameter k. This parameter is the power exponent mentionned in Pareto formula and Karamata theorems.

m is the median of the distribution. g is the scale parameter and the inverse of the density at the median: $g = 1/8/f(m)$. As a first estimate, it is approximatively one fourth of the standard deviation $g \approx \sigma/4$ but is independant from it.

dkiener1 function is defined for x in (-Inf, +Inf) by:

$$dkiener1(x, m, g, k) = 1/4/g/cosh(ashp((x-m)/g, k))/(1 + cosh(kashp((x-m)/g, k)))$$

pkiener1 function is defined for q in (-Inf, +Inf) by:

$$pkiener1(q, m, g, k) = 1/(1 + exp(-kashp((q-m)/g, k)))$$

qkiener1 function is defined for p in (0, 1) by:

$$qkiener1(p, m, g, k) = m + 2 * g * k * sinh(logit(p)/k)$$

`rkiener1` generates n random quantiles.

In addition to the classical d, p, q, r functions, the prefixes dp, dq, l, dl, ql are also provided.

`dpkiener1` is the density function calculated from the probability p. It is defined for p in (0, 1) by:

$$dpkiener1(p, m, g, k) = p * (1 - p)/2/g/cosh(logit(p)/k)$$

`dqkiener1` is the derivate of the quantile function calculated from the probability p. It is defined for p in (0, 1) by:

$$dqkiener1(p, m, g, k) = 2 * g/p/(1 - p) * cosh(logit(p)/k)$$

`lkiener1` function is equivalent to kashp function but with additional parameters `m` and `g`. Being computed from the x (or q) vector, it can be compared to the logit of the empirical probability logit(p) through a nonlinear regression with ordinary or weighted least squares to estimate the distribution parameters. It is defined for x in (-Inf, +Inf) by:

$$lkiener1(x, m, g, k) = kashp((x - m)/g, k)$$

`dlkiener1` is the density function calculated from the logit of the probability lp = logit(p). It is defined for lp in (-Inf, +Inf) by:

$$dlkiener1(lp, m, g, k) = p * (1 - p)/2/g/cosh(lp/k)$$

`qlkiener1` is the quantile function calculated from the logit of the probability lp = logit(p). It is defined for lp in (-Inf, +Inf) by:

$$qlkiener1(lp, m, g, k) = m + g * k * 2 * sinh(lp/k)$$

## References

P. Kiener, Explicit models for bilateral fat-tailed distributions and applications in finance with the package FatTailsR, 8th R/Rmetrics Workshop and Summer School, Paris, 27 June 2014. Download it from: http://www.inmodelia.com/exemples/2014-0627-Rmetrics-Kiener-en.pdf

## See Also

The power hyperbola logistic distribution `logishp`, the asymmetric Kiener distributions of type II, III and IV `kiener2`, `kiener3`, `kiener4`, the regression function `regkienerLX`.

## Examples

```
require(graphics)

### Example 1
pp <- c(ppoints(11, a = 1), NA, NaN) ; pp
qkiener1(p = pp, k = 4)


### Example 2
k       <- 4.8
```

```
set.seed(2014)
mainTC  <- paste("qkiener1(p, m = 0, g = 1, k = ", k, ")")
mainsum <- paste("cumulated qkiener1(p, m = 0, g = 1, k = ", k, ")")
T       <- 500
C       <- 4
TC      <- qkiener1(p = runif(T*C), m = 0, g = 1, k = k)
matTC   <- matrix(TC, nrow = T, ncol = C, dimnames = list(1:T, letters[1:C]))
head(matTC)
plot.ts(matTC, main = mainTC)
#
matsum  <- apply(matTC, MARGIN=2, cumsum)
head(matsum)
plot.ts(matsum, plot.type = "single", main = mainsum)
### End example 2


### Example 3 (four plots: probability, density, logit, logdensity)
x  <- q  <- seq(-15, 15, length.out=101)
k     <- c(0.6, 1, 1.5, 2, 3.2, 10) ; names(k) <- k ; k
olty  <- c(2, 1, 2, 1, 2, 1, 1)
olwd  <- c(1, 1, 2, 2, 3, 3, 2)
ocol  <- c(2, 2, 4, 4, 3, 3, 1)
lleg  <- c("logit(0.999) = 6.9", "logit(0.99)   = 4.6", "logit(0.95)   = 2.9",
           "logit(0.50)   = 0", "logit(0.05)   = -2.9", "logit(0.01)   = -4.6",
           "logit(0.001) = -6.9  ")
op    <- par(mfrow=c(2,2), mgp=c(1.5,0.8,0), mar=c(3,3,2,1))

plot(x, plogis(x, scale = 2), type = "b", lwd = 2, ylim = c(0, 1),
     xaxs = "i", yaxs = "i", xlab = "", ylab = "",
     main = "pkiener1(q, m, g, k)")
for (i in 1:length(k)) lines(x, pkiener1(x, k = k[i]),
       lty = olty[i], lwd = olwd[i], col = ocol[i] )
legend("topleft", title = expression(kappa), legend = c(k, "logistic"),
       cex = 0.7, inset = 0.02, lty = olty, lwd = olwd, col = ocol )

plot(x, dlogis(x, scale = 2), type = "b", lwd = 2, ylim = c(0, 0.14),
     xaxs = "i", yaxs = "i", xlab = "", ylab = "", main = "dkiener1(x, m, g, k)" )
for (i in 1:length(k)) lines(x, dkiener1(x, k = k[i]),
       lty = olty[i], lwd = olwd[i], col = ocol[i] )
legend("topright", title = expression(kappa), legend = c(k, "logistic"),
       cex = 0.7, inset = 0.02, lty = olty, lwd = olwd, col = ocol )

plot(x, x/2, type = "b", lwd = 2, ylim = c(-7.5, 7.5), yaxt="n", xaxs = "i",
     yaxs = "i", xlab = "", ylab = "", main = "logit(pkiener1(q, m, g, k))")
axis(2, las=1, at=c(-6.9, -4.6, -2.9, 0, 2.9, 4.6, 6.9) )
for (i in 1:length(k)) lines(x, lkiener1(x, k = k[i]),
       lty = olty[i], lwd = olwd[i], col = ocol[i] )
lines(x, logit(pnorm(x, 0, 3.192)), type="l", lty=1, lwd=3, col=7) # erfx
legend("topleft", legend = lleg, cex = 0.7, inset = 0.02 )
legend("bottomright", title = expression(kappa),
       legend = c(k, "logistic", "Gauss"), cex = 0.7, inset = 0.02,
       lty = c(olty, 1), lwd = c(olwd, 3), col = c(ocol , 7) )
```

```
plot(x, log(dlogis(x, scale = 2)), lwd = 2, type = "b", ylim = c(-8, -1.5),
     xaxs = "i", yaxs = "i", xlab = "", ylab = "", main = "log(dkiener1(x, m, g, k))")
for (i in 1:length(k)) lines(x, log(dkiener1(x, k = k[i])),
       lty = olty[i], lwd = olwd[i], col = ocol[i] )
lines(x, dnorm(x, 0, 3.192, log = TRUE), type = "l", lty = 1, lwd = 3, col = 7)
legend("bottom", title = expression(kappa), legend = c(k, "logistic", "Gauss"),
       cex = 0.7, inset = 0.02, lty = c(olty, 1), lwd = c(olwd, 3), col = c(ocol , 7) )
### End example 3


### Example 4 (four plots: quantile, derivate, density and quantiles from p)
p   <- ppoints(199, a=0)
k   <- c(0.6, 1, 1.5, 2, 3.2, 10) ; names(k) <- k ; k
op  <- par(mfrow=c(2,2), mgp=c(1.5,0.8,0), mar=c(3,3,2,1))
plot(p, qlogis(p, scale = 2), type = "o", lwd = 2, ylim = c(-15, 15),
     xaxs = "i", yaxs = "i", xlab = "", ylab = "",
     main = "qkiener1(p, m, g, k)")
for (i in 1:length(k)) lines(p, qkiener1(p, k = k[i]),
          lty = olty[i], lwd = olwd[i], col = ocol[i] )
legend("topleft", title = expression(kappa), legend = c(k, "qlogis(x/2)"),
          inset = 0.02, lty = olty, lwd = olwd, col = ocol, cex = 0.7 )

plot(p, 2/p/(1-p), type = "o", lwd = 2, xlim = c(0, 1), ylim = c(0, 100),
     xaxs = "i", yaxs = "i", xlab = "", ylab = "",
     main = "dqkiener1(p, m, g, k)")
for (i in 1:length(k)) lines(p, dqkiener1(p, k = k[i]),
          lty = olty[i], lwd = olwd[i], col = ocol[i] )
legend("top", title = expression(kappa), legend = c(k, "p*(1-p)/2"),
          inset = 0.02, lty = olty, lwd = olwd, col = ocol, cex = 0.7 )

plot(qlogis(p, scale = 2), p*(1-p)/2, type = "o", lwd = 2, xlim = c(-15, 15),
     ylim = c(0, 0.14), xaxs = "i", yaxs = "i", xlab = "", ylab = "",
     main = "qkiener1, dpkiener1(p, m, g, k)")
for (i in 1:length(k)) lines(qkiener1(p, k = k[i]), dpkiener1(p, k = k[i]),
          lty = olty[i], lwd = olwd[i], col = ocol[i] )
legend("topleft", title = expression(kappa), legend = c(k, "p*(1-p)/2"),
          inset = 0.02, lty = olty, lwd = olwd, col = ocol, cex = 0.7 )
### End example 4
```

---

`kiener2` *Asymmetric Kiener Distribution (Type II)*

---

## Description

Density, distribution function, quantile function, random generation and additional formulae for the Kiener distribution of type II.

## Usage

```
dkiener2(x, m = 0, g = 1, a = 3.2, w = 3.2, log = FALSE)
```

```
pkiener2(q, m = 0, g = 1, a = 3.2, w = 3.2, lower.tail = TRUE,
  log.p = FALSE)

qkiener2(p, m = 0, g = 1, a = 3.2, w = 3.2, lower.tail = TRUE,
  log.p = FALSE)

rkiener2(n, m = 0, g = 1, a = 3.2, w = 3.2)

dpkiener2(p, m = 0, g = 1, a = 3.2, w = 3.2, log = FALSE)

dqkiener2(p, m = 0, g = 1, a = 3.2, w = 3.2, log = FALSE)

lkiener2(x, m = 0, g = 1, a = 3.2, w = 3.2)

dlkiener2(lp, m = 0, g = 1, a = 3.2, w = 3.2, log = FALSE)

qlkiener2(lp, m = 0, g = 1, a = 3.2, w = 3.2, lower.tail = TRUE)
```

## Arguments

| | |
|---|---|
| x | vector of quantiles. |
| q | vector of quantiles. |
| m | numeric. The median. |
| g | numeric. The scale parameter, preferably strictly positive. |
| a | numeric. The left tail parameter, preferably strictly positive. |
| w | numeric. The right tail parameter, preferably strictly positive. |
| p | vector of probabilities. |
| lp | vector of logit of probabilities. |
| n | number of observations. If length(n) > 1, the length is taken to be the number required. |
| log | logical; if TRUE, densities are given in log scale. |
| log.p | logical; if TRUE, probabilities p are given as log(p). |
| lower.tail | logical; if TRUE (default), probabilities are P[X < x] otherwise, P[X > x]. |

## Details

Kiener distributions of type II are distributions with asymmetric left and right fat tails described by the parameters a (alpha) for the left tail and w (omega) for the right tail. These parameters correspond to the power exponent that appear in Pareto formula and Karamata theorems.

As a and w are highly correlated, the use of Kiener distributions of type III (k and d) or type IV (k and e) is an alternate solution.

m is the median of the distribution. g is the scale parameter and the inverse of the density at the median: $g = 1/8/f(m)$. As a first estimate, it is approximatively one fourth of the standard deviation $g \approx \sigma/4$ but is independant from it.

The d, p functions have no explicit forms. They are provided here for convenience. They are estimated from a reverse optimization on the quantile function and can be (very) slow, depending the number of points to estimate. We recommand to use the quantile function as far as possible. WARNING: Results may become inconsistent when a or w are smaller than 1. Hopefully, this case seldom happens in finance.

qkiener2 function is defined for p in (0, 1) by:

$$qkiener2(p, m, g, a, w) = m + g * k * (-exp(-logit(p)/a) + exp(logit(p)/w))$$

where k is the harmonic mean of the tail parameters a and w calculated by $k = aw2k(a, w)$.

rkiener2 generates n random quantiles.

In addition to the classical d, p, q, r functions, the prefixes dp, dq, l, dl, ql are also provided.

dpkiener2 is the density function calculated from the probability p. It is defined for p in (0, 1) by:

$$dpkiener2(p, m, g, a, w) = p * (1 - p)/k/g/(exp(-logit(p)/a)/a + exp(logit(p)/w)/w$$

dqkiener2 is the derivate of the quantile function calculated from the probability p. It is defined for p in (0, 1) by:

$$dqkiener2(p, m, g, a, w) = k * g/p/(1 - p) * (exp(-logit(p)/a)/a + exp(logit(p)/w)/w)$$

lkiener2 function is estimated from a reverse optimization and can be (very) slow depending the number of points to estimate. Initialization is done by assuming a symmetric distribution lkiener1 around the harmonic mean k, then optimization is performed to take into account the true values a and w. The result can be then compared to the empirical probability logit(p). WARNING: Results may become inconsistent when a or w are smaller than 1. Hopefully, this case seldom happens in finance.

dlkiener2 is the density function calculated from the logit of the probability lp = logit(p). it is defined for lp in (-Inf, +Inf) by:

$$dlkiener2(lp, m, g, a, w) = p * (1 - p)/k/g/(exp(-lp/a)/a + exp(lp/w)/w)$$

qlkiener2 is the quantile function calculated from the logit of the probability. It is defined for lp in (-Inf, +Inf) by:

$$qlkiener2(lp, m, g, a, w) = m + g * k * (-exp(-lp/a) + exp(lp/w))$$

## References

P. Kiener, Explicit models for bilateral fat-tailed distributions and applications in finance with the package FatTailsR, 8th R/Rmetrics Workshop and Summer School, Paris, 27 June 2014. Download it from: http://www.inmodelia.com/exemples/2014-0627-Rmetrics-Kiener-en.pdf

## See Also

The symmetric Kiener distribution of type I kiener1, the asymmetric Kiener distributions of type III and IV kiener3, kiener4, the conversion functions aw2k, the regression function regkienerLX.

## Examples

```
require(graphics)

### Example 1
pp <- c(ppoints(11, a = 1), NA, NaN) ; pp
lp <- logit(pp) ; lp
qkiener2(  p = pp, m = 2, g = 1.5, a = 4, w = 6)
qkiener2(  p = pp, m = 2, g = 1.5, a = 4, w = 6)
qlkiener2(lp = lp, m = 2, g = 1.5, a = 4, w = 6)
dpkiener2( p = pp, m = 2, g = 1.5, a = 4, w = 6)
dlkiener2(lp = lp, m = 2, g = 1.5, a = 4, w = 6)
dqkiener2( p = pp, m = 2, g = 1.5, a = 4, w = 6)


### Example 2
a       <- 6
w       <- 4
set.seed(2014)
mainTC  <- paste("qkiener2(p, m = 0, g = 1, a = ", a, ", w = ", w, ")")
mainsum <- paste("cumulated qkiener2(p, m = 0, g = 1, a = ", a, ", w = ", w, ")")
T       <- 500
C       <- 4
TC      <- qkiener2(p = runif(T*C), m = 0, g = 1, a = a, w = w)
matTC   <- matrix(TC, nrow = T, ncol = C, dimnames = list(1:T, letters[1:C]))
head(matTC)
plot.ts(matTC, main = mainTC)
#
matsum  <- apply(matTC, MARGIN=2, cumsum)
head(matsum)
plot.ts(matsum, plot.type = "single", main = mainsum)
### End example 2


### Example 3 (four plots: probability, density, logit, logdensity)
x     <- q  <- seq(-15, 15, length.out=101)
w     <- c(0.6, 1, 1.5, 2, 3.2, 10) ; names(w) <- w
olty  <- c(2, 1, 2, 1, 2, 1, 1)
olwd  <- c(1, 1, 2, 2, 3, 3, 2)
ocol  <- c(2, 2, 4, 4, 3, 3, 1)
lleg  <- c("logit(0.999) = 6.9", "logit(0.99)   = 4.6", "logit(0.95)   = 2.9",
           "logit(0.50)   = 0", "logit(0.05)   = -2.9", "logit(0.01)   = -4.6",
           "logit(0.001) = -6.9  ")
op    <- par(mfrow=c(2,2), mgp=c(1.5,0.8,0), mar=c(3,3,2,1))

plot(x, plogis(x, scale = 2), type = "n", lwd = 2, ylim = c(0, 1),
     xaxs = "i", yaxs = "i", xlab = "", ylab = "",
     main = "pkiener2(q, m, g, a=2, w=...)")
for (i in 1:length(w)) lines(x, pkiener2(x, a = 2, w = w[i]),
       lty = olty[i], lwd = olwd[i], col = ocol[i] )
legend("topleft", title = expression(omega), legend = c(w),
       cex = 0.7, inset = 0.02, lty = olty, lwd = olwd, col = ocol )
```

```
plot(x, dlogis(x, scale = 2), type = "n", lwd = 2, ylim = c(0, 0.17),
     xaxs = "i", yaxs = "i", xlab = "", ylab = "",
     main = "dkiener2(q, m, g, a=2, w=...)")
for (i in 1:length(w)) lines(x, dkiener2(x, a = 2, w = w[i]),
       lty = olty[i], lwd = olwd[i], col = ocol[i] )
legend("topright", title = expression(omega), legend = c(w),
       cex = 0.7, inset = 0.02, lty = olty, lwd = olwd, col = ocol )

plot(x, x/2, type = "n", lwd = 1, ylim = c(-7.5, 7.5), yaxt="n", xaxs = "i",
     yaxs = "i", xlab = "", ylab = "",
     main = "logit(pkiener2(q, m, g, a=2, w=...))")
axis(2, las=1, at=c(-6.9, -4.6, -2.9, 0, 2.9, 4.6, 6.9) )
for (i in 1:length(w)) lines(x, lkiener2(x, a = 2, w = w[i]),
       lty = olty[i], lwd = olwd[i], col = ocol[i] )
legend("topleft", legend = lleg, cex = 0.7, inset = 0.02 )
legend("bottomright", title = expression(omega), legend = c(w),
       cex = 0.7, inset = 0.02, lty = c(olty), lwd = c(olwd), col = c(ocol) )

plot(x, dlogis(x, scale = 2, log=TRUE), type = "n", lwd = 2, ylim = c(-8, -1.5),
     xaxs = "i", yaxs = "i", xlab = "", ylab = "",
     main = "log(dkiener2(q, m, g, a=2, w=...))")
for (i in 1:length(w)) lines(x, dkiener2(x, a = 2, w = w[i], log=TRUE),
       lty = olty[i], lwd = olwd[i], col = ocol[i] )
legend("bottom", title = expression(omega), legend = c(w),
       cex = 0.7, inset = 0.02, lty = olty, lwd = olwd, col = ocol )
### End example 3


### Example 4 (four plots: quantile, derivate, density and quantiles from p)
p     <- ppoints(199, a=0)
w     <- c(0.6, 1, 1.5, 2, 3.2, 10) ; names(w) <- w ; w
op    <- par(mfrow=c(2,2), mgp=c(1.5,0.8,0), mar=c(3,3,2,1))

plot(p, qlogis(p, scale = 2), type = "l", lwd = 2, xlim = c(0, 1),
     ylim = c(-15, 15), xaxs = "i", yaxs = "i", xlab = "", ylab = "",
     main = "qkiener2(p, m, g, a=2, w=...)")
for (i in 1:length(w)) lines(p, qkiener2(p, a = 2, w = w[i]),
          lty = olty[i], lwd = olwd[i], col = ocol[i] )
legend("topleft", title = expression(omega), legend = c(w, "qlogis(x/2)"),
          inset = 0.02, lty = olty, lwd = olwd, col = ocol, cex = 0.7 )

plot(p, 2/p/(1-p), type = "l", lwd = 2, xlim = c(0, 1), ylim = c(0, 100),
     xaxs = "i", yaxs = "i", xlab = "", ylab = "",
     main = "dqkiener2(p, m, g, a=2, w=...)")
for (i in 1:length(w)) lines(p, dqkiener2(p, a = 2, w = w[i]),
          lty = olty[i], lwd = olwd[i], col = ocol[i] )
legend("top", title = expression(omega), legend = c(w, "p*(1-p)/2"),
          inset = 0.02, lty = olty, lwd = olwd, col = ocol, cex = 0.7 )

plot(qlogis(p, scale = 2), p*(1-p)/2, type = "l", lwd = 2, xlim = c(-15, 15),
     ylim = c(0, 0.18), xaxs = "i", yaxs = "i", xlab = "", ylab = "",
     main = "qkiener2, dpkiener2(p, m, g, a=2, w=...)")
for (i in 1:length(w)) {
```

```
     lines(qkiener2(p, a = 2, w = w[i]), dpkiener2(p, a = 2, w = w[i]),
           lty = olty[i], lwd = olwd[i], col = ocol[i] ) }
legend("topleft", title = expression(omega), legend = c(w, "p*(1-p)/2"),
          inset = 0.02, lty = olty, lwd = olwd, col = ocol, cex = 0.7 )

plot(qlogis(p, scale = 2), p, type = "l", lwd = 2, xlim = c(-15, 15),
     ylim = c(0, 1), xaxs = "i", yaxs = "i", xlab = "", ylab = "",
     main = "inverse axis qkiener2(p, m, g, a=2, w=...)")
for (i in 1:length(w)) lines(qkiener2(p, a = 2, w = w[i]), p,
          lty = olty[i], lwd = olwd[i], col = ocol[i] )
legend("topleft", title = expression(omega), legend = c(w, "qlogis(x/2)"),
          inset = 0.02, lty = olty, lwd = olwd, col = ocol, cex = 0.7 )
### End example 4
```

---

kiener3 *Asymmetric Kiener Distribution (Type III)*

---

## Description

Density, distribution function, quantile function, random generation and additional formulae for the Kiener distribution of type III.

## Usage

```
dkiener3(x, m = 0, g = 1, k = 3.2, d = 0, log = FALSE)

pkiener3(q, m = 0, g = 1, k = 3.2, d = 0, lower.tail = TRUE,
  log.p = FALSE)

qkiener3(p, m = 0, g = 1, k = 3.2, d = 0, lower.tail = TRUE,
  log.p = FALSE)

rkiener3(n, m = 0, g = 1, k = 3.2, d = 0)

dpkiener3(p, m = 0, g = 1, k = 3.2, d = 0, log = FALSE)

dqkiener3(p, m = 0, g = 1, k = 3.2, d = 0, log = FALSE)

lkiener3(x, m = 0, g = 1, k = 3.2, d = 0)

dlkiener3(lp, m = 0, g = 1, k = 3.2, d = 0, log = FALSE)

qlkiener3(lp, m = 0, g = 1, k = 3.2, d = 0, lower.tail = TRUE)
```

## Arguments

| | |
|---|---|
| x | vector of quantiles. |
| q | vector of quantiles. |

| | |
|---|---|
| m | numeric. The median. |
| g | numeric. The scale parameter, preferably strictly positive. |
| k | numeric. The tail parameter, preferably strictly positive. |
| d | numeric. The distorsion parameter between left and right tails. |
| p | vector of probabilities. |
| lp | vector of logit of probabilities. |
| n | number of observations. If length(n) > 1, the length is taken to be the number required. |
| log | logical; if TRUE, densities are given in log scale. |
| log.p | logical; if TRUE, probabilities p are given as log(p). |
| lower.tail | logical; if TRUE (default), probabilities are P[X < x] otherwise, P[X > x]. |

## Details

Kiener distributions of type III are distributions with asymmetric left and right fat tails described by a global tail parameter k and a distorsion parameter d.

Distributions of type III (kiener3) with parameters k (kappa) and d (delta) and distributions of type IV (kiener4) with parameters k (kappa) and e (epsilon)) have been created to disantangle the parameters a (alpha) and w (omega) of distributions of type II (kiener2). The tiny difference between the types III and IV ($d = e/k$) has not yet been fully evaluated. Both should be tested at that moment.

k is the harmonic mean of a and w and represents a global tail parameter.

d is a distorsion parameter between the left tail parameter a and the right tail parameter w. It verifies the inequality: $-k < d < k$ (whereas e of type IV verifies $-1 < e < 1$). The conversion functions (see aw2k) are:

$$1/k = (1/a + 1/w)/2$$

$$d = (-1/a + 1/w)/2$$

$$1/a = 1/k - d$$

$$1/w = 1/k + d$$

d (and e) should be of the same sign than the skewness. A negative value $d < 0$ implies $a < w$ and indicates a left tail heavier than the right tail. A positive value $d > 0$ implies $a > w$ and a right tail heavier than the left tail.

m is the median of the distribution. g is the scale parameter and the inverse of the density at the median: $g = 1/8/f(m)$. As a first estimate, it is approximatively one fourth of the standard deviation $g \approx \sigma/4$ but is independant from it.

The d, p functions have no explicit forms. They are provided here for convenience. They are estimated from a reverse optimization on the quantile function and can be (very) slow, depending the number of points to estimate. We recommand to use the quantile function as far as possible. WARNING: Results may become inconsistent when k is smaller than 1 or for very large absolute values of d. Hopefully, this case seldom happens in finance.

qkiener3 function is defined for p in (0, 1) by:

$$qkiener3(p, m, g, k, d) = m + 2 * g * k * sinh(logit(p)/k) * exp(d * logit(p))$$

rkiener3 generates n random quantiles.

In addition to the classical d, p, q, r functions, the prefixes dp, dq, l, dl, ql are also provided.

dpkiener3 is the density function calculated from the probability p. The formula is adapted from type II. It is defined for p in (0, 1) by:

$$dpkiener3(p, m, g, k, d) = p * (1 - p)/k/g/(exp(-logit(p)/a)/a + exp(logit(p)/w)/w$$

with a and w defined from k and d with the formula presented above.

dqkiener3 is the derivate of the quantile function calculated from the probability p. The formula is adapted from type II. It is defined for p in (0, 1) by:

$$dqkiener3(p, m, g, k, d) = k * g/p/(1 - p) * (exp(-logit(p)/a)/a + exp(logit(p)/w)/w)$$

with a and w defined above.

lkiener3 function is estimated from a reverse optimization and can be (very) slow depending the number of points to estimate. Initialization is done with a symmetric distribution lkiener1 of parameter k (thus $d = 0$). Then optimization is performed to take into account the true value of d. The results can then be compared to the empirical probability logit(p). WARNING: Results may become inconsistent when k is smaller than 1 or for very large absolute values of d. Hopefully, this case seldom happens in finance.

dlkiener3 is the density function calculated from the logit of the probability lp = logit(p). The formula is adapted from type II. it is defined for lp in (-Inf, +Inf) by:

$$dlkiener3(lp, m, g, k, d) = p * (1 - p)/k/g/(exp(-lp/a)/a + exp(lp/w)/w)$$

with a and w defined above.

qlkiener3 is the quantile function calculated from the logit of the probability. It is defined for lp in (-Inf, +Inf) by:

$$qlkiener3(lp, m, g, k, d) = m + 2 * g * k * sinh(lp/k) * exp(d * lp)$$

## References

P. Kiener, Explicit models for bilateral fat-tailed distributions and applications in finance with the package FatTailsR, 8th R/Rmetrics Workshop and Summer School, Paris, 27 June 2014. Download it from: http://www.inmodelia.com/exemples/2014-0627-Rmetrics-Kiener-en.pdf

## See Also

The symmetric Kiener distribution of type I kiener1, the asymmetric Kiener distributions of type II and IV kiener2, kiener4, the conversion functions aw2k, the regression function regkienerLX.

## Examples

```
require(graphics)

### Example 1
pp <- c(ppoints(11, a = 1), NA, NaN) ; pp
lp <- logit(pp) ; lp
qkiener3(  p = pp, m = 2, g = 1.5, k = aw2k(4, 6), d = aw2d(4, 6))
qlkiener3(lp = lp, m = 2, g = 1.5, k = aw2k(4, 6), d = aw2d(4, 6))
dpkiener3( p = pp, m = 2, g = 1.5, k = aw2k(4, 6), d = aw2d(4, 6))
dlkiener3(lp = lp, m = 2, g = 1.5, k = aw2k(4, 6), d = aw2d(4, 6))
dqkiener3( p = pp, m = 2, g = 1.5, k = aw2k(4, 6), d = aw2d(4, 6))


### Example 2
k       <- 4.8
d       <- 0.042
set.seed(2014)
mainTC  <- paste("qkiener3(p, m = 0, g = 1, k = ", k, ", d = ", d, ")")
mainsum <- paste("cumulated qkiener3(p, m = 0, g = 1, k = ", k, ", d = ", d, ")")
T       <- 500
C       <- 4
TC      <- qkiener3(p = runif(T*C), m = 0, g = 1, k = k, d = d)
matTC   <- matrix(TC, nrow = T, ncol = C, dimnames = list(1:T, letters[1:C]))
head(matTC)
plot.ts(matTC, main = mainTC)
#
matsum  <- apply(matTC, MARGIN=2, cumsum)
head(matsum)
plot.ts(matsum, plot.type = "single", main = mainsum)
### End example 2


### Example 3 (four plots: probability, density, logit, logdensity)
x     <- q  <- seq(-15, 15, length.out=101)
k     <- 3.2
d     <- c(-0.1, -0.03, -0.01, 0.01, 0.03, 0.1) ; names(d) <- d
olty  <- c(2, 1, 2, 1, 2, 1, 1)
olwd  <- c(1, 1, 2, 2, 3, 3, 2)
ocol  <- c(2, 2, 4, 4, 3, 3, 1)
lleg  <- c("logit(0.999) = 6.9", "logit(0.99)   = 4.6", "logit(0.95)   = 2.9",
           "logit(0.50)   = 0", "logit(0.05)   = -2.9", "logit(0.01)   = -4.6",
           "logit(0.001) = -6.9  ")
op    <- par(mfrow=c(2,2), mgp=c(1.5,0.8,0), mar=c(3,3,2,1))

plot(x, pkiener3(x, k = 3.2, d = 0), type = "l", lwd = 3, ylim = c(0, 1),
     xaxs = "i", yaxs = "i", xlab = "", ylab = "",
     main = "pkiener3(q, m, g, k=3.2, d=...)")
for (i in 1:length(d)) lines(x, pkiener3(x, k = 3.2, d = d[i]),
       lty = olty[i], lwd = olwd[i], col = ocol[i] )
legend("topleft", title = expression(delta), legend = c(d, "0"),
       cex = 0.7, inset = 0.02, lty = olty, lwd = olwd, col = ocol )
```

```
plot(x, dkiener3(x, k = 3.2, d = 0), type = "l", lwd = 3, ylim = c(0, 0.14),
     xaxs = "i", yaxs = "i", xlab = "", ylab = "",
     main = "dkiener3(q, m, g, k=3.2, d=...)")
for (i in 1:length(d)) lines(x, dkiener3(x, k = 3.2, d = d[i]),
       lty = olty[i], lwd = olwd[i], col = ocol[i] )
legend("topright", title = expression(delta), legend = c(d, "0"),
       cex = 0.7, inset = 0.02, lty = olty, lwd = olwd, col = ocol )

plot(x, lkiener3(x, k = 3.2, d = 0), type = "l", lwd =3, ylim = c(-7.5, 7.5),
     yaxt="n", xaxs = "i", yaxs = "i", xlab = "", ylab = "",
     main = "logit(pkiener3(q, m, g, k=3.2, d=...))")
axis(2, las=1, at=c(-6.9, -4.6, -2.9, 0, 2.9, 4.6, 6.9) )
for (i in 1:length(d)) lines(x, lkiener3(x, k = 3.2, d = d[i]),
       lty = olty[i], lwd = olwd[i], col = ocol[i] )
legend("topleft", legend = lleg, cex = 0.7, inset = 0.02 )
legend("bottomright", title = expression(delta), legend = c(d, "0"),
       cex = 0.7, inset = 0.02, lty = c(olty), lwd = c(olwd), col = c(ocol) )

plot(x, dkiener3(x, k = 3.2, d = 0, log = TRUE), type = "l", lwd = 3,
     ylim = c(-8, -1.5), xaxs = "i", yaxs = "i", xlab = "", ylab = "",
     main = "log(dkiener3(q, m, g, k=2, d=...))")
for (i in 1:length(d)) lines(x, dkiener3(x, k = 3.2, d = d[i], log=TRUE),
       lty = olty[i], lwd = olwd[i], col = ocol[i] )
legend("bottom", title = expression(delta), legend = c(d, "0"),
       cex = 0.7, inset = 0.02, lty = olty, lwd = olwd, col = ocol )
### End example 3


### Example 4 (four plots: quantile, derivate, density and quantiles from p)
p     <- ppoints(199, a=0)
d     <- c(-0.1, -0.03, -0.01, 0.01, 0.03, 0.1) ; names(d) <- d
op    <- par(mfrow=c(2,2), mgp=c(1.5,0.8,0), mar=c(3,3,2,1))

plot(p, qlogis(p, scale = 2), type = "l", lwd = 2, xlim = c(0, 1),
      ylim = c(-15, 15), xaxs = "i", yaxs = "i", xlab = "", ylab = "",
     main = "qkiener3(p, m, g, k=3.2, d=...)")
for (i in 1:length(d)) lines(p, qkiener3(p, k = 3.2, d = d[i]),
          lty = olty[i], lwd = olwd[i], col = ocol[i] )
legend("topleft", title = expression(delta), legend = c(d, "qlogis(x/2)"),
          inset = 0.02, lty = olty, lwd = olwd, col = ocol, cex = 0.7 )

plot(p, 2/p/(1-p), type = "l", lwd = 2, xlim = c(0, 1), ylim = c(0, 100),
     xaxs = "i", yaxs = "i", xlab = "", ylab = "",
     main = "dqkiener3(p, m, g, k=3.2, d=...)")
for (i in 1:length(d)) lines(p, dqkiener3(p, k = 3.2, d = d[i]),
          lty = olty[i], lwd = olwd[i], col = ocol[i] )
legend("top", title = expression(delta), legend = c(d, "p*(1-p)/2"),
          inset = 0.02, lty = olty, lwd = olwd, col = ocol, cex = 0.7 )

plot(qlogis(p, scale = 2), p*(1-p)/2, type = "l", lwd = 2, xlim = c(-15, 15),
     ylim = c(0, 0.14), xaxs = "i", yaxs = "i", xlab = "", ylab = "",
     main = "qkiener3, dpkiener3(p, m, g, k=3.2, d=...)")
for (i in 1:length(d)) {
```

```
     lines(qkiener3(p, k = 3.2, d = d[i]), dpkiener3(p, k = 3.2, d = d[i]),
           lty = olty[i], lwd = olwd[i], col = ocol[i] ) }
legend("topleft", title = expression(delta), legend = c(d, "p*(1-p)/2"),
          inset = 0.02, lty = olty, lwd = olwd, col = ocol, cex = 0.7 )

plot(qlogis(p, scale = 2), p, type = "l", lwd = 2, xlim = c(-15, 15),
     ylim = c(0, 1), xaxs = "i", yaxs = "i", xlab = "", ylab = "",
     main = "inverse axis qkiener3(p, m, g, k=3.2, d=...)")
for (i in 1:length(d)) lines(qkiener3(p, k = 3.2, d = d[i]), p,
          lty = olty[i], lwd = olwd[i], col = ocol[i] )
legend("topleft", title = expression(delta), legend = c(d, "qlogis(x/2)"),
          inset = 0.02, lty = olty, lwd = olwd, col = ocol, cex = 0.7 )
### End example 4
```

---

kiener4 *Asymmetric Kiener Distribution (Type IV)*

---

## Description

Density, distribution function, quantile function, random generation and additional formulae for the Kiener distribution of type IV.

## Usage

```
dkiener4(x, m = 0, g = 1, k = 3.2, e = 0, log = FALSE)

pkiener4(q, m = 0, g = 1, k = 3.2, e = 0, lower.tail = TRUE,
  log.p = FALSE)

qkiener4(p, m = 0, g = 1, k = 3.2, e = 0, lower.tail = TRUE,
  log.p = FALSE)

rkiener4(n, m = 0, g = 1, k = 3.2, e = 0)

dpkiener4(p, m = 0, g = 1, k = 3.2, e = 0, log = FALSE)

dqkiener4(p, m = 0, g = 1, k = 3.2, e = 0, log = FALSE)

lkiener4(x, m = 0, g = 1, k = 3.2, e = 0)

dlkiener4(lp, m = 0, g = 1, k = 3.2, e = 0, log = FALSE)

qlkiener4(lp, m = 0, g = 1, k = 3.2, e = 0, lower.tail = TRUE)
```

## Arguments

| | |
|---|---|
| x | vector of quantiles. |
| q | vector of quantiles. |

| | |
|---|---|
| m | numeric. The median. |
| g | numeric. The scale parameter, preferably strictly positive. |
| k | numeric. The tail parameter, preferably strictly positive. |
| e | numeric. The eccentricity parameter between left and right tails. |
| p | vector of probabilities. |
| lp | vector of logit of probabilities. |
| n | number of observations. If length(n) > 1, the length is taken to be the number required. |
| log | logical; if TRUE, densities are given in log scale. |
| log.p | logical; if TRUE, probabilities p are given as log(p). |
| lower.tail | logical; if TRUE (default), probabilities are P[X < x] otherwise, P[X > x]. |

## Details

Kiener distributions of type IV are distributions with asymmetric left and right fat tails described by a global tail parameter k and an eccentricity parameter e.

Distributions of type III (kiener3) with parameters k (kappa) and d (delta) and distributions of type IV (kiener4) with parameters k (kappa) and e (epsilon)) have been created to disantangle the parameters a (alpha) and w (omega) of distributions of type II (kiener2). The tiny difference between the types III and IV ($d = e/k$) has not yet been fully evaluated. Both should be tested at that moment.

k is the harmonic mean of a and w and represents a global tail parameter.

e is an eccentricity parameter between the left tail parameter a and the right tail parameter w. It verifies the inequality: $-1 < e < 1$ (whereas d of type III verifies $-k < d < k$). The conversion functions (see aw2k) are:

$$1/k = (1/a + 1/w)/2$$

$$e = (a - w)/(a + w)$$

$$a = k/(1 - e)$$

$$w = k/(1 + e)$$

e (and d) should be of the same sign than the skewness. A negative value $e < 0$ implies $a < w$ and indicates a left tail heavier than the right tail. A positive value $e > 0$ implies $a > w$ and a right tail heavier than the left tail.

m is the median of the distribution. g is the scale parameter and the inverse of the density at the median: $g = 1/8/f(m)$. As a first estimate, it is approximatively one fourth of the standard deviation $g \approx \sigma/4$ but is independant from it.

The d, p functions have no explicit forms. They are provided here for convenience. They are estimated from a reverse optimization on the quantile function and can be (very) slow, depending the number of points to estimate. We recommand to use the quantile function as far as possible. WARNING: Results may become inconsistent when k is smaller than 1 or for very large absolute values of e. Hopefully, these cases seldom happen in finance.

qkiener4 function is defined for p in (0, 1) by:

$$qkiener4(p, m, g, k, e) = m + 2 * g * k * sinh(logit(p)/k) * exp(e/k * logit(p))$$

rkiener4 generates n random quantiles.

In addition to the classical d, p, q, r functions, the prefixes dp, dq, l, dl, ql are also provided.

dpkiener4 is the density function calculated from the probability p. The formula is adapted from type II. It is defined for p in (0, 1) by:

$$dpkiener4(p, m, g, k, e) = p * (1 - p)/k/g/(exp(-logit(p)/a)/a + exp(logit(p)/w)/w$$

with a and w defined from k and e.

dqkiener4 is the derivate of the quantile function calculated from the probability p. The formula is adapted from type II. It is defined for p in (0, 1) by:

$$dqkiener4(p, m, g, k, e) = k * g/p/(1 - p) * (exp(-logit(p)/a)/a + exp(logit(p)/w)/w)$$

with a and w defined with the formula presented above.

lkiener4 function is estimated from a reverse optimization and can be (very) slow depending the number of points to estimate. Initialization is done with a symmetric distribution lkiener1 of parameter k (thus $e = 0$). Then optimization is performed to take into account the true value of e. The results can then be compared to the empirical probability logit(p). WARNING: Results may become inconsistent when k is smaller than 1 or for very large absolute values of e. Hopefully, these cases seldom happen in finance.

dlkiener4 is the density function calculated from the logit of the probability lp = logit(p). The formula is adapted from type II. it is defined for lp in (-Inf, +Inf) by:

$$dlkiener4(lp, m, g, k, e) = p * (1 - p)/k/g/(exp(-lp/a)/a + exp(lp/w)/w)$$

with a and w defined above.

qlkiener4 is the quantile function calculated from the logit of the probability. It is defined for lp in (-Inf, +Inf) by:

$$qlkiener4(lp, m, g, k, e) = m + 2 * g * k * sinh(lp/k) * exp(e/k * lp)$$

## References

P. Kiener, Explicit models for bilateral fat-tailed distributions and applications in finance with the package FatTailsR, 8th R/Rmetrics Workshop and Summer School, Paris, 27 June 2014. Download it from: http://www.inmodelia.com/exemples/2014-0627-Rmetrics-Kiener-en.pdf

## See Also

The symmetric Kiener distribution of type I kiener1, the asymmetric Kiener distributions of type II and III kiener2, kiener3, the conversion functions aw2k,

## Examples

```
require(graphics)

### Example 1
pp <- c(ppoints(11, a = 1), NA, NaN) ; pp
lp <- logit(pp) ; lp
qkiener4(  p = pp, m = 2, g = 1.5, k = aw2k(4, 6), e = aw2e(4, 6))
qlkiener4(lp = lp, m = 2, g = 1.5, k = aw2k(4, 6), e = aw2e(4, 6))
dpkiener4( p = pp, m = 2, g = 1.5, k = aw2k(4, 6), e = aw2e(4, 6))
dlkiener4(lp = lp, m = 2, g = 1.5, k = aw2k(4, 6), e = aw2e(4, 6))
dqkiener4( p = pp, m = 2, g = 1.5, k = aw2k(4, 6), e = aw2e(4, 6))


### Example 2
k       <- 4.8
e       <- 0.2
set.seed(2014)
mainTC  <- paste("qkiener4(p, m = 0, g = 1, k = ", k, ", e = ", e, ")")
mainsum <- paste("cumulated qkiener4(p, m = 0, g = 1, k = ", k, ", e = ", e, ")")
T       <- 500
C       <- 4
TC      <- qkiener4(p = runif(T*C), m = 0, g = 1, k = k, e = e)
matTC   <- matrix(TC, nrow = T, ncol = C, dimnames = list(1:T, letters[1:C]))
head(matTC)
plot.ts(matTC, main = mainTC)
#
matsum  <- apply(matTC, MARGIN=2, cumsum)
head(matsum)
plot.ts(matsum, plot.type = "single", main = mainsum)
### End example 2


### Example 3 (four plots: probability, density, logit, logdensity)
x     <- q  <- seq(-15, 15, length.out=101)
k     <- 3.2
e     <- c(-0.3, -0.15, -0.07, 0.07, 0.15, 0.30) ; names(e) <- e
olty  <- c(2, 1, 2, 1, 2, 1, 1)
olwd  <- c(1, 1, 2, 2, 3, 3, 2)
ocol  <- c(2, 2, 4, 4, 3, 3, 1)
lleg  <- c("logit(0.999) = 6.9", "logit(0.99)   = 4.6", "logit(0.95)   = 2.9",
           "logit(0.50)   = 0", "logit(0.05)   = -2.9", "logit(0.01)   = -4.6",
           "logit(0.001) = -6.9  ")
op    <- par(mfrow=c(2,2), mgp=c(1.5,0.8,0), mar=c(3,3,2,1))

plot(x, pkiener4(x, k = 3.2, e = 0), type = "l", lwd = 3, ylim = c(0, 1),
     xaxs = "i", yaxs = "i", xlab = "", ylab = "",
     main = "pkiener4(q, m, g, k=3.2, e=...)")
for (i in 1:length(e)) lines(x, pkiener4(x, k = 3.2, e = e[i]),
       lty = olty[i], lwd = olwd[i], col = ocol[i] )
legend("topleft", title = expression(epsilon), legend = c(e, "0"),
       cex = 0.7, inset = 0.02, lty = olty, lwd = olwd, col = ocol )
```

```
plot(x, dkiener4(x, k = 3.2, e = 0), type = "l", lwd = 3, ylim = c(0, 0.14),
     xaxs = "i", yaxs = "i", xlab = "", ylab = "",
     main = "dkiener4(q, m, g, k=3.2, e=...)")
for (i in 1:length(e)) lines(x, dkiener4(x, k = 3.2, e = e[i]),
       lty = olty[i], lwd = olwd[i], col = ocol[i] )
legend("topright", title = expression(epsilon), legend = c(e, "0"),
       cex = 0.7, inset = 0.02, lty = olty, lwd = olwd, col = ocol )

plot(x, lkiener4(x, k = 3.2, e = 0), type = "l", lwd =3, ylim = c(-7.5, 7.5),
     yaxt="n", xaxs = "i", yaxs = "i", xlab = "", ylab = "",
     main = "logit(pkiener4(q, m, g, k=3.2, e=...))")
axis(2, las=1, at=c(-6.9, -4.6, -2.9, 0, 2.9, 4.6, 6.9) )
for (i in 1:length(e)) lines(x, lkiener4(x, k = 3.2, e = e[i]),
       lty = olty[i], lwd = olwd[i], col = ocol[i] )
legend("topleft", legend = lleg, cex = 0.7, inset = 0.02 )
legend("bottomright", title = expression(epsilon), legend = c(e, "0"),
       cex = 0.7, inset = 0.02, lty = c(olty), lwd = c(olwd), col = c(ocol) )

plot(x, dkiener4(x, k = 3.2, e = 0, log = TRUE), type = "l", lwd = 3,
     ylim = c(-8, -1.5), xaxs = "i", yaxs = "i", xlab = "", ylab = "",
     main = "log(dkiener4(q, m, g, k=2, e=...))")
for (i in 1:length(e)) lines(x, dkiener4(x, k = 3.2, e = e[i], log=TRUE),
       lty = olty[i], lwd = olwd[i], col = ocol[i] )
legend("bottom", title = expression(epsilon), legend = c(e, "0"),
       cex = 0.7, inset = 0.02, lty = olty, lwd = olwd, col = ocol )
### End example 3


### Example 4 (four plots: quantile, derivate, density and quantiles from p)
p     <- ppoints(199, a=0)
e     <- c(-0.3, -0.15, -0.07, 0.07, 0.15, 0.30) ; names(e) <- e
op    <- par(mfrow=c(2,2), mgp=c(1.5,0.8,0), mar=c(3,3,2,1))

plot(p, qlogis(p, scale = 2), type = "l", lwd = 2, xlim = c(0, 1),
     ylim = c(-15, 15), xaxs = "i", yaxs = "i", xlab = "", ylab = "",
     main = "qkiener4(p, m, g, k=3.2, e=...)")
for (i in 1:length(e)) lines(p, qkiener4(p, k = 3.2, e = e[i]),
          lty = olty[i], lwd = olwd[i], col = ocol[i] )
legend("topleft", title = expression(epsilon), legend = c(e, "qlogis(x/2)"),
        inset = 0.02, lty = olty, lwd = olwd, col = ocol, cex = 0.7 )

plot(p, 2/p/(1-p), type = "l", lwd = 2, xlim = c(0, 1), ylim = c(0, 100),
     xaxs = "i", yaxs = "i", xlab = "", ylab = "",
     main = "dqkiener4(p, m, g, k=3.2, e=...)")
for (i in 1:length(e)) lines(p, dqkiener4(p, k = 3.2, e = e[i]),
          lty = olty[i], lwd = olwd[i], col = ocol[i] )
legend("top", title = expression(epsilon), legend = c(e, "p*(1-p)/2"),
          inset = 0.02, lty = olty, lwd = olwd, col = ocol, cex = 0.7 )

plot(qlogis(p, scale = 2), p*(1-p)/2, type = "l", lwd = 2, xlim = c(-15, 15),
     ylim = c(0, 0.14), xaxs = "i", yaxs = "i", xlab = "", ylab = "",
     main = "qkiener4, dpkiener4(p, m, g, k=3.2, e=...)")
for (i in 1:length(e)) {
```

```
     lines(qkiener4(p, k = 3.2, e = e[i]), dpkiener4(p, k = 3.2, e = e[i]),
           lty = olty[i], lwd = olwd[i], col = ocol[i] ) }
legend("topleft", title = expression(epsilon), legend = c(e, "p*(1-p)/2"),
          inset = 0.02, lty = olty, lwd = olwd, col = ocol, cex = 0.7 )

plot(qlogis(p, scale = 2), p, type = "l", lwd = 2, xlim = c(-15, 15),
     ylim = c(0, 1), xaxs = "i", yaxs = "i", xlab = "", ylab = "",
     main = "inverse axis qkiener4(p, m, g, k=3.2, e=...)")
for (i in 1:length(e)) lines(qkiener4(p, k = 3.2, e = e[i]), p,
          lty = olty[i], lwd = olwd[i], col = ocol[i] )
legend("topleft", title = expression(epsilon), legend = c(e, "qlogis(x/2)"),
          inset = 0.02, lty = olty, lwd = olwd, col = ocol, cex = 0.7 )
### End example 4
```

---

`laplacegaussnorm` *Laplace-Gauss Normal Distribution Object*

---

## Description

An object designed after regkienerLX to summarize the information related to a given dataset when the Laplace-Gauss normal distribution is applied on it.

## Usage

```
laplacegaussnorm(X)
```

## Arguments

| | |
|---|---|
| `X` | vector of quantiles. |

## Details

This function is designed after regkienerLX to provide a similar framework.

## Value

| | |
|---|---|
| `dfrXPn` | data.frame. X = initial quantiles. Pn = estimated normal probabilites. |
| `dfrXLn` | data.frame. X = initial quantiles. Ln = logit of estimated normal probabilites. |
| `dfrXDn` | data.frame. X = initial quantiles. Dn = estimated normal density. |
| `coefn` | numeric. The mean and the standard deviation of the dataset. |
| `dfrQnPn` | data.frame. Qn = estimated quantiles of interest. Pn = probability. |
| `dfrQnPn` | data.frame. Qn = estimated quantiles of interest. Pn = logit of probability. |

## See Also

The regression function `regkienerLX`.

## Examples

```
prices2returns <- function(x) { 100*diff(log(x)) }
CAC  <- prices2returns(as.numeric(EuStockMarkets[,3]))
lgn  <- laplacegaussnorm( CAC )
attributes(lgn)
head(lgn$dfrXPn)
head(lgn$dfrXLn)
head(lgn$dfrXDn)
lgn$coefn
lgn$dfrQnPn
lgn$dfrQnLn
```

---

loghp *Inverse Power Hyperbolas and Inverse Power Hyperbolic Functions*

---

## Description

The inverse power hyperbolas and the inverse power hyperbolic functions: arc-cosine-hp, arc-sine-hp, arc-tangent-hp, arc-secant-hp, arc-cosecant-hp and arc-cotangent-hp.

## Usage

```
loghp(x, k = 1)

acoshp(x, k = 1)

asinhp(x, k = 1)

atanhp(x, k = 1)

asechp(x, k = 1)

acosechp(x, k = 1)

acotanhp(x, k = 1)
```

## Arguments

| | |
|---|---|
| x | a numeric value, vector or matrix. |
| k | a numeric value, preferably strictly positive. |

## Details

loghp function is defined on (0, +Inf) by:

$$logshp(x, k) = 2 * k * sinh(log(x)/k)$$

acoshp function is defined on [1, +Inf) by:

$$acoshp(x,k) = 2 * k * sinh(acosh(x)/k)$$

asinhp function is defined on (-Inf, +Inf) by:

$$asinhp(x,k) = 2 * k * sinh(asinh(x)/k)$$

atanhp function is defined on (-1, +1) by:

$$atanhp(x,k) = 2 * k * sinh(atanh(x)/k)$$

asechp function is defined on (0, +1] by:

$$asechp(x,k) = 2 * k * sinh(acosh(1/x)/k)$$

acosechp function is defined on (-Inf, 0) U (0, +Inf) by:

$$acosechp(x,k) = 2 * k * sinh(asinh(1/x)/k)$$

acotanhp function is defined on (-Inf, -1) U (1, +Inf) by:

$$acotanhp(x,k) = 2 * k * sinh(atanh(1/x)/k)$$

If k is a vector of length > 1, then the use of the function `outer` is recommanded.

## See Also

The power hyperbolic functions `exphp`.

## Examples

```
### Example 1 (acoshp, asinhp, atanhp)
 loghp( c(ppoints(10), 1, 1/rev(ppoints(10))), k = 2)
acoshp( 1:10, k = 2)
asinhp( -5:5, k = 2)
atanhp( seq(-1, 1, by = 0.1), k = 2)
asechp( ppoints(20), k = 2)
acosechp( -5:5, k = 2)
acotanhp( c( -1/ppoints(10), 1/rev(ppoints(10))), k = 2)

x  <- (-2:3)*3
 loghp(exphp(x, k = 4), k = 4)
acoshp(coshp(x, k = 4), k = 4)
asinhp(sinhp(x, k = 4), k = 4)
atanhp(tanhp(x, k = 4), k = 4)


### Example 2 (loghp, acoshp, asinhp, atanhp)
k     <- c(0.6, 1, 1.5, 2, 3.2, 10) ; names(k) <- k
olty  <- c(2, 1, 2, 1, 2, 1, 1)
olwd  <- c(1, 1, 2, 2, 3, 4, 2)
ocol  <- c(2, 2, 4, 4, 3, 3, 1)
```

```
op    <- par(mfrow = c(2, 2), mgp = c(1.5, 0.8, 0), mar = c(3, 3, 2, 1))

xld     <- 0.05
xl      <- seq(0.05, 20, xld) ; names(xl) <- xl
Tlcoshp <- ts(cbind(outer(xl, k, loghp), "2*log(x)" = 2*log(xl)),
              start = xl[1], deltat = xld)
plot(Tlcoshp, plot.type = "single", xlim = c(0,20), ylim = c(-5,15),
     lty = olty, lwd = olwd, col = ocol, xaxs = "i", yaxs = "i",
     xlab="", ylab = "", main = "loghp(x, k)" )
legend("bottomright", title = expression(kappa), legend = colnames(Tlcoshp),
     inset = 0.02, lty = olty, lwd = olwd, col = ocol, cex = 0.7 )

## acoshp(x, k)
xcd     <- 0.5
xc      <- seq(1, 20, xcd) ; names(xc) <- xc
Tacoshp <- ts(cbind(outer(xc, k, acoshp), "2*acosh(x)" = 2*acosh(xc)),
              start = xc[1], deltat = xcd)
plot(Tacoshp, plot.type = "single", ylim = c(0,15), lty = olty, lwd = olwd, col = ocol,
        xaxs = "i", yaxs = "i", xlab = "", ylab = "", main = "acoshp(x, k)" )
legend("bottomright", title = expression(kappa), legend = colnames(Tacoshp),
        inset = 0.02, lty = olty, lwd = olwd, col = ocol, cex = 0.7 )

## asinhp(x, k)
xsd     <- 0.5
xs      <- seq(-10, 10, xsd) ; names(xs) <- xs
Tasinhp <- ts(cbind(outer(xs, k, asinhp), "2*asinh(x)" = 2*asinh(xs)),
              start = xs[1], deltat = xsd)
plot(Tasinhp, plot.type = "single", ylim = c(-10,10), lty = olty, lwd = olwd, col = ocol,
        xaxs = "i", yaxs = "i", xlab = "", ylab = "", main = "asinhp(x, k)" )
legend("topleft", title = expression(kappa), legend = colnames(Tasinhp),
        inset = 0.02, lty = olty, lwd = olwd, col = ocol, cex = 0.7 )

## atanhp(x, k)
xtd     <- 0.01
xt      <- seq(-1, 1, xtd) ; names(xt) <- xt
Tatanhp <- ts(cbind(outer(xt, k, atanhp), "2*atanh(x)" = 2*atanh(xt)),
              start = xt[1], deltat = xtd)
plot(Tatanhp, plot.type = "single", ylim = c(-10,10), lty = olty, lwd = olwd, col = ocol,
        xaxs = "i", yaxs = "i", xlab = "", ylab = "", main = "atanhp(x, k)" )
legend("topleft", title = expression(kappa), legend = colnames(Tatanhp),
        inset = 0.02, lty = olty, lwd = olwd, col = ocol, cex = 0.7 )
### End Example 2
```

---

`logishp` *The Power Hyperbola Logistic Distribution*

---

## Description

Density, distribution function, quantile function and random generation for the power hyperbola logistic distribution.

## Usage

```
dlogishp(x, k = 1, log = FALSE)

plogishp(q, k = 1)

qlogishp(p, k = 1)

rlogishp(n, k = 1)

dplogishp(p, k = 1, log = FALSE)

dqlogishp(p, k = 1, log = FALSE)

llogishp(x, k = 1)

dllogishp(lp, k = 1, log = FALSE)

qllogishp(lp, k = 1)
```

## Arguments

| | |
|---|---|
| x | vector of quantiles. |
| q | vector of quantiles. |
| k | numeric. The tail parameter, preferably strictly positive. Can be a vector (see details). |
| p | vector of probabilities. |
| lp | vector of logit of probabilities. |
| n | number of observations. If length(n) > 1, the length is taken to be the number required. |
| log | boolean. |

## Details

dlogishp function (log is available) is defined for x in (-Inf, +Inf) by:

$$dlogishp(x,k) = dkashp\_dx(x,k) * plogishp(x,k) * plogishp(-x,k)$$

plogishp function is defined for q in (-Inf, +Inf) by:

$$plogishp(q,k) = 1/(1 + exp(-kashp(q,k)))$$

qlogishp function is defined for p in (0, 1) by:

$$qlogishp(p,k) = 2 * k * sinh(logit(p)/k)$$

rlogishp function generates n random values.

In addition to the classical formats, the prefixes dp, dq, l, dl, ql are also provided:

dplogishp function (log is available) is defined for p in (0, 1) by:

$$dplogishp(p, k = 1) = p * (1 - p)/2/cosh(logit(p)/k)$$

dqlogishp function (log is available) is defined for p in (0, 1) by:

$$dqlogishp(p, k = 1) = 2/p/(1 - p) * cosh(logit(p)/k)$$

llogishp function is defined for x in (-Inf, +Inf) by:

$$llogishp(x, k) = kashp(x, k)$$

dllogishp function is defined for lp = logit(p) in (-Inf, +Inf) by :

$$dllogishp(lp, k) = p * (1 - p)/2/cosh(lp/k)$$

qllogishp function is defined for lp = logit(p) in (-Inf, +Inf) by :

$$qllogishp(lp, k) = 2 * k * sinh(lp/k)$$

If k is a vector, then the use of the function outer is recommanded.

## See Also

The Kiener distribution of type I kiener1 which has location (m) and scale (g) parameters.

## Examples

```
require(graphics)

### Example 1
pp <- c(ppoints(11, a = 1), NA, NaN) ; pp
plogishp(-5:5, k = 4)
dlogishp(-5:5, k = 4)
qlogishp(pp, k = 4)
outer(-5:5, 1:6, plogishp)
outer(-5:5, 1:6, dlogishp)
outer(runif(20), 1:6, qlogishp)

### Example 2
x     <- seq(-15, 15, length.out = 101)
k     <- c(0.6, 1, 1.5, 2, 3.2, 10) ; names(k) <- k ; k
olty  <- c(2, 1, 2, 1, 2, 1, 1)
olwd  <- c(1, 1, 2, 2, 3, 4, 2)
ocol  <- c(2, 2, 4, 4, 3, 3, 1)
op    <- par(mfrow = c(2,2), mgp = c(1.5,0.8,0), mar = c(3,3,2,1))

plot(x, plogis(x, scale = 2), type = "b", lwd = 2, ylim = c(0, 1),
     xaxs = "i", yaxs = "i", xlab = "", ylab = "", main = "plogishp(x, k)")
for (i in 1:length(k)) lines(x, plogishp(x, k = k[i]),
          lty = olty[i], lwd = olwd[i], col = ocol[i] )
legend("topleft", title = expression(kappa), legend = c(k, "plogis(x/2)"),
          inset = 0.02, lty = olty, lwd = olwd, col = ocol, cex = 0.7 )
```

```
plot(x, dlogis(x, scale = 2), type = "b", lwd = 2, xaxs = "i",
     yaxs = "i", xlab = "", ylab = "", main = "dlogishp(x, k)")
for (i in 1:length(k)) lines(x, dlogishp(x, k = k[i]),
          lty = olty[i], lwd = olwd[i], col = ocol[i] )

plot(x, x/2, type = "b", lwd = 2, ylim = c(-7.5, 7.5), xaxs = "i",
     yaxs = "i", xlab = "", ylab = "", main = "logit(logishp(h, k))")
for (i in 1:length(k)) lines(x, llogishp(x, k = k[i]),
          lty = olty[i], lwd = olwd[i], col = ocol[i] )

plot(x, log(dlogis(x, scale = 2)), lwd = 2, type = "b", xaxs = "i",
     yaxs = "i", xlab = "", ylab = "", main = "log(dlogishp(x, k))")
for (i in 1:length(k)) lines(x, dlogishp(x, k = k[i], log = TRUE),
          lty = olty[i], lwd = olwd[i], col = ocol[i] )
### End example 2

### Example 3
p <- ppoints(199, a=0)
plot(p, qlogis(p, scale = 2), type = "o", lwd = 2, ylim = c(-15, 15),
     xaxs = "i", yaxs = "i", xlab = "", ylab = "",
     main = "qlogishp(p, k)")
for (i in 1:length(k)) lines(p, qlogishp(p, k = k[i]),
          lty = olty[i], lwd = olwd[i], col = ocol[i] )
legend("topleft", title = expression(kappa), legend = c(k, "qlogis(x/2)"),
          inset = 0.02, lty = olty, lwd = olwd, col = ocol, cex = 0.7 )

plot(p, 2/p/(1-p), type = "o", lwd = 2, xlim = c(0, 1), ylim = c(0, 100),
     xaxs = "i", yaxs = "i", xlab = "", ylab = "",
     main = "dqlogishp(p, k)")
for (i in 1:length(k)) lines(p, dqlogishp(p, k = k[i]),
          lty = olty[i], lwd = olwd[i], col = ocol[i] )
legend("top", title = expression(kappa), legend = c(k, "p*(1-p)/2"),
          inset = 0.02, lty = olty, lwd = olwd, col = ocol, cex = 0.7 )

plot(qlogis(p, scale = 2), p*(1-p)/2, type = "o", lwd = 2, xlim = c(-15, 15),
     ylim = c(0, 0.14), xaxs = "i", yaxs = "i", xlab = "", ylab = "",
     main = "qlogishp, dplogishp(p, k)")
for (i in 1:length(k)) lines(qlogishp(p, k = k[i]), dplogishp(p, k = k[i]),
          lty = olty[i], lwd = olwd[i], col = ocol[i] )
legend("topleft", title = expression(kappa), legend = c(k, "p*(1-p)/2"),
          inset = 0.02, lty = olty, lwd = olwd, col = ocol, cex = 0.7 )
### End example 3
```

---

`logit` *Logit Function*

---

## Description

The logit function, widely used in this package, is a wrapper of `qlogis`.

The invlogit function is the inverse of the logit function and a wrapper of `plogis`.

## Usage

```
logit(p)

invlogit(x)
```

## Arguments

| | |
|---|---|
| p | a numeric value or vector. |
| x | a numeric value or vector. |

## Details

`logit` function is defined for p in (0, 1) by:

$$logit(p) = log(p/(1-p))$$

`invlogit` function is defined for x in (-Inf, +Inf) by:

$$invlogit(x) = exp(x)/(1+exp(x)) = plogis(x)$$

## See Also

The equivalent function `qlogis`.

The equivalent function `plogis`.

## Examples

```
logit( c(ppoints(11, a = 1), NA, NaN) )
invlogit( c(-Inf, -10:10, +Inf, NA, NaN) )
```

---

`regkienerLX` *A Regression Function for Kiener Distributions*

---

## Description

A function to estimate the distribution parameters of a given dataset with Kiener distributions of type I, II, III and IV. It performs a nonlinear regression of the logit of the empirical probabilities logit(p) on quantiles X.

## Usage

```
regkienerLX(X, model = "k4", dgts = c(3, 3, 1, 1, 1, 3, 2, 4, 4, 2, 2),
  maxk = 10, mink = 0.7, app = 0)
```

## Arguments

| | |
|---|---|
| X | vector of quantiles. |
| model | the model used for the regression: "k1", "k2", "k3", "k4". |
| dgts | vector of length 11. Control the rounding of output parameters. |
| maxk | numeric. The maximum value of tail parameter k. |
| mink | numeric. The minimum value of tail parameter k. |
| app | numeric. The parameter "a" in the function ppoints. |

## Details

This function is designed to estimate the parameters of Kiener distributions for a given dataset. It encapsulates the four distributions described in this package. "k1" uses model lqkiener1, "k2" uses model lqkiener2, "k3" uses model lqkiener3 and "k4" uses model lqkiener4.

A typical input is a numeric vector that describes the returns of a stock. Conversion from a (possible) time series format to a sorted numeric vector is done automatically and without any check of the initial format. There is also no check of missing values, Na, NaN, -Inf, +Inf. Empirical probabilities of each point in the sorted dataset is calculated with the function ppoints. The parameter app corresponds to the parameter a in ppoints but has been limited to the range (0, 0.5). Default value is 0 as large datasets are very common in finance.

A nonlinear regression is performed with nlsLM from the logit of the probabilities logit(p) over the quantiles X with one of the functions lqkiener1234. These functions have been selected as they have an explicit form in the four types (this is unfortunately not the case for dkiener234) and return satisfactory results with ordinary least squares. The median is calculated before the regression and is injected as a mandatory value in the regression function.

Model "k1" return results with 1+2=3 parameters and describes a (assumed) symmetric distribution. Parameters d and e are set to 0. Models "k2", "k3" and "k4" describe asymmetric distributions. They return results with 1+3=4 parameters. Model "k2" has a very clear parameter definition but unfortunately parameters a and w are highly correlated. Model "k3" has the least correlated parameters but the meaning of the distorsion parameter d, usually of order 1e-3, is not simple. Multiplying it by 1000 might be a good choice but has not been done here.

Model "k4" exhibits a reasonable correlation between each parameter and should be the preferred intermediate model between "k1" and "k2" models. The eccentricity parameter e is well defined and easy to understand: $e = (a - w)/(a + w)$, $a = k/(1 - e)$ and $w = k/(1 + e)$. It is a relative measure that could be expressed (and understood) as a percentage. Here, the order 1e-2 has been preserved.

Tail parameter lower and upper values are controlled by maxk and mink. An upper value $maxk = 10$ is appropriate for datasets of low and medium size, less than 50.000 points. For larger datasets, the upper limit can be extended up to $maxk = 20$. Such a limit returns results which are very closed to the logistic distribution, an alternate distribution which could be more appropriate. The lower limit mink is intended to avoid the value $k = 0$. Remind that value $k < 2$ describes distribution with no stable variance and $k < 1$ describes distribution with no stable mean.

The output is an object in a flat format of class clregk. It can be listed with the function attributes. First are the data.frames with the initial data and the estimated results. Second is the result of the regression regk0 given by nlsLM from which a few information have been extracted and listed here. Third are the regression parameters (without the median) in plain format (no rounding), the

variance-covariance matrix, the variance-covariance matrix times 1e+6 and the correlation matrix in a rounded format. Note that `regk0`, `coefk0`, `coefk0tt`, `vcovk0`, `mcork0` have a polymorphic format and changing parameters that depend from the selected model: "k1", "k2", "k3", "k4". They should be used with care in subsequent calculations. Fourth are the distribution parameters tailored to every model "k1", "k2", "k3", "k4" plus estimated quantiles at levels: c(0.001, 0.005, 0.01, 0.05, 0.5, 0.95, 0.99, 0.995, 0.999). They are intended to subsequent calculations. Fifth are the same parameters presented in a more readable format thanks to the vector `dgts` which control the rounding of the parameters in the following order: c("m", "g", "a", "k", "w", "d", "e", "vcovk0", "vcovk0m", "mcork0", "quantr"). Sixth are the estimated quantiles and probabilities of interest stored in a data.frame format.

## Value

| | |
|---|---|
| `dfrXP` | data.frame. X = initial quantiles. P = empirical probabilities. |
| `dfrXL` | data.frame. X = initial quantiles. L = logit of probabilities. |
| `dfrXR` | data.frame. X = initial quantiles. R = residuals after regression. |
| `dfrEP` | data.frame. E = estimated quantiles. P = probabilities. |
| `dfrEL` | data.frame. E = estimated quantiles. L = logit of probabilities. |
| `dfrED` | data.frame. E = estimated quantiles. D = estimated density (from probabilities). |
| `regk0` | object of class `nls` extracted from the regression function `nlsLM`. |
| `coefk0` | the regression parameters in plain format. The median is out of the regression. |
| `vcovk0` | rounded variance-covariance matrix. |
| `vcovk0m` | rounded 1e+6 times variance-covariance matrix. |
| `mcork0` | rounded correlation matrix. |
| `coefk` | all parameters in plain format. |
| `coefk1` | parameters for model "k1". |
| `coefk2` | parameters for model "k2". |
| `coefk3` | parameters for model "k3". |
| `coefk4` | parameters for model "k4". |
| `quantk` | quantiles of interest. |
| `coefr` | all parameters in a rounded format. |
| `coefr1` | rounded parameters for model "k1". |
| `coefr2` | rounded parameters for model "k2". |
| `coefr3` | rounded parameters for model "k3". |
| `coefr4` | rounded parameters for model "k4". |
| `quantr` | quantiles of interest in a rounded format. |
| `dfrQkPk` | data.frame. Qk = Estimated quantiles of interest. Pk = probabilities. |
| `dfrQkLk` | data.frame. Qk = Estimated quantiles of interest. Lk = Logit of probabilities. |

## See Also

`nlsLM`, `laplacegaussnorm`, Kiener distributions of type I, II, III and IV: `kiener1` `kiener2`, `kiener3`, `kiener4`.

## Examples

```
require(graphics)
require(minpack.lm)
require(timeSeries)

prices2returns <- function(x) { 100*diff(log(x)) }

### Load the various datasets (1-16)
DS <- list(
"USDCHF"       = prices2returns(as.numeric(USDCHF)) ,
"Microsoft"    = prices2returns(as.numeric(MSFT[,4])) ,
"DAX"          = prices2returns(as.numeric(EuStockMarkets[,1])) ,
"SMI"          = prices2returns(as.numeric(EuStockMarkets[,2])) ,
"CAC"          = prices2returns(as.numeric(EuStockMarkets[,3])) ,
"FTSE"         = prices2returns(as.numeric(EuStockMarkets[,4])) ,
"SBI"          = as.numeric(LPP2005REC[,1]) ,
"SPI"          = as.numeric(LPP2005REC[,2]) ,
"SII"          = as.numeric(LPP2005REC[,3]) ,
"LMI"          = as.numeric(LPP2005REC[,4]) ,
"MPI"          = as.numeric(LPP2005REC[,5]) ,
"ALT"          = as.numeric(LPP2005REC[,6]) ,
"LPP25"        = as.numeric(LPP2005REC[,7]) ,
"LPP40"        = as.numeric(LPP2005REC[,8]) ,
"LPP60"        = as.numeric(LPP2005REC[,9]) ,
"sunspot.year" = prices2returns(as.numeric(sunspot.year)+1000) )


### Select one dataset number (1-16)
j      <- 5

### and run this block
X      <- DS[[j]]
nameX  <- names(DS)[j]
reg    <- regkienerLX(X)
lleg   <- c("logit(0.999) = 6.9", "logit(0.99)   = 4.6",
            "logit(0.95)   = 2.9", "logit(0.50)   = 0",
            "logit(0.05)   = -2.9", "logit(0.01)   = -4.6",
            "logit(0.001) = -6.9  ")
pleg   <- c( paste("m =",  reg$coefr4[1]), paste("g  =", reg$coefr4[2]),
              paste("k  =", reg$coefr4[3]), paste("e  =", reg$coefr4[4]) )
op     <- par(mfrow=c(2,2), mgp=c(1.5,0.8,0), mar=c(3,3,2,1))
plot(X, type="l", main = nameX)
plot(reg$dfrXL, main = nameX, yaxt = "n")
axis(2, las=1, at=c(-9.2, -6.9, -4.6, -2.9, 0, 2.9, 4.6, 6.9, 9.2))
abline(h = c(-4.6, 4.6), lty = 4)
abline(v = c(reg$quantk[5], reg$quantk[9]), lty = 4)
legend("topleft", legend = lleg, cex = 0.7, inset = 0.02, bg = "#FFFFFF")
lines(reg$dfrEL, col = 2, lwd = 2)
points(reg$dfrQkLk, pch = 3, col = 2, lwd = 2, cex = 1.5)
plot(reg$dfrXP, main = nameX)
legend("topleft", legend = pleg, cex = 0.9, inset = 0.02 )
lines(reg$dfrEP, col = 2, lwd = 2)
```

```
plot(density(X), main = nameX)
lines(reg$dfrED, col = 2, lwd = 2)
attributes(reg)
head(reg$dfrXP)
head(reg$dfrXL)
head(reg$dfrXR)
head(reg$dfrEP)
head(reg$dfrEL)
head(reg$dfrED)
reg$regk0
reg$coefk0
reg$vcovk0
reg$vcovk0m
reg$mcork0
reg$coefk
reg$coefk1
reg$coefk2
reg$coefk3
reg$coefk4
reg$quantk
reg$coefr
reg$coefr1
reg$coefr2
reg$coefr3
reg$coefr4
reg$quantr
reg$dfrQkPk
reg$dfrQkLk
### End block
```

# Index