Package ‘TukeyGH77’

March 23, 2024

Type Package

Title Tukey g-&-h Distribution

Version 0.1.1

Date 2024-03-22

Description Functions for density, cumulative density, quantile and simulation of Tukey g-and-h (1977) distributions. The quantile-based transformation (Hoaglin 1985 <doi:10.1002/9781118150702.ch11>) and its reverse transformation, as well as the letter-value based estimates (Hoaglin 1985), are also provided.

License GPL-2

Depends R (>= 4.3.0)

Imports rstpm2, stats

Suggests fitdistrplus

Encoding UTF-8

Language en-US

RoxygenNote 7.3.1

NeedsCompilation no

Author Tingting Zhan [aut, cre, cph], Inna Chervoneva [ctb]

Maintainer Tingting Zhan <tingtingzhan@gmail.com>

Repository CRAN

Date/Publication 2024-03-23 11:20:04 UTC

R topics documented:

GH2z *Inverse of Tukey g-&-h Transformation*

Description

To transform Tukey g-&-h quantiles to standard normal quantiles.

Usage

```
GH2z(q, q0 = (q - A)/B, A = 0, B = 1, ...)
```

Arguments

`q`	double vector, quantiles q
`q0`	(optional) double vector, standardized quantiles $q_0 = (q - A)/B$
`A, B`	(optional) double *scalars*, location and scale parameters of Tukey g-&-h trasformation. Ignored if `q0` is provided.
`...`	parameters of internal helper function .GH2z

Details

Unfortunately, function GH2z, the inverse of Tukey g-&-h transformation, does not have a closed form and needs to be solved numerically.

For compute intensive jobs, use internal helper function .GH2z.

Value

Function GH2z returns a double vector of the same length as input `q`.

Examples

```
z = rnorm(1e3L)
all.equal.numeric(.GH2z(z2GH(z, g = .3, h = .1), g = .3, h = .1), z)
all.equal.numeric(.GH2z(z2GH(z, g = 0, h = .1), g = 0, h = .1), z)
all.equal.numeric(.GH2z(z2GH(z, g = .2, h = 0), g = .2, h = 0), z)
```

letterValue — *Letter-Value Estimation of Tukey g-&-h Distribution*

Description

Letter-value based estimation (Hoaglin, 1985) of Tukey g-, h- and g-&-h distribution. All equation numbers mentioned below refer to Hoaglin (1985).

Usage

```
letterValue(
  x,
  g_ = seq.int(from = 0.15, to = 0.25, by = 0.005),
  h_ = seq.int(from = 0.15, to = 0.35, by = 0.005),
  halfSpread = c("both", "lower", "upper"),
  ...
)
```

Arguments

`x`	double vector, one-dimensional observations
`g_`	double vector, probabilities used for estimating g parameter. Or, use `g_ = FALSE` to implement the constraint $g = 0$ (i.e., an h-distribution is estimated).
`h_`	double vector, probabilities used for estimating h parameter. Or, use `h_ = FALSE` to implement the constraint $h = 0$ (i.e., a g-distribution is estimated).
`halfSpread`	character scalar, either to use `'both'` for half-spreads (default), `'lower'` for half-spread, or `'upper'` for half-spread.
`...`	additional parameters, currently not in use

Details

Unexported function `letterV_g()` estimates parameter g using equation (10) for g-distribution and the equivalent equation (31) for g-&-h distribution.

Unexported function `letterV_B()` estimates parameter B for Tukey g-distribution (i.e., $g \neq 0$, $h = 0$), using equation (8a) and (8b).

Unexported function `letterV_Bh_g()` estimates parameters B and h when $g \neq 0$, using equation (33).

Unexported function `letterV_Bh()` estimates parameters B and h for Tukey h-distribution, i.e., when $g = 0$ and $h \neq 0$, using equation (26a), (26b) and (27).

Function letterValue plays a similar role as `fitdistrplus:::start.arg.default`, thus extends `fitdistrplus::fitdist` for estimating Tukey g-&-h distributions.

Value

Function letterValue returns a `'letterValue'` object, which is double vector of estimates $(\hat{A}, \hat{B}, \hat{g}, \hat{h})$ for a Tukey g-&-h distribution.

Note

Parameter g_ and h_ does not have to be truly unique; i.e., all.equal elements are allowed.

References

Hoaglin, D.C. (1985). Summarizing Shape Numerically: The g-and-h Distributions. doi:10.1002/9781118150702.ch11

Examples

```
set.seed(77652); x = rGH(n = 1e3L, g = -.3, h = .1)
letterValue(x, g_ = FALSE, h_ = FALSE)
letterValue(x, g_ = FALSE)
letterValue(x, h_ = FALSE)
(m3 = letterValue(x))

library(fitdistrplus)
fit = fitdist(x, distr = 'GH', start = as.list.default(m3))
plot(fit) # fitdistrplus:::plot.fitdist
```

TukeyGH *Tukey g-&-h Distribution*

Description

Density, distribution function, quantile function and simulation for Tukey g-&-h distribution with location parameter A, scale parameter B, skewness g and elongation h.

Usage

```
dGH(x, A = 0, B = 1, g = 0, h = 0, log = FALSE, ...)

rGH(n, A = 0, B = 1, g = 0, h = 0)

qGH(p, A = 0, B = 1, g = 0, h = 0, lower.tail = TRUE, log.p = FALSE)

pGH(q, A = 0, B = 1, g = 0, h = 0, lower.tail = TRUE, log.p = FALSE, ...)
```

Arguments

x, q	double vector, quantiles
A	double scalar, location parameter $A = 0$ by default
B	double scalar, scale parameter $B > 0$. Default $B = 1$
g	double scalar, skewness parameter $g = 0$ by default (i.e., no skewness)
h	double scalar, elongation parameter $h \geq 0$. Default $h = 0$ (i.e., no elongation)

`log, log.p`	logical scalar, if `TRUE`, probabilities p are given as $\log(p)$.
`...`	other parameters of function vuniroot2
`n`	integer scalar, number of observations
`p`	double vector, probabilities
`lower.tail`	logical scalar, if `TRUE` (default), probabilities are $Pr(X \leq x)$ otherwise, $Pr(X > x)$.

Value

Function dGH returns the density and accommodates vector arguments `A`, `B`, `g` and `h`. The quantiles `x` can be either vector or matrix. This function takes about 1/5 time of `gk::dgh`.

Function pGH returns the distribution function, only taking scalar arguments and vector quantiles q. This function takes about 1/10 time of function `gk::pgh`.

Function qGH returns the quantile function, only taking scalar arguments and vector probabilities p.

Function rGH generates random deviates, only taking scalar arguments.

Examples

```
(x = c(NA_real_, rGH(n = 5L, g = .3, h = .1)))
dGH(x, g = c(0,.1,.2), h = c(.1,.1,.1))

p0 = seq.int(0, 1, by = .2)
(q0 = qGH(p0, g = .2, h = .1))
range(pGH(q0, g = .2, h = .1) - p0)

q = (-2):3; q[2L] = NA_real_; q
(p1 = pGH(q, g = .3, h = .1))
range(qGH(p1, g = .3, h = .1) - q, na.rm = TRUE)
(p2 = pGH(q, g = .2, h = 0))
range(qGH(p2, g = .2, h = 0) - q, na.rm = TRUE)

curve(dGH(x, g = .3, h = .1), from = -2.5, to = 3.5)
```

`vuniroot2` *Vectorised One Dimensional Root (Zero) Finding*

Description

To solve a monotone function $y = f(x)$ for a given vector of y values.

Usage

```
vuniroot2(
  y,
  f,
  interval = stop("must provide a length-2 `interval`"),
  tol = .Machine$double.eps^0.25,
  maxiter = 1000L
)
```

Arguments

y	numeric vector of y values
f	monotone function $f(x)$ whose roots are to be solved
interval	length-2 numeric vector
tol	double scalar, desired accuracy, i.e., convergence tolerance
maxiter	integer scalar, maximum number of iterations

Details

Function vuniroot2, different from vuniroot, does

- accept `NA_real_` as element(s) of y
- handle the case when the analytic root is at `lower` and/or `upper`
- return a root of `Inf` (if `abs(f(lower))` >= `abs(f(upper)))` or `-Inf` (if `abs(f(lower))` < `abs(f(upper))`), when the function value `f(lower)` and `f(upper)` are not of opposite sign.

Value

Function vuniroot2 returns a numeric vector x as the solution of $y = f(x)$ with given vector y.

Examples

```
library(rstpm2)
stopifnot(packageDate('rstpm2') == as.Date('2023-12-03')) # not base::identical
lwr = rep(1, times = 9L); upr = rep(3, times = 9L)

# ?rstpm2::vuniroot does not accept NA \eqn{y}
tryCatch(vuniroot(function(x) x^2 - c(NA, 1:8), lower = lwr, upper = upr), error = identity)

# ?rstpm2::vuniroot not good when the analytic root is at `lower` or `upper`
f <- function(x) x^2 - 1:9
tryCatch(vuniroot(f, lower = lwr, upper = upr, extendInt = 'no'), warning = identity)
tryCatch(vuniroot(f, lower = lwr, upper = upr, extendInt = 'yes'), warning = identity)
tryCatch(vuniroot(f, lower = lwr, upper = upr, extendInt = 'downX'), error = identity)
tryCatch(vuniroot(f, lower = lwr, upper = upr, extendInt = 'upX'), warning = identity)

vuniroot2(c(NA, 1:9), f = function(x) x^2, interval = c(1, 3)) # all good
```

z2GH *Tukey g-&-h Transformation*

Description

To transform standard normal quantiles to Tukey g-&-h quantiles.

Usage

```
z2GH(z, A = 0, B = 1, g = 0, h = 0)
```

Arguments

`z`	double scalar or vector, standard normal quantiles.
`A, B, g, h`	double scalar or vector, parameters of Tukey g-&-h distribution

Details

Function z2GH transforms standard normal quantiles to Tukey g-&-h quantiles.

Value

Function z2GH returns a double scalar or vector.

Note

Function `gk:::z2gh` is not fully vectorized, i.e., cannot take vector `z` *and* vector `A/B/g/h`, as of 2023-07-20 (package gk version 0.6.0)

Index